aws

Guia do desenvolvedor

Amazon DocumentDB



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

Amazon DocumentDB: Guia do desenvolvedor

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

As marcas comerciais e imagens de marcas da Amazon não podem ser usadas no contexto de nenhum produto ou serviço que não seja da Amazon, nem de qualquer maneira que possa gerar confusão entre os clientes ou que deprecie ou desprestigie a Amazon. Todas as outras marcas comerciais que não pertencem à Amazon pertencem a seus respectivos proprietários, que podem ou não ser afiliados, patrocinados pela Amazon ou ter conexão com ela.

Table of Contents

O que é Amazon DocumentDB 1
Visão geral do 1
Clusters 3
Instâncias 4
Regiões e AZs 8
Regiões 8
Zonas de disponibilidade 9
Preços 11
Teste gratuito 12
Monitoramento 12
Interfaces 13
Console de gerenciamento da AWS 13
AWS CLI 13
Drivers do MongoDB 13
Próximas etapas 13
Como funciona 14
Versões do Amazon DocumentDB 14
Endpoints do Amazon DocumentDB 17
Suporte a TLS 20
Armazenamento do Amazon DocumentDB 21
Replicação do Amazon DocumentDB 22
Confiabilidade do Amazon DocumentDB 22
Opções de preferência de leitura 23
Exclusões de TTL 28
Recursos faturáveis 29
O que é um banco de dados de documentos? 32
Casos de uso 32
Noções básicas sobre documentos 33
Trabalhar com documentos 39
Guia de conceitos básicos 52
Pré-requisitos 52
Etapa 1: criar um cluster 54
Etapa 2: conectar-se ao seu cluster 57
Etapa 3: inserir e consultar dados 58

Etapa 4: explorar 60
Início rápido de uso CloudFormation 60
Pré-requisitos 61
Lançamento de um Amazon DocumentDB CloudFormation pilha 62
Acessar o cluster do Amazon DocumentDB 67
Proteção contra encerramento e exclusão 68
Migração e atualização 69
Guia de início rápido 69
Preparação do arquivo de origem 70
Configuração do DMS 70
Habilitação da compactação do DocumentDB 70
Criação de uma tarefa de replicação 70
Monitorar o andamento 71
Mais informações 71
Runbook de migração 72
Compatibilidade 73
Descoberta de workload 75
Criar novo cluster de destino 79
Migração de índices 79
Migração do usuário 80
Migrações de dados 82
Monitoramento 93
Validação 95
Migração do Couchbase Server 101
Introdução 101
Comparação com o Amazon DocumentDB 102
Descoberta 104
Planejamento 110
Migração 113
Validação 122
Datas de suporte da versão do mecanismo 125
Calendário de lançamento para as principais versões do Amazon DocumentDB 126
Calendário de lançamento para versões secundárias do Amazon DocumentDB 126
Atualizando uma versão principal do motor 127
Caminhos de atualização com suporte 128
Pré-requisitos 129

Práticas recomendadas 131
O que muda após a atualização para o Amazon DocumentDB 8.0 134
Post-upgrade considerações para clusters atualizados da versão 3.6 ou 4.0 135
Executar a atualização 136
Solução de problemas 138
Atualizando uma versão secundária do motor 139
Antes de realizar um upgrade da versão secundária 139
Realizar uma atualização da versão secundária 140
Verificando a atualização da versão secundária 141
Atualizando usando AWS DMS 141
Etapa 1: ativar fluxos de alterações 142
Etapa 2: modificar a duração da retenção dos fluxos de alterações 142
Etapa 3: migrar seus índices 142
Etapa 4: criar um AWS DMS instância de replicação 144
Etapa 5: criar um AWS DMS endpoint de origem 145
Etapa 6: criar um AWS DMS endpoint alvo 148
Etapa 7: criar e executar uma tarefa de migração 150
Etapa 8: alterar o endpoint da aplicação para o cluster Amazon DocumentDB de destino 152
Usar um release de de suporte de longo prazo (LTS) 152
Versão atual do LTS 152
Quem deve usar o LTS 153
Como criar um novo cluster LTS 153
Como fazer o upgrade da versão 3.6.0 ou 4.0.0 do motor para um cluster LTS 5.0.0 154
Como verificar se meu cluster existente da versão 5.0.0 do motor está no LTS 154
Suporte estendido 155
Visão geral do Suporte estendido 156
Cobranças do Suporte estendido 156
Responsabilidades 157
Características e configurações 158
Status do motor 158
Tipos de cluster 158
Recursos 159
Tipos de instância 160
AWS integrações 161
Escolher regiões e zonas de disponibilidade 162
Disponibilidade de regiões 163

OpenSearch integração ... 166
Amazon OpenSearch Service como destino ... 166
Limitações ... 170
Desenvolvendo ... 171
Conecte-se ao DocumentDB ... 171
Conexão programática ... 172
Conectar-se de fora de uma Amazon VPC ... 202
Conectar-se como um conjunto de réplicas ... 204
Conectar usando o Studio 3T ... 209
Conecte-se usando DataGrip ... 218
Conecte usando o Amazon EC2 ... 223
Conecte-se usando o driver JDBC ... 261
Conecte-se usando o driver ODBC ... 277
Programação com o Amazon DocumentDB ... 291
Guia de programação em Java do DocumentDB ... 291
Usando a validação do esquema JSON ... 324
Consultas ... 334
Consultar documentos ... 335
Plano de consulta ... 338
Explicar os resultados ... 340
Planejador de consultas v2 ... 345
Planejador de consultas v3 ... 360
Dados geoespaciais ... 366
Índice parcial ... 372
Busca de texto ... 379
Índices ... 387
Gerenciamento de índices ... 388
Tipos de índice ... 399
Propriedades de índice ... 408
Usar fluxos de alterações ... 427
Operações compatíveis ... 428
Eventos expandidos ... 429
Faturamento ... 430
Limitações ... 430
Ativar fluxos de alterações ... 431
Usar fluxos de alterações com Python ... 433

Pesquisa completa de documentos ... 435
Usando $change StreamSplitLargeEvent para dividir grandes eventos em fragmentos ... 436
Retomar um fluxo de alterações ... 437
Retomar com `startAtOperationTime` ... 439
Retomar com `postBatchResumeToken` ... 440
Transações ... 442
Modificar a duração da retenção do log ... 442
Fluxos de alterações em instâncias secundárias ... 445
Transações ... 447
Requisitos ... 447
Práticas recomendadas ... 448
Limitações ... 448
Monitoramento e diagnóstico ... 449
Nível de isolamento de transação ... 450
Casos de uso ... 450
Comandos compatíveis ... 488
Capacidades sem suporte ... 488
Sessões ... 489
Erros de transação ... 491
Gravações repetíveis ... 491
Requisitos ... 492
Operações compatíveis ... 492
Limitações ... 493
Habilitando ... 493
Como funciona ... 494
Tratamento de erros ... 494
Migração ... 495
Usando agrupamento ... 495
Limitações ... 496
Uso das visualizações do ... 496
Práticas recomendadas ... 497
Compatibilidade do operador agregador ... 498
Aproveitando índices e visualizações ... 498
IA generativa ... 498
SageMaker Tela ... 499
Pesquisa vetorial ... 502

Utilizar AWS Lambda com fluxos de mudança ... 514
Limitações ... 515
Compatibilidade com o MongoDB ... 516
Compatibilidade com MongoDB 8.0 ... 516
O que há de novo no Amazon DocumentDB 8.0 ... 516
Conceitos básicos do Amazon DocumentDB ... 517
Atualize ou migre para o Amazon DocumentDB 5.0 ou 8.0 ... 518
Diferenças funcionais ... 518
Compatibilidade com MongoDB 5.0 ... 519
Novidades do Amazon DocumentDB 5.0 ... 519
O que há de novo no Amazon DocumentDB 5.0.1 ... 520
Conceitos básicos do Amazon DocumentDB 5.0 ... 517
Atualize ou migre para o Amazon DocumentDB 5.0 ... 518
Diferenças funcionais ... 518
Compatibilidade do MongoDB 4.0 ... 523
Atributos do Amazon DocumentDB 4.0 ... 523
Conceitos básicos do Amazon DocumentDB 4.0 ... 524
Atualizar ou migrar para o Amazon DocumentDB 4.0 ... 525
Diferenças funcionais ... 525
Diferenças funcionais com o MongoDB ... 526
Benefícios funcionais do Amazon DocumentDB ... 526
Diferenças funcionais atualizadas ... 528
Diferenças funcionais com o MongoDB ... 531
APIs, operações e tipos de dados do MongoDB compatíveis ... 541
Comandos do banco de dados ... 542
Operadores de consulta e projeção ... 551
Operadores de atualização ... 555
Geoespacial ... 557
Métodos de cursor ... 558
Operadores de pipeline de agregação ... 561
Tipos de dados ... 578
Índices e propriedades de índice ... 579
Operadores de pipeline de agregação ... 581
Geoespacial ... 1215
Operadores de consulta e projeção ... 1239
Operadores de atualização ... 1308

Gerenciando clusters e instâncias 1363
Visão geral de tarefas operacionais 1363
Adicionar uma réplica a um cluster do Amazon DocumentDB 1364
Descrever clusters e instâncias 1365
Criar um snapshot de cluster 1367
Restauração a partir de um snapshot 1368
Remover uma instância de um cluster 1369
Excluir um cluster 1369
Clusters globais 1370
O que é um cluster global? 1370
Por que os clusters globais são úteis? 1370
Quais são as limitações atuais dos clusters globais? 1371
Guia de início rápido 1372
Gerenciar clusters globais 1384
Conectar clusters globais 1392
Monitorar clusters globais 1392
Recuperação de desastres 1393
Gerenciamento de clusters do 1418
Noções básicas de clusters 1419
Configurações do cluster 1422
Configurações de armazenamento de cluster 1424
Determinar o status de um cluster 1427
Ciclo de vida do cluster 1429
Escalar clusters 1472
Clonar um volume para um cluster 1475
Entender a tolerância a falhas de cluster 1488
Gerenciar instâncias 1490
Determinar o status de uma instância 1490
Ciclo de vida da instância 1490
Gerenciar classes de instância 1514
NVMe-backed instâncias 1527
Gerenciamento de grupos de sub-redes 1530
Criação de um grupo de sub-redes 1531
Como descrever um grupo de sub-redes 1537
Modificação de um grupo de sub-redes 1540
Exclusão de um grupo de sub-redes 1543

Alta disponibilidade e replicação 1544
Escalabilidade de leitura 1545
Alta disponibilidade 1545
Adicionar réplicas do 1547
Failover 1547
Lag de replicação 1552
Gerenciamento de eventos 1553
Exibição de categorias de eventos 1554
Visualizando eventos do Amazon DocumentDB 1556
Gerenciando grupos de parâmetros de cluster 1559
Descrevendo grupos de parâmetros de cluster 1560
Criando grupos de parâmetros de cluster 1568
Modificando grupos de parâmetros de cluster 1571
Modificando clusters para usar grupos de parâmetros de cluster personalizados 1576
Copiando grupos de parâmetros de cluster 1577
Redefinindo grupos de parâmetros de cluster 1579
Excluindo grupos de parâmetros de cluster 1582
Referência de parâmetros de cluster 1585
Noções básicas sobre endpoints 1608
Localizar os endpoints de um cluster 1609
Localizar o endpoint de uma instância 1611
Conectar-se a endpoints 1614
Compreender os ARNs do Amazon DocumentDB 1616
Criação de um ARN 1616
Localizar um ARN 1620
Tagging resources 1623
Visão geral de tags de recurso do 1623
Restrições de tag 1624
Adicionar ou atualizar tags 1625
Listar tags 1626
Remover tags 1628
Manutenção do Amazon DocumentDB 1629
Ações de manutenção para o Amazon DocumentDB 1630
Numeração de versões 1631
Gerenciar janelas de manutenção. 1632
Notificações de patch do mecanismo 1635

Visualizar manutenção pendente ... 1636
Atualizações do mecanismo ... 1640
Versões secundárias ... 1643
Atualizações do sistema operacional ... 1643
User-initiated atualizações ... 1648
Clusters globais ... 1649
Noções básicas dos perfis vinculados ao serviço ... 1650
Service-linked permissões de função ... 1650
Criar um perfil vinculado ao serviço ... 1652
Modificar um perfil vinculado ao serviço ... 1652
Excluir um perfil vinculado ao serviço ... 1653
Regiões suportadas para funções vinculadas ao serviço Amazon DocumentDB ... 1654
Otimização de desempenho ... 1655
Problemas comuns de desempenho ... 1655
CPU de alta performance ... 1655
Problemas de conexão ... 1657
Consulta em execução lenta ... 1664
Anti-patterns ... 1669
Índice composto antipadrão com mais de 3 atributos ... 1669
Consultas de longa duração ... 1672
Índices pouco usados e redundantes ... 1674
Multi-key índices com matrizes grandes ... 1677
Dicas de melhoria de desempenho ... 1678
1. Use $match como primeiro estágio em pipelines de agregação ... 1678
2. Use $project no pipeline de agregação para minimizar o tamanho dos dados do pipeline ... 1679
3. Ative a compactação de documentos para reduzir os custos e I/O custos de armazenamento e melhorar o desempenho das consultas ... 1679
4. Utilize índices para um desempenho de consulta ideal ... 1680
5. Otimize modelos de dados com base em padrões de consulta ... 1681
Análise do plano de consulta ... 1683
Cronometragem geral ... 1685
Etapas de execução ... 1685
Compreendendo os documentos examinados em Explique as estatísticas de execução do plano ... 1687
Index-only escaneia ... 1695

Quando o Amazon DocumentDB usa um escaneamento somente de índice 1696
Requisitos para uma consulta coberta 1698
Restrições 1699
Confirmando uma verificação somente de índice 1699
Práticas recomendadas 1701
Planejar filtros de cache 1701
Comandos compatíveis 1702
Formas de consulta suportadas 1704
Configurando, listando e limpando filtros 1704
Verificando se um filtro foi aplicado 1706
Exemplos 1707
Comportamento com dicas e alterações no índice 1710
Práticas recomendadas 1710
Diretrizes operacionais básicas 1711
Dimensionamento de instância 1712
Trabalhar com índices 1714
Práticas recomendadas de segurança 1715
Otimização de custos 1715
Uso de métricas para identificar problemas de performance 1716
Workloads de TTL e séries temporais 1720
Migrações 1721
Trabalhar com grupos de parâmetros de cluster 1721
Consultas de pipeline de agregação 1722
`BatchInsert` e BatchUpdate 1722
Gerenciamento de Índices 1722
Criação do índice do Amazon DocumentDB 1722
Manutenção de índices do Amazon DocumentDB 1729
Gerenciamento da compactação de documentos 1731
Gerenciamento da compactação de documentos 1732
Monitoramento da compactação de documentos 1734
Gerenciando a compactação baseada em dicionário 1735
Considerações sobre a performance 1736
Habilitando a compactação baseada em dicionário 1736
Conceitos básicos 1737
Monitoramento 1737
Segurança 1739

Gerenciamento de senhas com AWS Secrets Manager 1740
Limitações da integração do Secrets Manager com o Amazon DocumentDB 1741
Visão geral do gerenciamento de senhas de usuários primários com AWS Secrets Manager 1741
Impondo o gerenciamento do Amazon DocumentDB da senha do usuário principal em AWS Secrets Manager 1742
Gerenciamento da senha do usuário principal para um cluster com o Secrets Manager 1742
Proteção de dados 1743
Client-side criptografia em nível de campo 1744
Criptografar dados em repouso 1751
Criptografia de dados em trânsito 1759
Gerenciamento de chaves 1770
Gerenciamento de Identidade e Acesso 1771
Público 1772
Autenticação com identidades 1772
Gerenciar o acesso usando políticas 1773
Como o Amazon DocumentDB funciona com o IAM 1775
Identity-based exemplos de políticas 1782
Solução de problemas 1785
Gerenciar permissões de acesso aos recursos do Amazon DocumentDB 1787
Usar políticas baseadas em identidade (políticas do IAM) 1792
AWS políticas gerenciadas para o Amazon DocumentDB 1797
Referência de permissões da API do Amazon DocumentDB 1816
Cross-service prevenção adjunta confusa 1825
Autenticação usando identidade do IAM 1826
Conceitos básicos dos usuários e das funções do IAM 1827
Configurar tipos de computação 1830
Monitorar solicitações de autenticação 1831
Usar a autenticação do IAM 1831
Drivers que oferecem suporte ao IAM 1832
Perguntas frequentes sobre autenticação de identidade do IAM 1832
Autenticação usando SCRAM-SHA-256 1833
Criação de usuários do 1833
Atualizar usuários 1835
Autenticando 1835
Gerenciar usuários do Amazon DocumentDB 1835

Usuário primário e `administrador de serviços` ... 1836
Criar usuários adicionais ... 1836
Alternância automática de senhas ... 1839
Role-Based Controle de acesso ... 1839
Conceitos do RBAC ... 1840
Introdução às funções integradas do RBAC ... 1842
Introdução às funções definidas pelo usuário do RBAC ... 1846
Conectar-se ao Amazon DocumentDB como um usuário ... 1850
Comandos comuns ... 1852
Diferenças funcionais ... 1857
Limites ... 1857
Acesso ao banco de dados usando o controle de Role-Based acesso ... 1858
Atualizar certificados ... 1867
Atualizar sua aplicação e seu cluster do Amazon DocumentDB ... 1868
Alternância automática de certificados do servidor ... 1872
Solução de problemas ... 1873
Perguntas frequentes ... 1874
Validação de conformidade ... 1880
Resiliência ... 1882
Segurança da infraestrutura ... 1883
Endpoints da VPC (AWS PrivateLink) ... 1884
Considerações sobre endpoints da VPC do ... 1885
Disponibilidade de regiões ... 1885
Como criar um endpoint da VPC de interface para a API do Amazon DocumentDB ... 1886
Criação de uma política de endpoint de VPC para a API Amazon DocumentDB ... 1886
Práticas recomendadas de segurança ... 1888
Auditoria de eventos ... 1889
Eventos com suporte ... 1890
Ativar a auditoria ... 1896
Desativar a auditoria ... 1902
Acessar seus eventos ... 1905
Filtrar eventos de DML ... 1906
Amazon VPC e o Amazon DocumentDB ... 1911
Clusters do DocumentDB em uma VPC ... 1912
Acesso a um cluster em uma VPC ... 1925
Crie uma IPv4-only VPC para um cluster ... 1927

Criação de uma VPC de pilha dupla para um cluster ... 1934
Fazer backup e restaurar ... 1945
Backup e restauração: conceitos ... 1946
Noções básicas do uso do armazenamento de backup do ... 1948
Despejo, restauração, importação e exportação de dados ... 1950
mongodump ... 1950
mongorestore ... 1951
mongoexport ... 1951
importação de mongo ... 1952
Tutorial ... 1953
Considerações sobre snapshot de cluster ... 1955
Armazenamento de backup ... 1956
Janela de backup ... 1956
Período de retenção de backup ... 1959
Copiar criptografia de snapshot de cluster ... 1959
Comparação dos snapshots automáticos e manuais ... 1960
Criação de um snapshot manual de cluster ... 1962
Cópia de um snapshot de cluster ... 1965
Copiar snapshots compartilhados ... 1966
Copiando instantâneos em Regiões da AWS ... 1966
Limitações ... 1966
Lidar com a criptografia ... 1966
Considerações de grupos de parâmetros ... 1967
Cópia de um snapshot de cluster ... 1967
Compartilhamento de um snapshot de cluster ... 1977
Compartilhar um snapshot criptografado ... 1978
Compartilhar um snapshot ... 1981
Restauração de um snapshot de cluster ... 1985
Restauração para um ponto no tempo ... 1992
Exclusão de um snapshot de cluster ... 1998
Monitorar e registrar em log ... 2001
Monitorar o status de um cluster ... 2002
Valores de status do cluster ... 2003
Monitorar o status de um cluster ... 2005
Monitorar o status de uma instância ... 2006
Valores de status de instâncias ... 2007

Monitorando o status da instância usando o Console de gerenciamento da AWS or AWS CLI ... 2010
Status de integridade da instância ... 2012
Monitorando o status de integridade da instância usando o Console de gerenciamento da AWS ... 2012
Visualização das recomendações do Amazon DocumentDB ... 2014
Assinaturas de eventos ... 2017
Assinar eventos ... 2018
Gerenciar assinaturas do ... 2020
Categorias e mensagens ... 2024
Monitorando o Amazon DocumentDB com CloudWatch ... 2027
Métricas do Amazon DocumentDB ... 2027
Visualização CloudWatch de dados ... 2047
Dimensões do Amazon DocumentDB ... 2053
Monitoramento de métricas de contadores ... 2053
Monitorar conexões de banco de dados ... 2054
Registro de chamadas de API do Amazon DocumentDB com CloudTrail ... 2054
Informações do Amazon DocumentDB em CloudTrail ... 2055
Definir o perfil das operações ... 2056
Operações compatíveis ... 2057
Limitações ... 2057
Ativar o profiler ... 2057
Desativar o profiler ... 2062
Desativar a exportação de logs do profiler ... 2063
Acessar seus logs do profiler ... 2065
Consultas comuns ... 2066
Monitoramento com o Performance Insights ... 2066
Conceitos de Performance Insights ... 2068
Ativar e desativar o Performance Insights ... 2072
Configurar políticas de acesso para o Performance Insights ... 2075
Análise de métricas usando o painel do Performance Insights ... 2079
Recuperar métricas com a API do Performance Insights ... 2099
CloudWatch Métricas da Amazon para insights de desempenho ... 2114
Métricas de contadores do Performance Insights ... 2117
Sem servidor ... 2119
Casos de uso da tecnologia sem servidor ... 2119

Uso do Amazon DocumentDB Sem Servidor para workloads provisionadas existentes 2121
Vantagens do Amazon DocumentDB Sem Servidor 2122
Como a tecnologia sem servidor funciona 2123
Visão geral do 2123
Configurações para clusters do Amazon DocumentDB 2125
Capacidade de escalabilidade Amazon DocumentDB Sem Servidor 2125
Escalabilidade do Amazon DocumentDB Sem Servidor 2127
Estado ocioso (0,5 DCUs) 2129
Requisitos e limitações da tecnologia sem servidor 2130
Requisitos de atributos 2130
Criação de um cluster de banco de dados que use tecnologia sem servidor 2131
Criação de um cluster do Amazon DocumentDB Sem Servidor 2132
Adição de uma instância do Amazon DocumentDB Sem Servidor 2135
Migração para a tecnologia sem servidor 2135
Migração de clusters do DocumentDB existentes para a tecnologia sem servidor 2136
Migração do MongoDB para o DocumentDB Sem Servidor 2141
Gerenciamento da tecnologia sem servidor 2141
Visualização e modificação da configuração do intervalo de capacidade de escalabilidade de um cluster 2141
Visualização e modificação do nível de promoção de leitores com tecnologia sem servidor 2145
Limites das instâncias com tecnologia sem servidor 2147
Configuração de escalabilidade com tecnologia sem servidor 2150
Escolha do intervalo de capacidade de escalabilidade para um cluster do DocumentDB Sem Servidor 2150
Escolha da configuração `MinCapacity` para um cluster do DocumentDB Sem Servidor .. 2150
Escolha da configuração `MaxCapacity` para um cluster do DocumentDB Sem Servidor .. 2153
Evitar erros de falta de memória 2155
Por que minha instância com tecnologia sem servidor não está reduzindo a escala verticalmente? 2156
Monitoramento da tecnologia sem servidor 2157
Sem memória: status de parâmetros incompatíveis 2158
CloudWatch Métricas da Amazon para DocumentDB sem servidor 2158
Monitoramento da performance do DocumentDB Sem Servidor com Insights de performance 2163
Clusters elásticos 2164
Casos de uso do cluster elástico 2165

Perfis de usuário ... 2165
Gerenciamento de conteúdo e registros históricos ... 2165
Vantagens dos clusters elásticos ... 2165
AWS integração de serviços ... 2165
Disponibilidade de regiões e versões ... 2166
Disponibilidade regional para clusters elásticos ... 2166
Disponibilidade da versão ... 2167
Limitações ... 2167
Gerenciamento de clusters elásticos ... 2167
Operações de gravação e de consulta ... 2168
Gerenciamento de coleções e índices ... 2168
Administração e diagnóstico ... 2169
Opt-in features ... 2169
Como funciona ... 2169
Fragmentação de clusters elásticos do Amazon DocumentDB ... 2170
Migração de clusters elásticos ... 2173
Escalabilidade dos clusters elásticos ... 2173
Confiabilidade dos clusters elásticos ... 2173
Armazenamento e disponibilidade de clusters elásticos ... 2174
Diferenças funcionais entre o Amazon DocumentDB 4.0 e clusters elásticos ... 2174
Conceitos básicos ... 2176
Pré-requisitos ... 2176
Etapa 1: Criar um cluster do ElastiCache ... 2178
Etapa 2: conectar a seu cluster elástico ... 2184
Etapa 3: fragmentar sua coleção; inserir e consultar dados ... 2185
Etapa 4: explorar ... 2188
Práticas recomendadas ... 2188
Selecionar chaves de fragmento ... 2189
Gerenciamento de conexões ... 2189
Coleções não fragmentadas ... 2189
Escalar clusters elásticos ... 2190
Monitoramento de clusters elásticos ... 2190
Gerenciar clusters elásticos ... 2191
Como modificar configurações de clusters elásticos ... 2191
Como monitorar um cluster elástico ... 2195
Como excluir um cluster elástico ... 2199

Como gerenciar snapshots de cluster elástico ... 2201
Interromper e iniciar um cluster elástico ... 2216
Manutenção de clusters elásticos ... 2221
Criptografia de dados em repouso ... 2229
Como os clusters elásticos do Amazon DocumentDB usam concessões em AWS KMS ... 2231
Criar uma chave gerenciada pelo cliente ... 2231
Monitorar suas chaves de criptografia para clusters elásticos do Amazon DocumentDB ... 2233
Saiba mais ... 2238
Service-linked papéis ... 2239
Service-linked permissões de função para clusters elásticos ... 2239
Cotas ... 2244
Tipos de instâncias compatíveis ... 2244
Regiões aceitas ... 2246
Cotas regionais ... 2248
Cotas de agregação ... 2251
Cotas de cluster ... 2251
Cotas de instâncias ... 2253
Restrições de nomenclatura ... 2260
Restrições de TTL ... 2262
Cotas elásticas de clusters ... 2262
Cotas elásticas de fragmentos de cluster ... 2264
Cotas de CPU, memória, conexão e cursor de cluster elástico por fragmento ... 2264
Solução de problemas ... 2266
Problemas de conectividade ... 2266
Não é possível conectar-se a um endpoint do Amazon DocumentDB ... 2266
Testar uma conexão com uma instância do Amazon DocumentDB ... 2272
Conectar a um endpoint inválido ... 2272
A configuração do driver afeta o número de conexões ... 2273
Índices ... 2273
Criação de índice fracassa ... 2273
Problemas e falhas de latência de criação de índice em segundo plano ... 2274
Sobrecarga de índices ... 2274
Desempenho e utilização de recursos ... 2276
Exibir estatísticas de inserção, atualização e exclusão ... 2276
Analise o desempenho do cache ... 2278
Encontre e encerre consultas bloqueadas ou de longa duração ... 2279

Veja um plano de consulta e otimize uma consulta 2281
Como posso ver um plano de consulta em clusters elásticos? 2283
Listar todas as operações em execução em uma instância 2285
Saiba quando uma consulta está progredindo 2288
Determine por que um sistema de repente funciona lentamente 2291
Determinar a causa da alta utilização da CPU 2292
Encontre os cursores abertos em uma instância 2293
Veja a versão atual do mecanismo Amazon DocumentDB 2294
Analise o uso do índice e identifique índices não utilizados 2294
Identifique índices ausentes 2296
Como faço para determinar a sobrecarga da coleção de banco de dados? 2298
Resumo de consultas úteis 2299
Coleta de resíduos 2301
Entendendo a coleta de lixo no Amazon DocumentDB 2301
Processo de coleta de resíduos 2302
Arquitetura de armazenamento e armazenamento estendido 2302
Monitoramento da coleta de resíduos 2304
Exemplo de saída CollStats 2306
Perguntas frequentes 2308
Códigos de erro 2311
Notas da versão 2345
Notas de lançamento da versão do mecanismo Amazon DocumentDB 2346
Notas gerais de lançamento do Amazon DocumentDB 2412
Notas de lançamento dos clusters elásticos do Amazon DocumentDB 2417
Histórico do documento 2422
..... mmcdxxxv

O que é Amazon DocumentDB (compatível com MongoDB)

O Amazon DocumentDB (compatível com MongoDB) é um serviço de banco de dados rápido, confiável e inteiramente gerenciado. O Amazon DocumentDB facilita a configuração, a operação e a escalabilidade de MongoDB-compatible bancos de dados na nuvem. Com o Amazon DocumentDB, é possível executar o mesmo código de aplicação e usar os mesmos drivers e ferramentas que você usa com o MongoDB.

Antes de usar o Amazon DocumentDB, é necessário revisar os conceitos e recursos descritos em Como funciona. Depois disso, conclua as etapas em Guia de conceitos básicos.

Tópicos

- Visão geral do Amazon DocumentDB
- Clusters
- Instâncias
- Regiões e zonas de disponibilidade
- Definição de preço do Amazon DocumentDB
- Monitoramento
- Interfaces
- Próximas etapas
- Amazon DocumentDB: como funciona
- O que é um banco de dados de documentos?

Visão geral do Amazon DocumentDB

A seguir estão alguns recursos gerais do Amazon DocumentDB:

- O Amazon DocumentDB oferece suporte a dois tipos de clusters: clusters baseados em instâncias e clusters elásticos. Os clusters elásticos suportam cargas de trabalho com milhões reads/writes por segundo e petabytes de capacidade de armazenamento. Para obter mais informações sobre clusters elásticos, consulte Usando clusters elásticos do Amazon DocumentDB. O conteúdo abaixo se refere aos clusters baseados em instâncias do Amazon DocumentDB.
- O Amazon DocumentDB aumenta automaticamente o tamanho do volume de armazenamento à medida que as necessidades de armazenamento do seu banco de dados aumentam. Seu

volume de armazenamento cresce em incrementos de 10 GB, até um máximo de 256 TiB para a versão 8.0 e superior do motor (128 TiB para versões anteriores do mecanismo). Você não precisa provisionar nenhum armazenamento em excesso para o seu cluster para lidar com o crescimento futuro.

- Com o Amazon DocumentDB, você pode aumentar o throughput de leitura para dar suporte a solicitações de aplicações de alto volume criando até 15 instâncias de réplica. As réplicas do Amazon DocumentDB compartilham o mesmo armazenamento subjacente, reduzindo os custos e evitando a necessidade de realizar gravações nos nós da réplica. Esse recurso libera mais capacidade de processamento para atender às solicitações de leitura e reduz a defasagem das réplicas, muitas vezes para menos de 10 milissegundos. É possível adicionar réplicas em minutos, independentemente do tamanho do volume de armazenamento. O Amazon DocumentDB também fornece um endpoint de leitura, para que a aplicação possa se conectar sem precisar rastrear réplicas à medida que elas são adicionadas e removidas.
- O Amazon DocumentDB permite aumentar ou diminuir a escala dos recursos de computação e memória para cada uma das suas instâncias. As operações de escalabilidade de computação geralmente são concluídas em poucos minutos.
- O Amazon DocumentDB é executado na Amazon Virtual Private Cloud (Amazon VPC), para que você possa isolar seu banco de dados em sua própria rede virtual. Você também pode definir configurações de firewall para controlar o acesso de rede ao cluster.
- O Amazon DocumentDB monitora continuamente a integridade do seu cluster. Em caso de falha na instância, o Amazon DocumentDB reinicia automaticamente a instância e os processos associados. O Amazon DocumentDB não exige uma repetição de recuperação de falhas dos redo logs do banco de dados, o que reduz consideravelmente os tempos de reinicialização. O Amazon DocumentDB também isola o cache do banco de dados do processo do banco de dados, permitindo que o cache sobreviva à reinicialização da instância.
- Em caso de falha na instância, o Amazon DocumentDB automatiza o failover para uma das até 15 réplicas do Amazon DocumentDB que você cria em outras zonas de disponibilidade. Se nenhuma réplica tiver sido provisionada e ocorrer uma falha, o Amazon DocumentDB tentará criar uma nova instância do Amazon DocumentDB automaticamente.
- O recurso de backup no Amazon DocumentDB permite a point-in-time recuperação do seu cluster. Esse atributo permite que você restaure seu cluster para qualquer segundo dos últimos cinco minutos do período de retenção. É possível configurar o período de retenção de backup automático para até 35 dias. Os backups automatizados são armazenados no Amazon Simple Storage Service (Amazon S3), que foi projetado para oferecer durabilidade de 99,999999999%. Os

backups do Amazon DocumentDB são automáticos, incrementais e contínuos, e não têm impacto no desempenho do seu cluster.

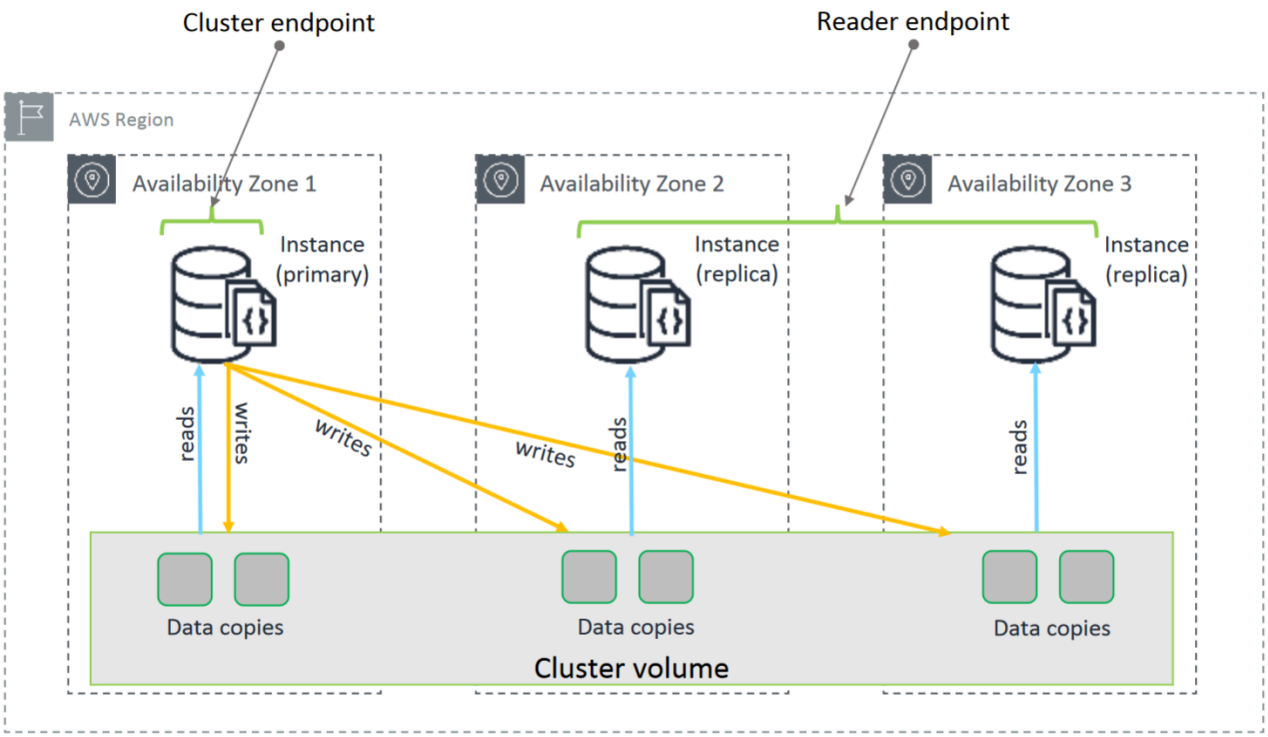
- Com o Amazon DocumentDB, você pode criptografar seus bancos de dados usando chaves que você cria e controla por meio AWS Key Management Service de ().AWS KMS Em um cluster de banco de dados executado com a criptografia do Amazon DocumentDB, os dados armazenados em repouso no armazenamento subjacente são criptografados. Os backups automatizados, snapshots e réplicas no mesmo cluster também são criptografados.
- O Amazon DocumentDB é autorizado pelo Programa Federal de Gerenciamento de Riscos e Autorizações (Federal Risk and Authorization Management Program, ou FedRAMP). Ele tem autorização FedRAMP High para regiões AWS GovCloud (EUA) e autorização FedRAMP Moderate para regiões dos EUA. AWS East/West Para obter detalhes sobre os esforços de conformidade AWS e os esforços de conformidade, consulte AWS Serviços no escopo do programa de conformidade.

Se você não conhece os AWS serviços, use os seguintes recursos para saber mais:

- AWS oferece serviços para computação, bancos de dados, armazenamento, análise e outras funcionalidades. Para obter uma visão geral de todos os AWS serviços, consulte Computação em nuvem com a Amazon Web Services.
- AWS fornece vários serviços de banco de dados. Para obter orientação sobre qual serviço é melhor para seu ambiente, consulte Bancos de dados na AWS.

Clusters

Um cluster consiste em 0 a 16 instâncias e em um volume de armazenamento de cluster que gerencia os dados para essas instâncias. Todas as gravações são feitas por meio da instância principal. Todas as instâncias (principais e de réplicas) são compatíveis com leituras. Os dados do cluster são armazenados no volume do cluster com cópias em três zonas de disponibilidade diferentes.



Os clusters baseados em instâncias do Amazon DocumentDB 5.0 oferecem suporte a duas configurações de armazenamento para um cluster de banco de dados: o padrão Amazon DocumentDB e o Amazon DocumentDB. I/O-optimized Para obter mais informações, consulte Configurações de armazenamento de cluster do Amazon DocumentDB.

Instâncias

Uma instância do Amazon DocumentDB é um ambiente de banco de dados na nuvem. Uma instância pode conter vários bancos de dados criados pelo usuário. Você pode criar e modificar uma instância usando o Console de gerenciamento da AWS ou AWS CLI o.

A capacidade de computação e a memória de uma instância são determinadas de acordo com sua classe de instância. É possível selecionar a instância que melhor atenda às suas necessidades. Se suas necessidades mudarem com o tempo, será possível escolher uma classe de instância diferente. Para conhecer as especificações de classes de instância, consulte Especificações de classe de instância.

As instâncias do Amazon DocumentDB são executadas somente no ambiente Amazon VPC. A Amazon VPC permite controlar o seu ambiente de rede virtual: você pode escolher o seu próprio intervalo de endereços IP, criar sub-redes e configurar listas de controle de acesso e roteamento (ACLs).

Antes de criar instâncias do Amazon DocumentDB, é necessário criar um cluster para conter as instâncias.

Nem todas as classes de instância são suportadas em todas Região da AWS. A tabela a seguir mostra quais classes de instância são suportadas em cada região.

> **Note**
>
> Para obter uma lista completa de tipos de instância com suporte no Amazon DocumentDB em cada classe de instância, consulte Especificações de classe de instância.

Classes de instância compatíveis por região

Região	R8G	R6GD	R6G	R5	R4	T4G	T3	Sem servidor
Leste dos EUA (Ohio)	Compatível	Compatível	Compatível	Compatível	Compatível	Compatível	Compatível	Compatível
Leste dos EUA (Norte da Virgínia)	Compatível	Compatível	Compatível	Compatível	Compatível	Compatível	Compatível	Compatível
Oeste dos EUA (Oregon)	Compatível	Compatível	Compatível	Compatível	Compatível	Compatível	Compatível	Compatível
África (Cidade do Cabo)			Compatível	Compatível		Compatível	Compatível	Compatível
América do Sul (São Paulo)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
Ásia-Pacífico (Hong Kong)	Compatível		Compatível	Compatível		Compatível	Compatível	Compatível
Ásia-Pacífico (Hyderabad)			Compatível	Compatível		Compatível	Compatível	Compatível

Região	R8G	R6GD	R6G	R5	R4	T4G	T3	Sem servidor
Ásia-Pacífico (Malásia)			Compatível			Compatível	Compatível	Compatível
Ásia-Pacífico (Mumbai)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
Ásia-Pacífico (Osaka)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
Ásia-Pacífico (Seul)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
Ásia-Pacífico (Sydney)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
Ásia-Pacífico (Jacarta)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
Ásia-Pacífico (Melbourne)			Compatível	Compatível		Compatível	Compatível	Compatível
Ásia-Pacífico (Singapura)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
Ásia-Pacífico (Tailândia)			Compatível			Compatível	Compatível	Compatível
Ásia-Pacífico (Tóquio)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
Canadá (Central)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
Oeste do Canadá (Calgary)			Compatível	Compatível		Compatível	Compatível	Compatível

Região	R8G	R6GD	R6G	R5	R4	T4G	T3	Sem servidor
Europa (Frankfurt)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
Europa (Zurique)		Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
Europa (Irlanda)	Compatível	Compatível	Compatível	Compatível	Compatível	Compatível	Compatível	Compatível
Europa (Londres)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
Europa (Milão)			Compatível	Compatível		Compatível	Compatível	Compatível
Europa (Paris)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
Europa (Espanha)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
Europa (Estocolmo)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
México (Centro)			Compatível			Compatível	Compatível	Compatível
Oriente Médio (Emirados Árabes Unidos)			Compatível	Compatível		Compatível	Compatível	Compatível
China (Pequim)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
China (Ningxia)	Compatível		Compatível	Compatível		Compatível	Compatível	Compatível

Região	R8G	R6GD	R6G	R5	R4	T4G	T3	Sem servidor
Israel (Tel Aviv)			Compatível	Compatível		Compatível	Compatível	Compatível
AWS GovCloud (US-West)	Compatível	Compatível	Compatível	Compatível			Compatível	Compatível
AWS GovCloud (US-East)		Compatível	Compatível	Compatível		Compatível	Compatível	Compatível

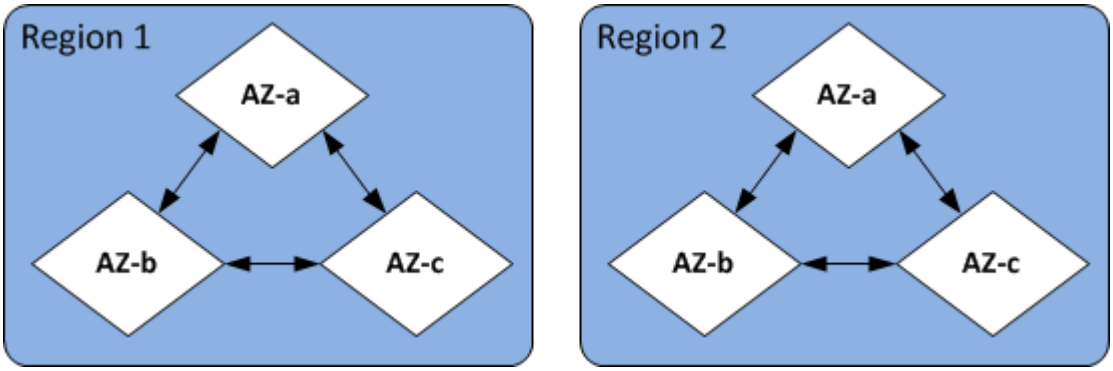
Regiões e zonas de disponibilidade

Regiões da AWS e as zonas de disponibilidade definem os locais físicos do cluster e das instâncias.

Regiões

AWS Os recursos de computação em nuvem estão alojados em instalações de data center altamente disponíveis em diferentes áreas do mundo (por exemplo, América do Norte, Europa ou Ásia). Cada localização de data center é chamada de Região da AWS.

Cada um foi Região da AWS projetado para ser completamente isolado do outro Regiões da AWS. Dentro de cada região, há várias zonas de disponibilidade. Ao iniciar seus nós em diferentes zonas de disponibilidade, você é capaz de alcançar o máximo possível de tolerância a falhas. O diagrama a seguir mostra uma visão de alto nível de como Regiões da AWS as zonas de disponibilidade funcionam.



Zonas de disponibilidade

Cada um Região da AWS contém vários locais distintos chamados de zonas de disponibilidade. Além de ser projetada para ser isolada das falhas de outras zonas de disponibilidade, cada zona de disponibilidade fornece conectividade de rede de baixa latência e baixo custo para outras zonas de disponibilidade da mesma região. Ao executar instâncias para um cluster em várias zonas de disponibilidade, você pode proteger suas aplicações contra o evento improvável de falha de uma zona de disponibilidade.

A arquitetura do Amazon DocumentDB separa armazenamento e computação. Para a camada de armazenamento, o Amazon DocumentDB replica seis cópias de seus dados em três zonas de AWS disponibilidade. Por exemplo, se você estiver executando um cluster do Amazon DocumentDB em uma região que ofereça suporte apenas a duas zonas de disponibilidade, seu armazenamento de dados será replicado de seis maneiras em três zonas de disponibilidade, mas suas instâncias de computação estarão disponíveis somente em duas zonas de disponibilidade.

A tabela a seguir lista o número de zonas de disponibilidade que você pode usar em um determinado momento Região da AWS para provisionar instâncias de computação para seu cluster.

Nome da região	Região	Zonas de disponibilidade (computação)
Leste dos EUA (Ohio)	`us-east-2`	3
Leste dos EUA (Norte da Virgínia)	`us-east-1`	6
Oeste dos EUA (Oregon)	`us-west-2`	4
África (Cidade do Cabo)	`af-south-1`	3
América do Sul (São Paulo)	`sa-east-1`	3
Ásia-Pacífico (Hong Kong)	`ap-east-1`	3

Nome da região	Região	Zonas de disponibilidade (computação)
Ásia-Pacífico (Hyderabad)	ap-south-2	3
Ásia-Pacífico (Malásia)	ap-southeast-5	3
Ásia-Pacífico (Mumbai)	ap-south-1	3
Ásia-Pacífico (Osaka)	ap-northeast-3	3
Ásia-Pacífico (Seul)	ap-northeast-2	4
Ásia-Pacífico (Singapura)	ap-southeast-1	3
Ásia-Pacífico (Sydney)	ap-southeast-2	3
Ásia-Pacífico (Jacarta)	ap-southeast-3	3
Ásia-Pacífico (Melbourne)	ap-southeast-4	3
Ásia-Pacífico (Tailândia)	ap-southeast-7	3
Ásia-Pacífico (Tóquio)	ap-northeast-1	3
Canadá (Central)	ca-central-1	3
Oeste do Canadá (Calgary)	ca-west-1	3
Região China (Pequim)	cn-north-1	3

Nome da região	Região	Zonas de disponibilidade (computação)
China (Ningxia)	`cn-northwest-1`	3
Europa (Frankfurt)	`eu-central-1`	3
Europa (Zurique)	`eu-central-2`	3
Europa (Irlanda)	`eu-west-1`	3
Europa (Londres)	`eu-west-2`	3
Europa (Milão)	`eu-south-1`	3
Europa (Paris)	`eu-west-3`	3
Europa (Espanha)	`eu-south-2`	3
Europa (Estocolmo)	`eu-north-1`	3
México (Centro)	`mx-central-1`	3
Oriente Médio (Emirados Árabes Unidos)	`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

Definição de preço do Amazon DocumentDB

Os clusters do Amazon DocumentDB são faturados com base nos seguintes componentes:

- Horas de instância (por hora): com base na classe da instância (por exemplo, `db.r5.xlarge`). A definição de preço está listada em uma base por hora, mas é calculada em segundos e mostra

o tempo no formato decimal. O uso do Amazon DocumentDB é faturado em incrementos de um segundo, com um mínimo de dez minutos. Para obter mais informações, consulte Gerenciar classes de instância.

- I/O solicitações (por 1 milhão de solicitações por mês) — Número total de I/O solicitações de armazenamento que você faz em um ciclo de cobrança.
- Armazenamento de backup (por GiB por mês): o armazenamento de backup é o armazenamento associado a backups automatizados de banco de dados e a qualquer DB snapshot ativo que você tenha feito. Aumentar seu período de retenção de backup ou fazer snapshots de bancos de dados adicionais aumenta o armazenamento de backup utilizado por seu banco de dados. O armazenamento de backup é medido em GB-months e por segundo não se aplica. Para obter mais informações, consulte Backup e restauração no Amazon DocumentDB.
- Transferência de dados (por GB) — Transferência de dados para dentro e para fora da sua instância de ou para a Internet ou outros Regiões da AWS.

Para obter informações detalhadas, consulte Preços do Amazon DocumentDB.

Teste gratuito

É possível testar o Amazon DocumentDB gratuitamente usando a avaliação gratuita de 1 mês. Para obter mais informações, consulte Teste gratuito no site de preços do Amazon DocumentDB ou consulte as Perguntas frequentes sobre o teste gratuito do Amazon DocumentDB.

Monitoramento

Existem várias maneiras de controlar o desempenho e a integridade de uma instância. Você pode usar o CloudWatch serviço gratuito da Amazon para monitorar o desempenho e a integridade de uma instância. É possível encontrar gráficos de performance no console do Amazon DocumentDB. É possível assinar eventos do Amazon DocumentDB para obter notificações quando ocorrerem alterações em uma instância, um snapshot, um grupo de parâmetros ou um grupo de segurança.

Para obter mais informações, consulte as informações a seguir:

- Monitorando o Amazon DocumentDB com CloudWatch
- Registro de chamadas de API do Amazon DocumentDB com AWS CloudTrail

Interfaces

Há várias maneiras de você interagir com o Amazon DocumentDB, incluindo o Console de gerenciamento da AWS e o. AWS CLI

Console de gerenciamento da AWS

Console de gerenciamento da AWS É uma interface de usuário simples baseada na web. É possível gerenciar os clusters e as instâncias no console sem necessidade de programação. Para acessar o console do Amazon DocumentDB, faça login Console de gerenciamento da AWS e abra o console do Amazon DocumentDB em. https://console.aws.amazon.com/docdb

AWS CLI

Você pode usar o AWS Command Line Interface (AWS CLI) para gerenciar seus clusters e instâncias do Amazon DocumentDB. Com o mínimo de configuração, você pode começar a usar todas as funcionalidades fornecidas pelo console do Amazon DocumentDB do seu programa de terminal favorito.

- Para instalar o AWS CLI, consulte Instalando a interface de linha de AWS comando.
- Para começar a usar o AWS CLI para Amazon DocumentDB, consulte Referência de interface de linha de AWS comando para Amazon DocumentDB.

Drivers do MongoDB

Para desenvolver e gravar aplicações em um cluster do Amazon DocumentDB, você também pode usar os drivers do MongoDB com o Amazon DocumentDB. Para obter mais informações, consulte a guia da shell do MongoDB em Conectar-se com o TLS habilitado ou Conectar-se com o TLS desabilitado.

Próximas etapas

Nas seções anteriores, você conheceu os componentes de infraestrutura básicos que o Amazon DocumentDB oferece. O que você deve fazer em seguida? Dependendo de suas circunstâncias, consulte um dos tópicos a seguir para começar:

- Comece a usar o Amazon DocumentDB criando um cluster e uma instância usando CloudFormation Início rápido do uso do Amazon DocumentDB CloudFormation.

- Comece a usar o Amazon DocumentDB criando um cluster e uma instância usando as instruções no nosso Guia de conceitos básicos.
- Comece a usar o Amazon DocumentDB criando um cluster elástico usando as instruções na Como iniciar com clusters elásticos do Amazon DocumentDB.
- Migre sua implementação do MongoDB para o Amazon DocumentDB usando as orientações em Migração e atualização do Amazon DocumentDB

Amazon DocumentDB: como funciona

O Amazon DocumentDB (com compatibilidade com o MongoDB) é um serviço de banco de dados totalmente gerenciado MongoDB-compatible . Com o Amazon DocumentDB, é possível executar o mesmo código de aplicação e usar os mesmos drivers e ferramentas que você usa com o MongoDB. O Amazon DocumentDB é compatível com o MongoDB 3.6, 4.0, 5.0 e 8.0.

Tópicos

- Versões do Amazon DocumentDB
- Endpoints do Amazon DocumentDB
- Suporte a TLS
- Armazenamento do Amazon DocumentDB
- Replicação do Amazon DocumentDB
- Confiabilidade do Amazon DocumentDB
- Opções de preferência de leitura
- Exclusões de TTL
- Recursos faturáveis

Versões do Amazon DocumentDB

> **Note**
>
> A numeração de versões secundárias se aplica ao Amazon DocumentDB 5.0 e versões posteriores. O Amazon DocumentDB 3.6 e 4.0 usam somente as versões principais com patches de mecanismo.

A partir do Amazon DocumentDB 5.0, a versão do mecanismo usa um esquema de numeração de quatro níveis: major.major.minor.patch.

- Versão principal — Identificada pelos dois primeiros números da versão (por exemplo, 5.0). A versão principal representa o nível de compatibilidade do MongoDB. As versões principais introduzem novos recursos, melhorias de desempenho e atualizações de compatibilidade do MongoDB. A atualização entre versões principais (por exemplo, de 4.0 para 5.0) requer uma atualização de versão principal (MVU). Para obter mais informações, consulte Atualização da versão principal implementada do Amazon DocumentDB no local.
- Versão secundária — Identificada pelo terceiro número da versão (por exemplo, 1 na versão 5.0.1). As versões secundárias incluem melhorias como novos recursos, correções de segurança e correções de erros na mesma versão principal. Você pode atualizar para uma nova versão secundária manualmente. Para obter mais informações, consulte Atualização da versão secundária do Amazon DocumentDB.
- Versão do patch — Os patches do mecanismo são aplicados em uma versão secundária e contêm correções críticas de segurança e correções de erros. A versão do patch é monitorada pela versão do patch do mecanismo (por exemplo,`3.0.17983`), que você pode verificar executando`db.runCommand({getEngineVersion: 1})`. Os patches são aplicados durante a janela de manutenção do cluster. Para obter mais informações, consulte Manutenção do Amazon DocumentDB.

Ao criar um novo cluster, você pode especificar qualquer versão de mecanismo disponível. Para obter informações sobre as versões disponíveis e as datas de suporte, consulteDatas de suporte da versão do mecanismo do Amazon DocumentDB.

Ao usar o Amazon DocumentDB, você começa criando um cluster. Um cluster consiste em zero ou mais instâncias de banco de dados e em um volume de cluster que gerencia os dados para essas instâncias. Um volume de cluster do Amazon DocumentDB é um volume de armazenamento de banco de dados virtual que abrange várias zonas de disponibilidade. Cada zona de disponibilidade tem uma cópia de dados do cluster.

Um cluster do Amazon DocumentDB consiste em dois componentes:

- Volume de cluster — Usa um serviço de armazenamento nativo de nuvem para replicar dados de seis maneiras em três zonas de disponibilidade, fornecendo armazenamento resiliente e disponível. Um cluster do Amazon DocumentDB tem exatamente um volume de cluster, que pode

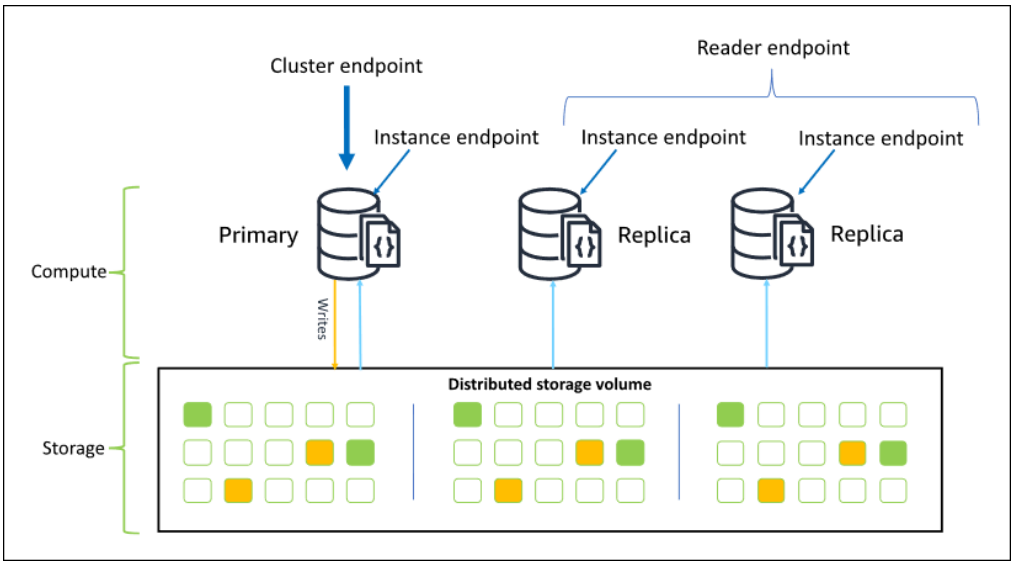
armazenar até 256 TiB de dados para a versão 8.0 e superior do motor (128 TiB para versões anteriores do mecanismo).

- Instâncias — Fornecem a potência do processamento para o banco de dados, gravando dados e lendo dados do volume de armazenamento do cluster. Um cluster do Amazon DocumentDB pode ter de 0–16 instâncias.

Instâncias atendem a uma das duas funções:

- Instância principal — Oferece suporte a operações de leitura e gravação e executa todas as modificações de dados no volume do cluster. Cada cluster do Amazon DocumentDB tem uma instância primária.
- Instância de réplica — Oferece suporte a operações somente leitura. Um cluster do Amazon DocumentDB pode ter até 15 réplicas, além da instância principal. Ter várias réplicas permite distribuir workloads de leitura. Além disso, ao colocar réplicas em zonas de disponibilidade separadas, você também aumenta a disponibilidade do cluster.

O diagrama a seguir ilustra a relação entre o volume do cluster, a instância principal e as réplicas em um cluster do Amazon DocumentDB:



As instâncias de cluster não precisam ser da mesma classe de instância e podem ser provisionadas e encerradas conforme desejado. Essa arquitetura permite escalar a capacidade computacional do cluster, independentemente do armazenamento.

Quando a aplicação grava dados na instância principal, ela executa uma gravação durável no volume do cluster. Em seguida, ele replica o estado dessa gravação (não os dados) em cada réplica

ativa. As réplicas do Amazon DocumentDB não participam do processamento de gravações e, portanto, as réplicas do Amazon DocumentDB são vantajosas para a escalabilidade de leitura. As leituras das réplicas do Amazon DocumentDB são eventualmente consistentes com o atraso mínimo da réplica, geralmente menos de 100 milissegundos após a instância principal gravar os dados. É garantido que as leituras das réplicas sejam lidas na ordem em que foram gravadas na instância principal. O atraso de réplica varia dependendo da taxa de alteração de dados, e períodos de alta atividade de gravação podem aumentar o atraso da réplica. Para obter mais informações, consulte as métricas `ReplicationLag` em Métricas do Amazon DocumentDB.

Endpoints do Amazon DocumentDB

O Amazon DocumentDB fornece várias opções de conexão para atender a uma ampla variedade de casos de uso. Para se conectar a uma instância em um cluster do Amazon DocumentDB, você especifica o endpoint da instância. Um endpoint é um endereço de host e um número de porta, separados por dois-pontos.

Conecte-se ao seu cluster usando o endpoint do cluster no modo de conjunto de réplicas (consulteConectar-se ao Amazon DocumentDB como um conjunto de réplicas). Use essa abordagem, a menos que você precise se conectar ao endpoint do leitor ou a um endpoint da instância. Para rotear solicitações para suas réplicas, escolha uma configuração de preferência de leitura do driver que maximize a escalabilidade de leitura, sem deixar de atender aos requisitos de consistência de leitura da aplicação. A preferência de leitura `secondaryPreferred` permite leituras de réplica e libera a instância primária para trabalhar mais.

Os endpoints a seguir estão disponíveis em um cluster do Amazon DocumentDB.

Endpoint do cluster

O endpoint de cluster conecta-se à instância principal atual do cluster. O endpoint do cluster pode ser usado para operações de leitura e gravação. Um cluster do Amazon DocumentDB tem exatamente um endpoint de cluster.

O endpoint de cluster dá suporte a failover para conexões de leitura e gravação para o cluster. Se a instância principal atual do cluster falhar e o cluster tiver pelo menos uma réplica de leitura ativa, o endpoint do cluster redirecionará automaticamente as solicitações de conexão para uma nova instância principal. Ao se conectar ao seu cluster Amazon DocumentDB, use o endpoint do cluster no modo de conjunto de réplicas (consulte). Conectar-se ao Amazon DocumentDB como um conjunto de réplicas

Veja a seguir um exemplo de endpoint do cluster do Amazon DocumentDB:

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

O exemplo a seguir é um exemplo de string de conexão utilizando esse endpoint de cluster:

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

Para obter informações sobre como localizar os endpoints de um cluster, consulte Localizar os endpoints de um cluster.

Endpoint de leitor

O endpoint do leitor balanceia a carga de conexões somente leitura em todas as réplicas disponíveis no cluster. Um endpoint de leitor de cluster funcionará como o endpoint do cluster se você estiver se conectando por meio do modo `replicaSet`, ou seja, na cadeia de conexão, o parâmetro do conjunto de réplicas é `&replicaSet=rs0`. Nesse caso, será possível realizar operações de gravação no primário. No entanto, se você se conectar ao cluster especificando `directConnection=true`, a tentativa de realizar uma operação de gravação em uma conexão com o endpoint do leitor ocasionará erro. Um cluster do Amazon DocumentDB tem exatamente um endpoint de leitor.

Se o cluster contiver apenas uma instância (principal), o endpoint do leitor se conectará à instância principal. Quando você adicionar uma instância de réplica ao cluster do Amazon DocumentDB, o endpoint do leitor abrirá as conexões somente leitura para a nova réplica depois que ela estiver ativa.

Veja a seguir um exemplo de endpoint de leitor para um cluster do Amazon DocumentDB:

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

O exemplo a seguir é um exemplo de string de conexão utilizando um endpoint de leitor:

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

O endpoint do leitor balanceia a carga de conexões somente leitura, e não solicitações de leitura. Se algumas conexões do endpoint de leitor forem mais utilizadas do que outras, suas solicitações de leitura poderão não ser igualmente equilibradas entre as instâncias do cluster. É recomendável

distribuir solicitações conectando-se ao endpoint do cluster como um conjunto de réplicas e utilizando a opção de preferência de leitura secondaryPreferred.

Para obter informações sobre como localizar os endpoints de um cluster, consulte Localizar os endpoints de um cluster.

Endpoint da instância

Um endpoint da instância se conecta a uma instância específica no cluster. O endpoint da instância para a instância principal atual pode ser usado para operações de leitura e gravação. No entanto, a tentativa de executar operações de gravação em um endpoint da instância para uma réplica de leitura resulta em um erro. Um cluster do Amazon DocumentDB tem um endpoint de instância por instância ativa.

Um endpoint de instância oferece controle direto sobre conexões para uma instância específica, para cenários nos quais o endpoint de cluster ou o endpoint de leitor talvez não seja apropriado. Um exemplo de caso de uso é o provisionamento de uma workload de analytics periódica somente leitura. É possível provisionar uma instância de réplica maior que o normal, conectar-se diretamente à nova instância maior com o endpoint da instância, executar as consultas de analytics e encerrar a instância. Usar o endpoint da instância impede que o tráfego de analytics cause impacto em outras instâncias do cluster.

Este é um exemplo de endpoint de instância para uma única instância em um cluster do Amazon DocumentDB:

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

O exemplo a seguir é um exemplo de string de conexão utilizando esse endpoint da instância:

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

> **Note**
>
> A função de uma instância como principal ou de réplica pode mudar devido a um evento de failover. As aplicações nunca devem presumir que um endpoint de instância específico seja a instância principal. Não se conecte aos endpoints da instância para aplicativos de produção. Em vez disso, use o endpoint do cluster no modo de conjunto de réplicas (consulteConectar-se ao Amazon DocumentDB como um conjunto de réplicas). Para obter mais controle

avançado da prioridade de failover da instância, consulte Entender a tolerância a falhas de cluster do Amazon DocumentDB.

Para obter informações sobre como localizar os endpoints de um cluster, consulte Localizar o endpoint de uma instância.

Modo de conjuntos de réplicas

É possível se conectar ao endpoint de cluster do Amazon DocumentDB no modo de conjunto de réplicas especificando o nome do conjunto de réplicas `rs0`. A conexão no modo de conjunto de réplicas fornece a capacidade de especificar as opções Read Concern, Write Concern e Read Preference. Para obter mais informações, consulte Consistência de leituras.

O exemplo a seguir é de uma string de conexão conectando-se no modo de conjunto de réplicas:

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

Quando você se conecta no modo de conjunto de réplicas, o cluster do Amazon DocumentDB aparece para os seus drivers e clientes como um conjunto de réplicas. As instâncias adicionadas e removidas do cluster do Amazon DocumentDB são refletidas automaticamente na configuração do conjunto de réplicas.

Cada cluster do Amazon DocumentDB consiste em um único conjunto de réplicas com o nome padrão `rs0`. O nome do conjunto de réplicas não pode ser modificado.

A conexão ao endpoint do cluster no modo de conjunto de réplicas é o método recomendado para uso geral.

> **Note**
>
> Todas as instâncias em um cluster do Amazon DocumentDB atendem a mesma porta TCP para conexões.

Suporte a TLS

Para obter mais detalhes sobre a conexão ao Amazon DocumentDB usando o Transport Layer Security (TLS), consulte Criptografia de dados em trânsito.

Armazenamento do Amazon DocumentDB

Os dados do Amazon DocumentDB são armazenados em um volume de cluster, que é um volume virtual único que usa unidades de estado sólido (SSDs). Um volume de cluster consiste em seis cópias dos dados, que são replicados automaticamente em diversas zonas de disponibilidade em uma única região da Região da AWS. Essa replicação ajuda a garantir que seus dados sejam resilientes, com menor possibilidade de perda de dados. Isso também ajuda a garantir que o cluster esteja mais disponível durante um failover, pois as cópias dos dados já existem em outras zonas de disponibilidade. Essas cópias podem continuar a atender às solicitações de dados para as instâncias no cluster do Amazon DocumentDB.

Como o armazenamento de dados do é faturado

O Amazon DocumentDB aumenta automaticamente o tamanho de um volume de cluster à medida que a quantidade de dados aumenta. Um volume de cluster do Amazon DocumentDB pode crescer até um tamanho máximo de 256 TiB para a versão 8.0 e superior do motor (128 TiB para versões anteriores do mecanismo). No entanto, você só é cobrado pelo espaço usado em um volume de cluster do Amazon DocumentDB. A partir do Amazon DocumentDB 4.0, quando os dados são removidos, como ao excluir uma coleção ou índice, o espaço total alocado diminui em uma quantidade equivalente. Assim, é possível reduzir as cobranças de armazenamento excluindo coleções, índices e bancos de dados que não são mais necessários. Na versão 3.6 do Amazon DocumentDB, o volume do cluster pode reutilizar o espaço que for liberado quando você remover dados, mas o volume em si nunca diminui. Como resultado, na versão 3.6, é possível não testemunhar alteração nenhuma no armazenamento ao descartar uma coleção ou índice, mesmo que o espaço liberado seja reutilizado.

> **Note**
>
> Com o Amazon DocumentDB 3.6, os custos de armazenamento são baseados no “limite máximo” de armazenamento (o valor máximo que foi alocado para o cluster Amazon DocumentDB a qualquer momento). É possível gerenciar custos evitando práticas de ETL que criam grandes volumes de informações temporárias ou que carregam grandes volumes de novos dados antes de remover dados antigos desnecessários. Se a remoção de dados de um cluster do Amazon DocumentDB resultar em uma quantidade substancial de espaço alocado mas não utilizado, a redefinição do nível mais alto da marca d'água vai exigir o despejo de dados lógicos e a restauração de um novo cluster usando uma ferramenta como `mongodump` ou `mongorestore`. A criação e restauração de um snapshot não reduz o

armazenamento alocado, pois o layout físico do armazenamento subjacente permanece o mesmo no snapshot restaurado.

> **Note**
>
> O uso de utilitários como `mongodump` e `mongorestore` incorre em I/O cobranças com base nos tamanhos dos dados que estão sendo lidos e gravados no volume de armazenamento.

Para obter informações sobre armazenamento de dados e I/O preços do Amazon DocumentDB, consulte Perguntas frequentes sobre preços e preços do Amazon DocumentDB (com compatibilidade com MongoDB). https://aws.amazon.com/documentdb/faqs/#Pricing

Replicação do Amazon DocumentDB

Em um cluster do Amazon DocumentDB, cada instância de réplica expõe um endpoint independente. Esses endpoints de réplica fornecem acesso somente leitura aos dados no volume do cluster. Eles permitem escalar a workload de leitura para os dados em várias instâncias replicadas. Eles também ajudam a melhorar o desempenho das leituras de dados e a aumentar a disponibilidade dos dados em seu cluster do Amazon DocumentDB. As réplicas do Amazon DocumentDB também são alvos de failover e são promovidas rapidamente se a instância primária do seu cluster do Amazon DocumentDB falhar.

Confiabilidade do Amazon DocumentDB

O Amazon DocumentDB foi projetado para ser confiável, durável e tolerante a falhas. (Para melhorar a disponibilidade, você deve configurar seu cluster Amazon DocumentDB para que ele tenha várias instâncias de réplica em diferentes zonas de disponibilidade.) O Amazon DocumentDB inclui vários recursos automáticos que o tornam uma solução de banco de dados confiável.

Reparo automático de armazenamento

O Amazon DocumentDB mantém várias cópias dos dados em três zonas de disponibilidade, reduzindo bastante a chance de perda de dados devido a uma falha de armazenamento. O Amazon DocumentDB detecta automaticamente as falhas no volume do cluster. Quando um segmento de um volume de cluster falha, o Amazon DocumentDB repara imediatamente o segmento. Ele usa os dados dos outros volumes que compõem o volume do cluster para ajudar a garantir que os dados no segmento reparado sejam atuais. Como resultado, o Amazon DocumentDB evita a perda de dados

e reduz a necessidade de executar uma restauração pontual para recuperação de uma falha de instância.

Aquecimento de cache possível de recuperar

O Amazon DocumentDB gerencia seu cache de páginas em um processo separado do banco de dados, de modo que o cache de páginas possa sobreviver independentemente do banco de dados. No evento improvável de uma falha no banco de dados, o cache da página permanece na memória. Isso garante que o grupo de buffers seja aquecido com o estado mais atual quando o banco de dados é reiniciado.

Recuperação de falha

O Amazon DocumentDB foi projetado para se recuperar de uma falha quase instantaneamente e continuar fornecendo seus dados de aplicações. O Amazon DocumentDB executa a recuperação de falhas de forma assíncrona em threads paralelos, de maneira que o banco de dados seja aberto e fique disponível imediatamente após a falha.

Governança de recursos

O Amazon DocumentDB protege recursos para processos críticos, como verificações de saúde. Quando uma instância sofre alta pressão de memória, o Amazon DocumentDB limita as solicitações. Algumas operações podem ficar em fila até que a pressão da memória diminua. Se a pressão da memória continuar, as operações em fila podem atingir o tempo limite.

Monitore a limitação de pouca memória com as seguintes CloudWatch métricas:`LowMemThrottleQueueDepth`,,,`LowMemThrottleMaxQueueDepth`. `LowMemNumOperationsThrottled` `LowMemNumOperationsTimedOut` Para obter mais informações, consulte Monitorando o Amazon DocumentDB com CloudWatch.

Se você observar uma pressão constante na memória, amplie sua instância para fornecer memória adicional para sua carga de trabalho.

Opções de preferência de leitura

O Amazon DocumentDB usa um serviço de armazenamento compartilhado nativo de nuvem que replica os dados seis vezes em três zonas de disponibilidade para fornecer altos níveis de durabilidade. O Amazon DocumentDB não depende da replicação de dados em várias instâncias para obter durabilidade. Os dados do cluster são duráveis, quer contenham uma única instância ou 15 instâncias.

Tópicos

- Durabilidade de gravação
- Isolamento de leitura
- Consistência de leituras
- Alta disponibilidade
- Leituras de escalabilidade

Durabilidade de gravação

O Amazon DocumentDB usa um sistema de armzenamento exclusivo, distribuído, tolerante a falhas e de recuperação automática. Esse sistema replica seis cópias (V=6) de seus dados em três zonas de AWS disponibilidade para oferecer alta disponibilidade e durabilidade. Ao gravar dados, o Amazon DocumentDB garante que todas as gravações sejam gravadas de forma durável na maioria dos nós antes de confirmar a gravação para o cliente. Se você estiver executando um conjunto de réplicas do MongoDB de três nós, o uso de uma Write Concern de `{w:3, j:true}` produzirá a melhor configuração possível em comparação com o Amazon DocumentDB.

As gravações em um cluster do Amazon DocumentDB devem ser processadas pela instância principal do cluster. A tentativa de gravar em um leitor resulta em um erro. Uma gravação reconhecida de uma instância principal do Amazon DocumentDB é durável e não pode ser revertida. O Amazon DocumentDB é altamente durável por padrão e não oferece suporte a uma opção de gravação não durável. Você não pode modificar o nível de durabilidade (ou seja, preocupação de gravação). O Amazon DocumentDB ignora w=anything e é efetivamente w: 3 e j: true. Você não pode reduzi-lo.

Devido à separação de armazenamento e computação na arquitetura do Amazon DocumentDB, um cluster com uma única instância é resiliente. A durabilidade é processada na camada de armazenamento. Como resultado, um cluster do Amazon DocumentDB com uma única instância e um com três instâncias alcança o mesmo nível de durabilidade. É possível configurar o cluster para seu caso de uso específico e, ao mesmo tempo, proporcionar resiliência aos seus dados.

As gravações em um cluster do Amazon DocumentDB são atômicas em um único documento.

O Amazon DocumentDB não oferece suporte à opção `wtimeout` e não retornará um erro se um valor for especificado. É garantido que as gravações na instância principal do Amazon DocumentDB não sejam bloqueadas indefinidamente.

Isolamento de leitura

As leituras de uma instância do Amazon DocumentDB retornam apenas dados que sejam duráveis antes do início da consulta. As leituras nunca retornam dados modificados depois que a consulta começa a execução, nem as leituras contaminadas são possíveis sob qualquer circunstância.

Consistência de leituras

Os dados lidos em um cluster do Amazon DocumentDB são duráveis e não serão revertidos. É possível modificar a consistência de leitura para as leituras do Amazon DocumentDB especificando a preferência de leitura para a solicitação ou conexão. O Amazon DocumentDB não oferece suporte a uma opção de leitura não durável.

As leituras a partir de uma instância principal do cluster do Amazon DocumentDB são altamente consistentes em condições operacionais normais e consistência de leitura após gravação. Se ocorrer um evento de failover entre a leitura e a gravação subsequentes, o sistema poderá retornar em breve uma leitura que não seja altamente consistente. Todas as leituras a partir de uma réplica de leitura são, por fim, consistentes e retornam os dados na mesma ordem e, geralmente, com atraso de replicação inferior a 100 ms.

Preferências de leitura do Amazon DocumentDB

O Amazon DocumentDB oferece suporte à configuração de uma opção de preferência de leitura apenas ao ler dados do endpoint do cluster no modo de conjunto de réplicas. Definir uma opção de preferência de leitura afeta como o cliente ou o driver do MongoDB encaminha solicitações de leitura para instâncias no cluster do Amazon DocumentDB. É possível definir opções de preferência de leitura para uma consulta específica ou como uma opção geral no driver do MongoDB. (Consulte a documentação do cliente ou do driver para obter instruções sobre como definir uma opção de preferência de leitura.)

Se o cliente ou o driver não estiver se conectando a um endpoint de cluster do Amazon DocumentDB no modo de conjunto de réplicas, o resultado da especificação de uma preferência de leitura será indefinido.

O Amazon DocumentDB não é compatível com a configuração de conjuntos de tags como uma preferência de leitura.

Opções de preferência de leitura suportadas

- **`primary`**— A especificação de uma preferência de leitura `primary` ajuda a garantir que todas as leituras sejam encaminhadas para a instância principal do cluster. Se a instância principal

estiver indisponível, a operação de leitura falhará. Uma preferência de leitura `primary` produz a consistência de leitura após gravação e é apropriada para casos de uso que priorizam a consistência de leitura após gravação em alta disponibilidade e escalabilidade de leitura.

O exemplo a seguir especifica uma preferência de leitura `primary`:

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

- **`primaryPreferred`**— A especificação de rotas de preferência de leitura `primaryPreferred` lê para a instância principal em operação normal. Se houver um failover principal, o cliente encaminhará solicitações para uma réplica. Uma preferência de leitura `primaryPreferred` produz consistência de leitura após gravação durante a operação normal e leituras eventualmente consistentes durante um evento de failover. Uma preferência de leitura `primaryPreferred` é apropriada para casos de uso que priorizam a consistência de leitura após gravação em escalabilidade de leitura, mas ainda exigem alta disponibilidade.

  O exemplo a seguir especifica uma preferência de leitura `primaryPreferred`:

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

- **`secondary`**— A especificação de uma preferência de leitura `secondary` garante que as leituras sejam encaminhadas apenas para uma réplica, nunca para a instância principal. Se não houver instâncias de réplica em um cluster, a solicitação de leitura falhará. Uma preferência de leitura `secondary` produz leituras eventualmente consistentes e é apropriada para casos de uso que priorizam a throughput de gravação da instância principal em alta disponibilidade e consistência de leitura após gravação.

  O exemplo a seguir especifica uma preferência de leitura `secondary`:

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

- **`secondaryPreferred`**— A especificação de uma preferência de leitura `secondaryPreferred` garante que as leituras sejam encaminhadas para uma réplica de leitura quando uma ou mais réplicas estiverem ativas. Se não houver instâncias de réplica ativas em um cluster, a

solicitação de leitura será encaminhada para a instância principal. Uma preferência de leitura `secondaryPreferred` produz leituras eventualmente consistentes quando a leitura é atendida por uma réplica de leitura. Ela produz consistência de leitura após gravação quando a leitura é atendida pela instância principal (exceto eventos de failover). Uma preferência de leitura `secondaryPreferred` é apropriada para casos de uso que priorizam a escalabilidade de leitura e a alta disponibilidade sobre a consistência de leitura após gravação.

O exemplo a seguir especifica uma preferência de leitura `secondaryPreferred`:

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

- **`nearest`**— A especificação de uma preferência de leitura `nearest` encaminha as leituras baseadas apenas na latência medida entre o cliente e todas as instâncias no cluster do Amazon DocumentDB. Uma preferência de leitura `nearest` produz leituras eventualmente consistentes quando a leitura é atendida por uma réplica de leitura. Ela produz consistência de leitura após gravação quando a leitura é atendida pela instância principal (exceto eventos de failover). Uma preferência de leitura `nearest` é apropriada para casos de uso que priorizam alcançar a menor latência de leitura e alta disponibilidade sobre a consistência de leitura após gravação e escalabilidade de leitura.

  O exemplo a seguir especifica uma preferência de leitura `nearest`:

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

Alta disponibilidade

O Amazon DocumentDB oferece suporte a configurações de cluster altamente disponíveis usando réplicas como destinos de failover para a instância principal. Se a instância principal falhar, uma réplica do Amazon DocumentDB será promovida como a nova principal, com uma breve interrupção durante a qual as solicitações de leitura e gravação feitas na instância principal falham com uma exceção.

Se o cluster do Amazon DocumentDB não incluir réplicas, a instância principal será recriada durante uma falha. Como promover uma réplica do Amazon DocumentDB é muito mais rápido do que recriar a instância primária, crie uma ou mais réplicas do Amazon DocumentDB como destinos de failover.

As réplicas que devem ser usadas como destinos de failover devem ser da mesma classe de instância da instância principal. Elas devem ser provisionadas em zonas de disponibilidade diferentes da principal. É possível controlar quais réplicas são preferenciais como destinos de failover. Para obter as melhores práticas sobre como configurar o Amazon DocumentDB para alta disponibilidade, consulte Entender a tolerância a falhas de cluster do Amazon DocumentDB.

Leituras de escalabilidade

As réplicas do Amazon DocumentDB são ideais para escalabilidade de leitura. Elas são totalmente dedicadas a operações de leitura no volume de cluster, ou seja, as réplicas não processam gravações. A replicação de dados acontece dentro do volume de cluster e não entre as instâncias. Portanto, os recursos de cada réplica são dedicados ao processamento de consultas, e não às replicações e gravações de dados.

Se a aplicação precisar de mais capacidade de leitura, será possível adicionar uma réplica ao cluster rapidamente (geralmente em menos de dez minutos). Se os requisitos de capacidade de leitura diminuírem, será possível remover as réplicas desnecessárias. Com as réplicas do Amazon DocumentDB, você paga apenas pela capacidade de leitura de que precisa.

O Amazon DocumentDB oferece suporte a escalabilidade de leitura do lado do cliente por meio do uso de opções de preferência de leitura. Para obter mais informações, consulte Preferências de leitura do Amazon DocumentDB.

Exclusões de TTL

As exclusões de uma área de índice TTL alcançada por meio de um processo em segundo plano são o melhor esforço e não são garantidas dentro de um período de tempo específico. Fatores como tamanho de instância, utilização de recursos da instância, tamanho do documento e throughput geral podem afetar a sincronização de uma exclusão de TTL.

Quando o monitor TTL exclui seus documentos, cada exclusão resulta em custos de E/S, o que aumentará sua fatura. Se as taxas de throughput e de exclusão de TTL aumentarem, espere um aumento em sua fatura devido ao aumento no uso de E/S.

Ao criar um índice TTL em uma coleção existente, você deve excluir todos os documentos expirados antes de criar o índice. A implementação atual do TTL é otimizada para excluir uma pequena fração de documentos na coleção, o que é típico se o TTL foi ativado na coleção desde o início, e pode resultar em IOPS maior do que o necessário se um grande número de documentos precisar ser excluído de uma só vez.

Caso você não queira criar um índice TTL para excluir documentos, é possível segmentar documentos em coleções com base no tempo e simplesmente descartar essas coleções quando os documentos não forem mais necessários. Por exemplo: você pode criar uma coleção por semana e descartá-la sem incorrer em custos de E/S. Isso pode ser significativamente mais econômico do que usar um índice TTL.

Recursos faturáveis

Identificação de recursos faturáveis do Amazon DocumentDB

Como um serviço de banco de dados totalmente gerenciado, o Amazon DocumentDB cobra por instâncias, armazenamento I/Os, backups e transferência de dados. Para obter mais informações, consulte Preços do Amazon DocumentDB (compatível com MongoDB).

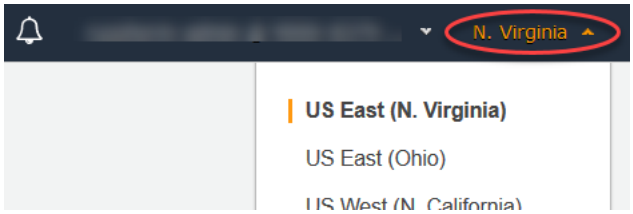
Para descobrir recursos faturáveis em sua conta e potencialmente excluir os recursos, você pode usar o Console de gerenciamento da AWS ou. AWS CLI

Usar o Console de gerenciamento da AWS

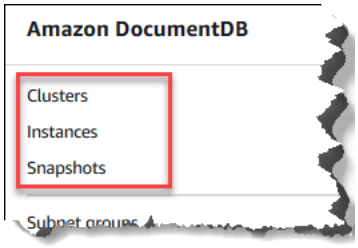
Usando o Console de gerenciamento da AWS, você pode descobrir os clusters, instâncias e snapshots do Amazon DocumentDB que você provisionou para um determinado. Região da AWS

Para descobrir clusters, instâncias e snapshots

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. Para descobrir recursos faturáveis em uma região diferente da sua região padrão, no canto superior direito da tela, escolha o Região da AWS que você deseja pesquisar.



3. No painel de navegação, escolha o tipo de recurso faturável de interesse em: Clusters, Instances (Instâncias) ou Snapshots.



4. Todos os seus clusters provisionados, instâncias ou snapshots para a região são listados no painel direito. Você será cobrado por clusters, instâncias e snapshots.

Usar o AWS CLI

Usando o AWS CLI, você pode descobrir os clusters, instâncias e snapshots do Amazon DocumentDB que você provisionou para um determinado. Região da AWS

Para descobrir clusters e instâncias

O código a seguir lista todos os clusters e instâncias para a região especificada. Se você deseja procurar clusters e instâncias em sua região padrão, omita o parâmetro `--region`.

Example

Para Linux, macOS ou 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"
```

A saída dessa operação é semelhante à seguinte.

```
"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 descobrir snapshots

O código a seguir lista todos os snapshots para a região especificada. Se você deseja procurar snapshots em sua região padrão, omita o parâmetro `--region`.

Para Linux, macOS ou 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]'
```

A saída dessa operação é semelhante à seguinte.

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

Você só precisa excluir `manual` snapshots. Os snapshots `Automated` são excluídos quando você exclui o cluster.

Exclusão de recursos faturáveis indesejados

Para excluir um cluster, primeiro exclua todas as instâncias no cluster.

- Para excluir instâncias, consulte Excluir uma instância do Amazon DocumentDB.

> **Important**
>
> Mesmo se você excluir as instâncias em um cluster, você ainda será cobrado pelo uso de armazenamento e backup associado a esse cluster. Para interromper todas as cobranças, você também deverá excluir seu cluster e snapshots manuais.

- Para excluir clusters, consulte Excluir um cluster do Amazon DocumentDB.
- Para excluir snapshots manuais, consulte Exclusão de um snapshot de cluster.

O que é um banco de dados de documentos?

Alguns desenvolvedores não pensam em seu modelo de dados em termos de linhas e colunas normalizadas. Normalmente, na camada da aplicação, os dados são representados como um documento JSON, pois é mais intuitivo para os desenvolvedores pensarem em seu modelo de dados como um documento.

A popularidade dos bancos de dados de documentos cresceu porque eles permitem que você mantenha a persistência dos dados em um banco de dados usando o mesmo formato de modelo de documento usado no código da aplicação. Os bancos de dados de documentos fornecem APIs poderosas e intuitivas para desenvolvimento flexível e ágil.

Tópicos

- Casos de uso de um banco de dados
- Noções básicas sobre documentos
- Trabalhar com documentos

Casos de uso de um banco de dados

Seu caso de uso indica se você precisa de um banco de dados de documentos ou algum outro tipo de banco de dados para gerenciar os dados. Os bancos de dados de documentos são úteis para workloads que exigem um esquema flexível para desenvolvimento rápido e iterativo. A seguir estão alguns exemplos de casos de uso para os quais os bancos de dados de documentos podem fornecer vantagens significativas:

Tópicos

- Perfis de usuário

- Real-time big data
- Gerenciamento de conteúdo

Perfis de usuário

Como os bancos de dados de documentos têm um esquema flexível, eles podem armazenar documentos com atributos e valores de dados diferentes. Os bancos de dados de documentos são uma solução prática para perfis online nos quais usuários diferentes fornecem tipos de informações diferentes. Usando um banco de dados de documentos, você pode armazenar o perfil de cada usuário de forma eficiente, armazenando apenas os atributos específicos de cada usuário.

Suponha que um usuário opte por adicionar ou remover informações do perfil. Nesse caso, o documento pode ser facilmente substituído por uma versão atualizada que contenha quaisquer atributos e dados adicionados recentemente ou que omita quaisquer atributos e dados omitidos recentemente. Os bancos de dados de documentos gerenciam facilmente esse nível de individualidade e fluidez.

Real-time big data

Historicamente, a capacidade de extrair informações de dados operacionais era prejudicada pelo fato de bancos de dados operacionais e bancos de dados analíticos serem mantidos em ambientes diferentes — operacionais e, respectivamente. business/reporting A capacidade de extrair informações operacionais em tempo real é fundamental em um ambiente de negócios altamente competitivo. Ao usar bancos de dados de documentos, uma empresa pode armazenar e gerenciar dados operacionais de qualquer origem e, simultaneamente, alimentar os dados para o mecanismo de BI escolhido para análise. Não há necessidade de ter dois ambientes.

Gerenciamento de conteúdo

Para gerenciar o conteúdo com eficiência, é necessário coletar e agregar o conteúdo de várias fontes e, em seguida, enviá-lo ao cliente. Devido ao esquema flexível, os bancos de dados de documentos são perfeitos para coletar e armazenar qualquer tipo de dados. É possível usá-los para criar e incorporar novos tipos de conteúdo, incluindo conteúdo gerado pelo usuário, como imagens, comentários e vídeos.

Noções básicas sobre documentos

Os bancos de dados de documentos são usados para armazenar dados semiestruturados como um documento em vez de normalizar dados em várias tabelas, cada uma com uma estrutura única e

fixa, como em um banco de dados relacional. Os documentos armazenados em um banco de dados de documentos usam pares de chave/valor aninhados para fornecer a estrutura ou o esquema do documento. No entanto, diferentes tipos de documentos podem ser armazenados no mesmo banco de dados de documentos, atendendo assim ao requisito de processamento de dados semelhantes em formatos diferentes. Por exemplo, como cada documento é autodescritivo, os JSON-encoded documentos de uma loja on-line descritos no tópico Exemplo de documentos em um banco de dados de documentos podem ser armazenados no mesmo banco de dados de documentos.

Tópicos

- Terminologia do SQL versus não relacional
- Documentos simples
- Documentos incorporados
- Exemplo de documentos em um banco de dados de documentos
- Noções básicas sobre normalização em um banco de dados de documentos

Terminologia do SQL versus não relacional

A tabela a seguir compara a terminologia usada pelos bancos de dados de documentos (MongoDB) com a terminologia usada pelos bancos de dados SQL.

SQL	MongoDB
Tabela	Coleta
Linha	Documento
Coluna	Campo
Chave primária	ObjectId
Índice	Índice
Visualizar	Visualizar
Tabela ou objeto aninhado	Documento incorporado
Array	Array

Documentos simples

Todos os documentos em um banco de dados de documentos são autodescritivos. Essa documentação usa documentos JSON-like formatados, embora você possa usar outros meios de codificação.

Um documento simples tem um ou mais campos que estão todos no mesmo nível no documento. No exemplo a seguir, os campos `SSN`, `LName`, `FName`, `DOB`, `Street`, `City`, `State-Province`, `PostalCode` e `Country` são todos irmãos no documento.

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

Quando as informações são organizadas em um documento simples, cada campo é gerenciado individualmente. Para recuperar o endereço de uma pessoa, é necessário recuperar `Street`, `City`, `State-Province`, `PostalCode` e `Country` como itens de dados individuais.

Documentos incorporados

Um documento complexo organiza os dados criando documentos incorporados no documento. Documentos incorporados ajudam a gerenciar dados em agrupamentos e como itens de dados individuais, o que for mais eficiente em um determinado caso. Usando o exemplo anterior, você poderia incorporar um documento `Address` no documento principal. Isso resulta na estrutura do documento a seguir:

```
{
   "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"
    }
}
```

Agora, você pode acessar os dados no documento como campos individuais (`"SSN":`), como um documento incorporado (`"Address":`) ou como membro de um documento incorporado (`"Address":{"Street":}`).

Exemplo de documentos em um banco de dados de documentos

Como afirmado anteriormente, como cada documento em um banco de dados de documentos é autodescritivo, a estrutura dos documentos em um banco de dados de documentos pode ser diferente. Os dois documentos seguintes, um de um livro e o outro de um periódico, são estruturalmente diferentes. No entanto, ambos podem estar no mesmo banco de dados de documentos.

Veja a seguir um exemplo de documento de livro:

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

Veja a seguir um exemplo de documento de periódico com dois artigos:

```
{
    "_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 a estrutura desses dois documentos. Com um banco de dados relacional, você precisa de tabelas de "periódico" e de "livros" separadas ou de uma única tabela com campos não utilizados, como "Publicação", "Problema", "Artigos" e "MI", como valores `null`. Como os bancos de dados de documentos são semiestruturados, com cada documento definindo a própria estrutura, esses dois documentos podem coexistir no mesmo banco de dados de documentos sem campos `null`. Bancos de dados de documentos são ideais para lidar com dados esparsos.

O desenvolvimento em um banco de dados de documentos permite um desenvolvimento rápido e interativo. Isso ocorre porque você pode alterar a estrutura de dados de um documento dinamicamente, sem precisar alterar o esquema para toda a coleção. Os bancos de dados de documentos são adequados para o desenvolvimento ágil e para ambientes que mudam dinamicamente.

Noções básicas sobre normalização em um banco de dados de documentos

Bancos de dados de documentos não são normalizados; os dados encontrados em um documento podem ser repetidos em outro documento. Além disso, algumas discrepâncias de dados podem existir entre documentos. Por exemplo, considere o cenário em que você faz uma compra em um

armazenamento on-line e todos os detalhes de suas compras são armazenados em um único documento. O documento pode ser semelhante ao documento JSON a seguir:

```
{
    "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"
}
```

Todas essas informações são armazenadas como um documento em uma coleção de transações. Mais tarde, você percebe que esqueceu de comprar um item. Então, você faz logon novamente na mesma loja e faz outra compra, que também é armazenada como outro documento na coleção de transações.

```
{
```

```
    "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 a redundância entre esses dois documentos seu nome e ID de comprador (e, se você usou o mesmo cartão de crédito, suas informações de cartão de crédito). Mas não tem problema, porque o armazenamento é barato e cada documento registra completamente uma única transação que pode ser recuperada rapidamente com uma consulta simples de chave/valor que não requer junções.

Existe também uma aparente discrepância entre os dois documentos as informações do seu cartão de crédito. Essa é apenas uma discrepância aparente porque é provável que você tenha usado um cartão de crédito diferente para cada compra. Cada documento é preciso em relação à transação que ele documenta.

Trabalhar com documentos

Como um banco de dados de documentos, o Amazon DocumentDB facilita o armazenamento, a consulta e a indexação de dados JSON. No Amazon DocumentDB, uma coleção de banco de dados de documentos é semelhante a uma tabela em um banco de dados relacional, exceto pelo fato de não haver um único esquema aplicado a todos os documentos. As coleções permitem agrupar documentos semelhantes, mantendo-os todos no mesmo banco de dados, sem exigir que eles tenham a estrutura idêntica.

Usando os exemplos de documentos das seções anteriores, é provável que você tenha coleções de `reading_material` e `office_supplies`. É função do software impor a que coleção um documento pertence.

Os exemplos a seguir usam a API do MongoDB para mostrar como adicionar, consultar, atualizar e excluir documentos.

Tópicos

- Adicionar documentos
- Consultar documentos
- Como atualizar documentos
- Como excluir documentos

Adicionar documentos

No Amazon DocumentDB, um banco de dados é criado quando você adiciona um documento a uma coleção. Neste exemplo, você está criando uma coleção chamada `example` no banco de dados `test`, que é o banco de dados padrão quando ao conectar-se a um cluster. Como a coleção é criada implicitamente quando o primeiro documento é inserido, não há erro ao verificar o nome da coleção. Portanto, um erro de digitação no nome da coleção, como `eexample` em vez de `example`, criará e adicionará o documento à coleção `eexample` em vez de à coleção pretendida. A verificação de erros deve ser processada pela sua aplicação.

Os exemplos a seguir usam a API do MongoDB para adicionar documentos.

Tópicos

- Adicionar um único documento
- Adicionar diversos documentos

Adicionar um único documento

Para adicionar um único documento a uma coleção, use a operação `insertOne( {} )` com o documento que você deseja adicionar à coleção.

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

```
        },
        "UnitPrice": 0.89
    }
)
```

A saída dessa operação é semelhante ao seguinte (formato JSON).

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

Adicionar diversos documentos

Para adicionar diversos documentos a uma coleção, use a operação `insertMany( [{},..., {}] )` com uma lista dos documentos que você deseja adicionar à coleção. Embora os documentos nessa lista específica tenham esquemas diferentes, todos eles podem ser adicionados à mesma coleção.

```
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
            }
        }
    ]
)
```

A saída dessa operação é semelhante ao seguinte (formato JSON).

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

Consultar documentos

Às vezes, pode ser necessário examinar o inventário da sua loja online para que os clientes possam visualizar e comprar o que você está vendendo. Consultar uma coleção é relativamente fácil, quer você queira todos os documentos na coleção ou apenas os documentos que satisfazem a um critério específico.

Para consultar documentos, use a operação `find()`. O comando `find()` tem um único parâmetro do documento que define os critérios a serem usados na escolha dos documentos a serem retornados. A saída de `find()` é um documento formatado como uma única linha de texto sem quebras de linha. Para formatar o documento de saída para facilitar a leitura, use `find().pretty()`. Todos os exemplos deste tópico usam `.pretty()` para formatar a saída.

Use os quatro documentos inseridos na coleção `example` nos últimos dois exercícios - `insertOne()` e `insertMany()`.

Tópicos

- Recuperar todos os documentos em uma coleção
- Recuperar documentos que correspondem a um valor de campo
- Recuperar documentos que correspondem a um documento incorporado
- Recuperar documentos que correspondem a um valor de campo em um documento incorporado

- Recuperar documentos que correspondem a uma matriz
- Recuperar documentos que correspondem a um valor em uma matriz
- Recuperar documentos usando operadores

Recuperar todos os documentos em uma coleção

Para recuperar todos os documentos em sua coleção, use a operação `find()` com um documento de consulta vazio.

A consulta a seguir retorna todos os documentos da coleção `example`.

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

Recuperar documentos que correspondem a um valor de campo

Para recuperar todos os documentos que correspondem a um campo e valor, use a operação `find()` com um documento de consulta que identifica os campos e valores a serem correspondidos.

Usando documentos anteriores, essa consulta retorna todos os documentos em que o campo "Item" será igual a "Pen".

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

Recuperar documentos que correspondem a um documento incorporado

Para localizar todos os documentos que correspondem a um documento incorporado, use a operação `find()` com um documento de consulta que especifica o nome do documento incorporado e todos os campos e valores desse documento incorporado.

Ao vincular um documento incorporado, o documento incorporado do documento deve ter o mesmo nome que na consulta. Além disso, os campos e os valores no documento incorporado devem corresponder à consulta.

A consulta a seguir retorna apenas o documento "Poster Paint". Isso ocorre porque a "Caneta" tem valores diferentes para "`OnHand`" e "`MinOnHand`", e a "Tinta spray" tem um campo a mais (`OrderQnty`) que o documento de consulta.

```
db.example.find({"Inventory": {
```

```
    "OnHand": 47,
    "MinOnHand": 50 } } ).pretty()
```

Recuperar documentos que correspondem a um valor de campo em um documento incorporado

Para localizar todos os documentos que correspondem a um documento incorporado, use a operação `find()` com um documento de consulta que especifica o nome do documento incorporado e todos os campos e valores desse documento incorporado.

Considerando os documentos anteriores, a consulta a seguir usa "notação de pontos" para especificar o documento incorporado e os campos de interesse. Qualquer documento que seja correspondente será retornado, independentemente de quais outros campos possam estar presentes no documento incorporado. A consulta retorna "Poster Paint" e "Spray Paint", pois ambos correspondem aos campos e valores especificados.

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

Recuperar documentos que correspondem a uma matriz

Para localizar todos os documentos que correspondem a uma matriz, use a operação `find()` com o nome da matriz de interesse e todos os valores na matriz. A consulta retorna todos os documentos que têm uma matriz com esse nome, com valores idênticos aos da matriz, e na mesma ordem que na consulta.

A consulta a seguir retorna apenas o documento "Pen", pois o "Poster Paint" tem uma cor adicional (White), e "Spray Paint" tem as cores em ordem diferente.

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

Recuperar documentos que correspondem a um valor em uma matriz

Para localizar todos os documentos que possuem um valor de matriz específico, use a operação `find()` com o valor e o nome da matriz de interesse.

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

A operação anterior retorna todos os três documentos, pois cada um deles tem uma matriz chamada `Colors` e o valor `Red` em algum lugar da matriz. Se você especificar o valor "`White`", a consulta retornará apenas "Tinta de cartaz".

Recuperar documentos usando operadores

A consulta a seguir retorna todos os documentos em que o valor "`Inventory.OnHand`" é inferior a 50.

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

Para obter uma lista de operadores de consulta compatíveis, consulte Operadores de consulta e projeção.

Como atualizar documentos

Normalmente, seus documentos não são estáticos e são atualizados como parte de seus fluxos de trabalho de aplicações. Os exemplos a seguir mostram algumas maneiras de como atualizar documentos.

Para atualizar um documento existente, use a operação `update()`. A operação `update()` tem dois parâmetros de documento. O primeiro documento identifica qual(is) documento(s) atualizar. O segundo documento especifica as atualizações a fazer.

Ao atualizar um campo existente - seja esse campo um campo simples, uma matriz ou um documento incorporado - você especifica o nome do campo e seus valores. No final da operação, é como se o campo no documento antigo tivesse sido substituído pelo novo campo e por novos valores.

Tópicos

- Atualizar os valores de um campo existente
- Adição de um novo campo
- Substituir um documento incorporado
- Inserir novos campos em um documento incorporado
- Remover um campo de um documento
- Remover um campo de vários documentos

Atualizar os valores de um campo existente

Use os quatro documentos a seguir, que você adicionou anteriormente, para as seguintes operações de atualização.

```
{
    "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 atualizar um campo simples

Para atualizar um campo simples, use `update()` com `$set` para especificar o nome e o novo valor do campo. O exemplo a seguir altera o `Item` de "Pen" para "Gel Pen".

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

```
)
```

Os resultados dessa operação são semelhantes ao seguinte.

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

Para atualizar uma matriz

O exemplo a seguir substitui a matriz de cores existente por uma nova matriz que inclui `Orange` e descarta `White` da lista de cores. A nova lista de cores está na ordem especificada na operação `update()`.

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

Os resultados dessa operação são semelhantes ao seguinte.

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

Adição de um novo campo

Para modificar um documento adicionando um ou mais novos campos, use a operação `update()` com um documento de consulta que identifica o documento a ser inserido e os novos campos e valores a serem inseridos usando o operador `$set`.

O exemplo a seguir adiciona o campo `UnitPrice` com o valor `3.99` para o documento Spray Paints. Observe que o valor `3.99` é numérico, e não uma string.

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

Os resultados dessa operação são semelhantes ao seguinte (formato JSON).

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

Substituir um documento incorporado

Para modificar um documento substituindo um documento incorporado, use a operação `update()` com documentos que identificam o documento incorporado e seus novos campos e valores com o uso do operador `$set`.

Considerando o seguinte documento.

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

Substituir um documento incorporado

O exemplo a seguir substitui o documento de data atual por um novo que tem somente os campos `Month` e `Day`. O campo `Year` foi eliminado.

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

Os resultados dessa operação são semelhantes ao seguinte.

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

Inserir novos campos em um documento incorporado

Como adicionar campos a um documento incorporado

Para modificar um documento adicionando um ou mais novos campos a um documento incorporado, use a operação `update()` com documentos que identificam o documento incorporado e "notação de pontos" para especificar o documento incorporado e os novos campos e valores a serem inseridos usando o operador `$set`.

Considerando o documento a seguir, o código usa a "notação de pontos" para inserir os campos `Year` e `DoW` no documento `Date` incorporado, e `Words` no documento pai.

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

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

```
)
```

Os resultados dessa operação são semelhantes ao seguinte.

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

Remover um campo de um documento

Para modificar um documento removendo um campo do documento, use a operação `update()` com um documento de consulta que identifica o campo a ser removido do documento e o operador `$unset` para especificar o campo a ser removido.

O exemplo a seguir remove o campo `Words` do documento anterior.

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

Os resultados dessa operação são semelhantes ao seguinte.

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

Remover um campo de vários documentos

Para modificar um documento removendo um campo de vários documentos, use a operação `update()` com o operador `$unset` e a opção `multi` definida como `true`.

O exemplo a seguir remove o campo `Inventory` de todos os documentos na coleção de exemplo. Se um documento não tiver o campo `Inventory`, nenhuma ação será executada nesse documento. Se `multi: true` for omitido, a ação será executada apenas no primeiro documento que atenda ao critério.

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

Como excluir documentos

Para remover um documento do banco de dados, use a operação `remove()`, especificando qual documento remover. O código a seguir remove "Gel Pen" da sua coleção `example`.

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

Para remover todos os documentos do seu banco de dados, use a operação `remove()` com uma consulta vazia.

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

Conceitos básicos do Amazon DocumentDB

Há muitas formas de se conectar e começar a usar o Amazon DocumentDB. Este guia é a maneira mais rápida, simples e fácil para os usuários começarem a usar nosso poderoso banco de dados de documentos. Este guia usa AWS CloudShell para se conectar e consultar seu cluster do Amazon DocumentDB diretamente a partir do Console de gerenciamento da AWS. Novos clientes qualificados para o nível AWS gratuito podem usar o Amazon DocumentDB CloudShell gratuitamente. Se o seu AWS CloudShell ambiente ou cluster do Amazon DocumentDB fizer uso de recursos além do nível gratuito, serão cobradas AWS as taxas normais por esses recursos. Este guia ajudará você a começar a usar o Amazon DocumentDB em menos de cinco minutos.

> **Note**
>
> As instruções neste guia são específicas para criar e conectar-se a clusters baseados em instâncias do Amazon DocumentDB, nos quais o Amazon DocumentDB e o Amazon DocumentDB estão disponíveis. AWS CloudShell
>
> - Se você quiser criar e se conectar aos clusters elásticos do Amazon DocumentDB, consulte Como iniciar com clusters elásticos do Amazon DocumentDB.
> - Se você estiver localizado nas regiões AWS da China, consulteConectar o Amazon EC2 automaticamente.

Tópicos

- Pré-requisitos
- Etapa 1: criar um cluster
- Etapa 2: conectar-se ao seu cluster
- Etapa 3: inserir e consultar dados
- Etapa 4: explorar
- Início rápido do uso do Amazon DocumentDB CloudFormation

Pré-requisitos

Antes de criar o primeiro cluster do Amazon DocumentDB, você deve fazer o seguinte:

Crie uma conta (AWS) da Amazon Web Services

Antes de começar a usar o Amazon DocumentDB, você deve ter uma conta da Amazon Web Services (AWS). A AWS conta é gratuita. Você paga apenas pelos serviços e recursos usados.

Se você não tiver um Conta da AWS, conclua as etapas a seguir para criar um.

Para se inscrever em um Conta da AWS

1. Aberto https://signin.aws.amazon.com/signup? request_type=registrar.
2. Siga as instruções online.

   Parte do procedimento de inscrição envolve receber uma chamada telefônica ou uma mensagem de texto e inserir um código de verificação pelo teclado do telefone.

   Quando você se inscreve em um Conta da AWS, um Usuário raiz da conta da AWS é criado. O usuário-raiz tem acesso a todos os Serviços da AWS e recursos na conta. Como prática recomendada de segurança, atribua o acesso administrativo a um usuário e use somente o usuário-raiz para executar tarefas que exigem acesso de usuário-raiz.

Configure as permissões necessárias AWS Identity and Access Management (IAM).

O acesso para gerenciar recursos do Amazon DocumentDB, como clusters, instâncias e grupos de parâmetros de cluster, exige credenciais que AWS possam ser usadas para autenticar suas solicitações. Para obter mais informações, consulte Gerenciamento de identidade e Gerenciamento de acesso para o Amazon DocumentDB.

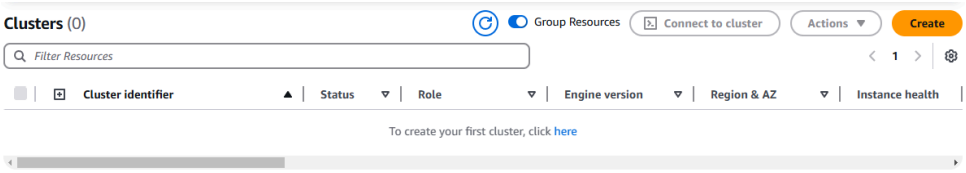
1. Na barra de pesquisa do Console de gerenciamento da AWS, insira **`IAM`** e selecione IAM no menu suspenso exibido.
2. Depois de chegar ao console do IAM, selecione Usuários no painel de navegação.
3. Selecione o nome de usuário.
4. Escolha Adicionar permissões.
5. Selecione Anexar políticas diretamente.
6. Digite `AmazonDocDBFullAccess` na barra de pesquisa e selecione-a quando ela aparecer nos resultados da pesquisa.
7. Escolha Próximo.
8. Escolha Adicionar permissões.

> **Note**
>
> Sua AWS conta inclui uma VPC padrão em cada região. Se você optar por usar uma Amazon VPC, conclua as etapas no tópico Criação de uma Amazon VPC no Guia do usuário da Amazon VPC.

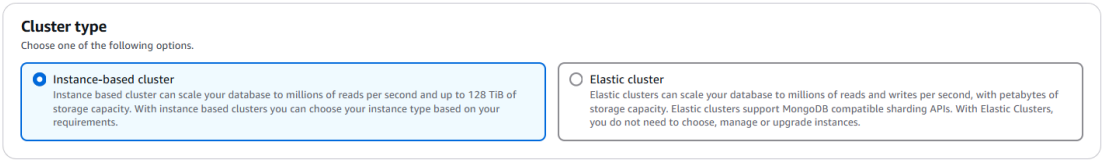
Etapa 1: criar um cluster

Nesta etapa, você criará um cluster do Amazon DocumentDB.

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No console de gerenciamento do Amazon DocumentDB, em Clusters, escolha Criar.



3. Na página Criar cluster do Amazon DocumentDB, na seção Tipo de cluster, escolha Instance-based cluster (essa é a opção padrão).

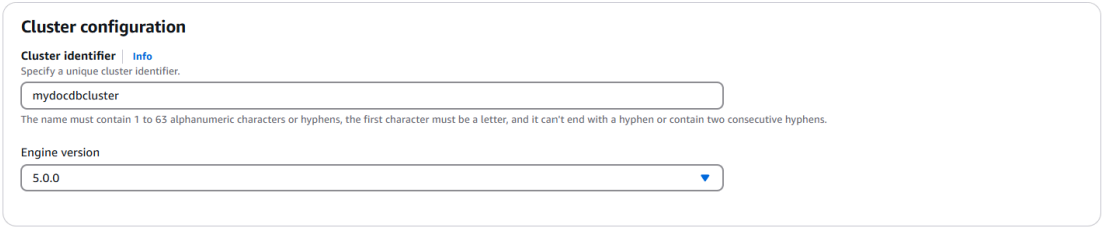


> **Note**
>
> A outra opção nessa categoria é o Cluster elástico. Para saber mais sobre os clusters elásticos do Amazon DocumentDB, consulte Usando clusters elásticos do Amazon DocumentDB.

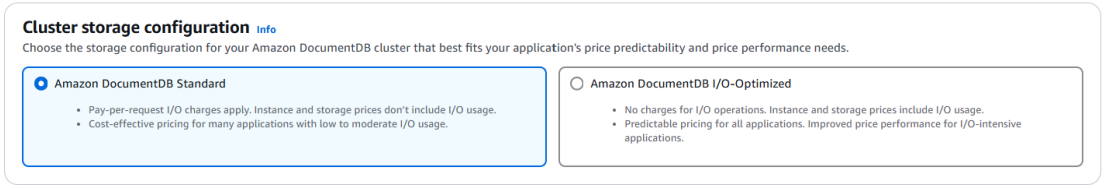
4. Na seção Configuração do cluster:

   a. Em Identificador do cluster, insira um nome exclusivo, como **`mydocdbcluster`**. Observe que o console mudará todos os nomes de cluster para minúsculas, independentemente de como eles sejam inseridos.

   b. Em Versão do mecanismo, escolha 5.0.0.



5. Na seção Configuração de armazenamento do cluster, escolha Amazon DocumentDB Padrão (essa é a opção padrão).



> **Note**
>
> A outra opção nessa categoria é o Amazon DocumentDB I/O-Optimized. Para saber mais sobre as opções, consulte Configurações de armazenamento de cluster do Amazon DocumentDB.

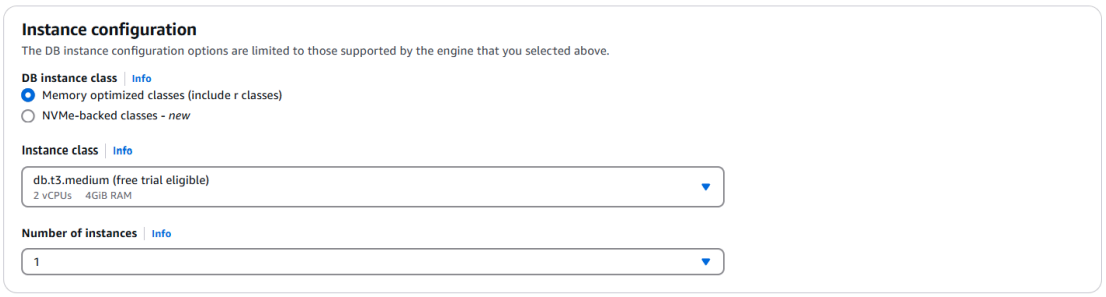
6. Na seção Instance configuration:

   a. Em Classe de instância de banco de dados, escolha Classes otimizadas para memória (incluir classes r) (esse é o padrão).

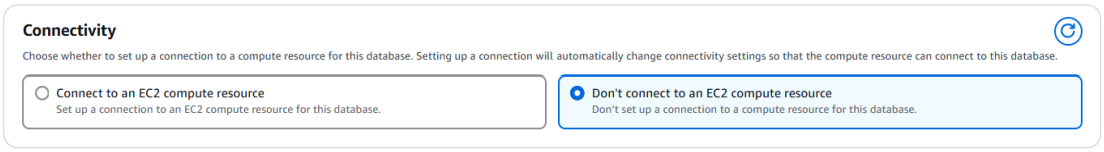
      A outra opção de instância são NVMe-backed classes. Para saber mais, consulte NVMe-backed instâncias.

   b. Em Classe de instância, escolha db.t3.medium. Isso é elegível para o teste AWS gratuito.

   c. Em Número de instâncias, escolha 1 instância. Escolher uma instância ajuda a minimizar os custos. Se for um sistema de produção, recomendamos que você provisione três instâncias para alta disponibilidade.

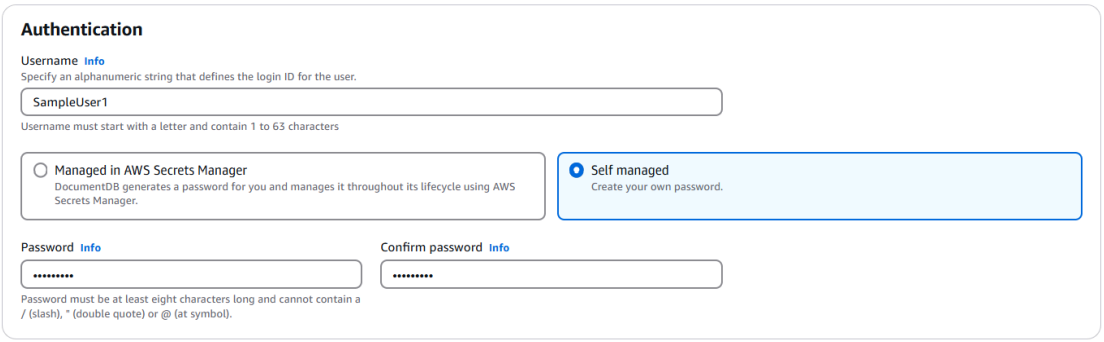


7. Na seção Conectividade, mantenha a configuração padrão Não se conectar a um recurso de computação do EC2.



8. Na seção Autenticação, insira um nome de usuário para o usuário principal e escolha Autogerenciado. Insira uma senha e confirme-a.

   Se, em vez disso, você escolheu Gerenciado em AWS Secrets Manager, consulte Gerenciamento de senhas com o Amazon DocumentDB e AWS Secrets Manager para obter mais informações.



9. Deixe todas as outras opções conforme o padrão e escolha Criar cluster.

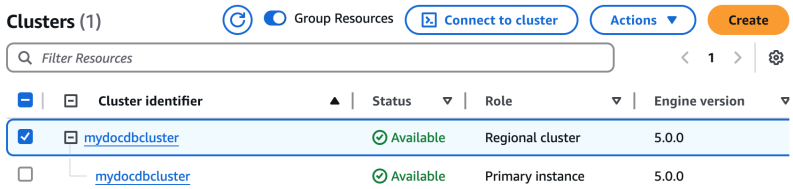
O Amazon DocumentDB agora está provisionando seu cluster, o que pode levar alguns minutos para ser concluído.

> **Note**
>
> Para obter informações sobre os valores de status do cluster, consulte Valores de status do cluster no capítulo Monitoramento do Amazon DocumentDB.

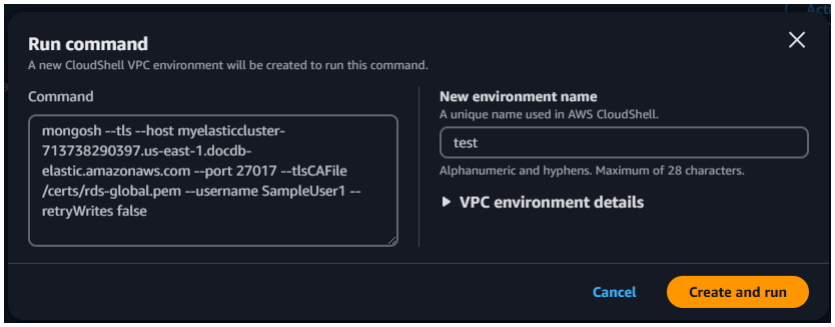
Etapa 2: conectar-se ao seu cluster

Conecte-se ao seu cluster Amazon DocumentDB usando AWS CloudShell.

1. No console de gerenciamento do Amazon DocumentDB, em Clusters, localize o cluster que você criou. Escolha o cluster clicando na caixa de seleção ao lado dele.



2. Escolha Conectar ao cluster (que fica ao lado do menu suspenso Ações). Esse botão será habilitado somente depois que você clicar na caixa de seleção ao lado do seu cluster e o status do cluster regional e da(s) instância(s) primária(S) forem exibidos como Disponível. A tela de comando CloudShell Executar é exibida.
3. No campo Nome do novo ambiente, insira um nome exclusivo, como "teste" e escolha Criar e executar. Os detalhes do ambiente da VPC são configurados automaticamente para o banco de dados do Amazon DocumentDB.



4. Quando solicitado, insira a senha que você criou na Etapa 1: criação de um cluster elástico do Amazon DocumentDB (subetapa 7).



Depois de você inserir sua senha e seu prompt se tornar `rs0 [direct: primary] <env-name>>`, a conexão ao seu cluster do Amazon DocumentDB será concluída com êxito.

> **Note**
>
> Para obter mais informações sobre solução de problemas, consulte Solução de problemas do Amazon DocumentDB.

Etapa 3: inserir e consultar dados

Agora que você está conectado ao seu cluster, pode executar algumas consultas para se familiarizar com o uso de um banco de dados de documentos.

1. Para inserir um único documento, digite o seguinte:

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

   Você obterá o seguinte resultado:

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

2. É possível ler o documento que escreveu com o comando `findOne()` (porque ele retorna apenas um único documento). Insira o seguinte:

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

   Você obterá o seguinte resultado:

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

3. Para realizar mais algumas consultas, considere um caso de uso de perfis de jogos. Primeiro, insira algumas entradas em uma coleção intitulada `profiles`. Insira o seguinte:

   ```
   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 }
   ```

```
])
```

Você obterá o seguinte resultado:

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

4. Use o `find()` comando para retornar todos os documentos na coleção de perfis. Insira o seguinte:

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

Você receberá uma saída que corresponderá aos dados digitados na Etapa 3.

5. Use uma consulta para um único documento por meio de um filtro. Insira o seguinte:

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

Você obterá o seguinte resultado:

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

6. Agora vamos tentar encontrar um perfil e modificá-lo usando o comando `findAndModify`. Atribuiremos ao usuário Matt mais 10z pontos com o código a seguir:

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

Você obtém o seguinte resultado (observe que a pontuação dele ainda não aumentou):

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

7. É possível verificar se a pontuação dele mudou com a seguinte consulta:

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

Você obterá a seguinte saída:

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

Etapa 4: explorar

Parabéns! Você concluiu com êxito o Guia de conceitos básicos para clusters baseados em instâncias do Amazon DocumentDB.

E depois? Saiba como aproveitar ao máximo esse banco de dados com alguns de seus atributos populares:

- Gerenciando o Amazon DocumentDB
- Escalabilidade
- Fazer backup e restaurar

> **Note**
>
> O cluster que você criou a partir desse exercício introdutório continuará a gerar custos, a menos que seja excluído. Para obter instruções, consulte Exclusão de um cluster do Amazon DocumentDB.

Início rápido do uso do Amazon DocumentDB CloudFormation

Esta seção contém etapas e outras informações para ajudar você a começar a usar rapidamente o Amazon DocumentDB (compatível com MongoDB) usando o CloudFormation. Para obter informações gerais sobre o Amazon DocumentDB, consulte O que é Amazon DocumentDB (compatível com MongoDB).

Essas instruções usam um AWS CloudFormation modelo para criar um cluster e instâncias em sua Amazon VPC padrão. Para obter instruções sobre a criação desses recursos por conta própria, consulte Conceitos básicos do Amazon DocumentDB.

> **Important**
>
> A CloudFormation pilha criada por esse modelo cria vários recursos, incluindo recursos no Amazon DocumentDB (por exemplo, um cluster e instâncias) e no Amazon Elastic Compute Cloud (por exemplo, um grupo de sub-redes).
> Alguns desses recursos não são recursos de nível gratuito. Para obter informações de precificação, consulte Precificação do Amazon DocumentDB e Precificação do Amazon EC2.
> É possível excluir a pilha ao terminar de usá-la para interromper as cobranças.

Essa CloudFormation pilha é destinada apenas para fins tutoriais. Se você usar esse modelo para um ambiente de produção, use políticas e segurança de IAM mais rígidas. Para obter informações sobre a proteção de recursos, consulte Segurança do Amazon VPC e Segurança e rede do Amazon EC2.

Tópicos

- Pré-requisitos
- Lançamento de um Amazon DocumentDB CloudFormation pilha
- Acessar o cluster do Amazon DocumentDB
- Proteção contra encerramento e exclusão

Pré-requisitos

Antes de criar um cluster do Amazon DocumentDB, você deve fazer o seguinte:

- Uma Amazon VPC padrão
- As permissões necessárias do IAM

Permissões obrigatórias do IAM

As permissões a seguir permitem que você crie recursos para a pilha do CloudFormation :

AWS Políticas gerenciadas

- `AWSCloudFormationReadOnlyAccess`
- `AmazonDocDBFullAccess`

Permissões do IAM adicionais

A política a seguir descreve as permissões adicionais necessárias para criar e excluir essa CloudFormation pilha.

Nos exemplos a seguir, substitua cada um *`user input placeholder`* pelas informações do seu recurso.

Par de chaves do Amazon EC2

Você deve ter um par de chaves (e o arquivo PEM) disponível na região em que você criará a CloudFormation pilha. Se precisar criar um par de chaves, consulte Criar um par de chaves usando o Amazon EC2 no Guia do usuário do Amazon EC2.

Lançamento de um Amazon DocumentDB CloudFormation pilha

Esta seção descreve como executar e configurar uma pilha CloudFormation do Amazon DocumentDB.

1. Faça login no Console de gerenciamento da AWS athttps://console.aws.amazon.com/.
2. A tabela a seguir lista os modelos de pilha do Amazon DocumentDB para cada Região da AWS. Escolha Launch Stack para o qual Região da AWS você deseja lançar sua pilha.

Região	Visualizar modelo	Visualizar no Designer	início
Leste dos EUA (Ohio)	Visualizar modelo	Visualizar no Designer	Launch Stack
Leste dos EUA (Norte da Virgínia)	Visualizar modelo	Visualizar no Designer	Launch Stack
Oeste dos EUA (Oregon)	Visualizar modelo	Visualizar no Designer	Launch Stack
Ásia-Pacífico (Mumbai)	Visualizar modelo	Visualizar no Designer	Launch Stack
Ásia-Pacífico (Seul)	Visualizar modelo	Visualizar no Designer	Launch Stack

Região	Visualizar modelo	Visualizar no Designer	início
Ásia-Pacífico (Singapura)	Visualizar modelo	Visualizar no Designer	Launch Stack
Ásia-Pacífico (Sydney)	Visualizar modelo	Visualizar no Designer	Launch Stack
Ásia-Pacífico (Tóquio)	Visualizar modelo	Visualizar no Designer	Launch Stack
Canadá (Central)	Visualizar modelo	Visualizar no Designer	Launch Stack
Europa (Frankfurt)	Visualizar modelo	Visualizar no Designer	Launch Stack
Europa (Irlanda)	Visualizar modelo	Visualizar no Designer	Launch Stack
Europa (Londres)	Visualizar modelo	Visualizar no Designer	Launch Stack
Europa (Paris)	Visualizar modelo	Visualizar no Designer	Launch Stack

3. Criar pilha - descreve o modelo do Amazon DocumentDB selecionado. Cada pilha é baseada em um modelo — um arquivo JSON ou YAML — que contém a configuração sobre os AWS recursos que você deseja incluir na pilha. Como você optou por iniciar uma pilha a partir dos modelos fornecidos acima, seu modelo já foi configurado para criar uma pilha do Amazon DocumentDB para a Região da AWS sua escolha.

   Quando você inicia uma CloudFormation pilha, a proteção contra exclusão do seu cluster Amazon DocumentDB é desativada por padrão. Se desejar habilitar a proteção contra exclusão para o cluster, conclua as etapas a seguir. Caso contrário, escolha Próximo para seguir para a próxima etapa.

   Para habilitar a proteção contra exclusão para o cluster do Amazon DocumentDB:

1. Escolha Visualizar no Designer, no canto inferior direito da página Criar pilha.
2. Modifique o modelo usando o editor JSON e YAML integrado na página CloudFormation Designer resultante do console. Role até a seção `Resources` e modifique-a para incluir `DeletionProtection`, conforme segue. Para obter mais informações sobre como usar o CloudFormation Designer, consulte O que é o 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. Escolha Criar pilha ()

no canto superior esquerdo da página para salvar as alterações e criar uma pilha com essas alterações ativadas.

4. Depois de salvar as alterações, você será redirecionado para a página Criar pilha.
5. Escolha Próximo para continuar.

4. Especificar detalhes da pilha - insira o nome e os parâmetros da pilha do modelo. Os parâmetros são definidos em seu modelo e permitem que você insira valores personalizados ao criar ou atualizar uma pilha.
   - Em Nome da pilha, insira um nome para a pilha ou aceite o nome fornecido. O nome da pilha pode incluir letras (A – Z e a –z), números (0 – 9) e traços (–).
   - Em Parâmetros, insira os seguintes detalhes:
     - DBClusterName— Insira um nome para seu cluster Amazon DocumentDB ou aceite o nome fornecido.

       Restrições de nomeação de cluster:
       - O comprimento é de [1 a 63] letras, números ou hífens.
       - O primeiro caractere deve ser uma letra.
       - Não pode terminar com um hífen ou conter dois hifens consecutivos.
       - Deve ser exclusivo para todos os clusters no Amazon RDS, Neptune e Amazon DocumentDB por Conta da AWS região.
     - DBInstanceClass— Na lista suspensa, selecione a classe de instância para seu cluster Amazon DocumentDB.
     - DBInstanceName— Insira um nome para sua instância do Amazon DocumentDB ou aceite o nome fornecido.

       Restrições de nomenclatura da instância:
       - O comprimento é de [1 a 63] letras, números ou hífens.
       - O primeiro caractere deve ser uma letra.
       - Não pode terminar com um hífen ou conter dois hifens consecutivos.
       - Deve ser exclusivo para todas as instâncias do Amazon RDS, Neptune e Amazon DocumentDB por Conta da AWS região.
     - MasterPassword— A senha da conta do administrador do banco de dados.

- MasterUser— O nome de usuário da conta do administrador do banco de dados. Eles MasterUser devem começar com uma letra e podem conter somente caracteres alfanuméricos.

Escolha Próximo para salvar as alterações e continuar.

5. Configurar opções de pilha - configure as tags, permissões e opções adicionais da pilha.

   - Tags - especifique os pares de tags (chave/valor) a serem aplicados aos recursos na pilha. É possível adicionar até 50 tags exclusivas para cada pilha.
   - Permissões: opcionais. Escolha uma função do IAM para definir explicitamente como CloudFormation criar, modificar ou excluir recursos na pilha. Se você não escolher uma função, CloudFormation usa permissões com base nas suas credenciais de usuário. Antes de especificar um perfil de serviço, certifique-se de ter permissão para aprová-la (`iam:PassRole`). A permissão `iam:PassRole` especifica quais perfis você pode usar.

     > **Note**
     >
     > Quando você especifica uma função de serviço, CloudFormation sempre usa essa função para todas as operações que são executadas nessa pilha. Outros usuários com permissão para executar operações nessa pilha poderão usar esse perfil, mesmo que não tenham permissão para aprová-la. Se o perfil inclui permissões que o usuário não precisa, você pode ampliar involuntariamente as permissões de um usuário. Certifique-se de que o perfil conceda o privilégio mínimo.

   - Opções avançadas - você pode definir as seguintes opções avançadas:

     - Política de pilha: opcional. Define os recursos que você deseja proteger contra atualizações não intencionais durante uma atualização da pilha. Por padrão, todos os recursos podem ser atualizados durante uma atualização da pilha.

       É possível inserir a política de pilha diretamente como JSON ou fazer upload de um arquivo JSON que contém a política de pilha. Para obter mais informações, consulte Prevenir atualizações de recursos de pilha.
     - Configuração de reversão: opcional. Especifique CloudWatch os alarmes de registros CloudFormation para monitorar ao criar e atualizar a pilha. Se a operação ultrapassar um limite de alarme, ela será CloudFormation revertida.
     - Opções de notificação: opcional. Especifique tópicos para o Simple Notification System (SNS).

- Opções de criação de pilha: opcionais. É possível especificar as seguintes opções:
  - Reversão em caso de falha - se a pilha deve ou não ser revertida em caso de falha na criação.
  - Tempo limite - o número de minutos antes da criação da pilha expirar.
  - Proteção contra encerramento - impede que a pilha seja excluída acidentalmente.

> **Note**
>
> CloudFormation a proteção contra encerramento é diferente do conceito de proteção contra exclusão do Amazon DocumentDB. Para obter mais informações, consulte Proteção contra encerramento e exclusão.

Escolha Próximo para continuar.

6. Analisar <stack-name> - analise o modelo, os detalhes e as opções de configuração da pilha. Você também pode abrir um link de quick-create na parte inferior da página para criar pilhas com estas mesmas configurações básicas.

   - Selecione Criar para criar a pilha.
   - Como alternativa, você pode escolher Criar conjunto de alterações. Um conjunto de alterações é uma pré-visualização de como esta pilha será configurada antes de sua criação. Isso permite que você examine diversas configurações antes de executar o conjunto de alterações.

Acessar o cluster do Amazon DocumentDB

Depois que a CloudFormation pilha for concluída, você poderá usar uma instância do Amazon EC2 para se conectar ao seu cluster Amazon DocumentDB. Para obter informações sobre como se conectar a uma instância do Amazon EC2, consulte Conectar-se à sua instância do Linux no Guia do usuário do Amazon EC2.

Quando estiver conectado, consulte as seguintes seções, que contêm informações sobre o uso do Amazon DocumentDB.

- Etapa 3: inserir e consultar dados
- Gerenciamento de clusters e instâncias do Amazon DocumentDB
- Monitoramento e registro no Amazon DocumentDB

Proteção contra encerramento e exclusão

É uma das melhores práticas do Amazon DocumentDB habilitar a proteção contra exclusão e a proteção contra encerramento. CloudFormation a proteção de encerramento é um recurso nitidamente diferente do recurso de proteção contra exclusão do Amazon DocumentDB.

- Proteção de encerramento — Você pode evitar que uma pilha seja excluída acidentalmente ativando a proteção de encerramento para sua CloudFormation pilha. Se um usuário tentar excluir uma pilha com proteção contra encerramento habilitada, a exclusão falhará e a pilha, incluindo seu status, permanecerá inalterada. A proteção contra encerramento é desativada por padrão quando você cria uma pilha usando CloudFormation. É possível ativar a proteção de encerramento em uma pilha ao criá-la. Para obter mais informações, consulte Definindo opções CloudFormation de pilha.
- Proteção contra exclusão - o Amazon DocumentDB também oferece a capacidade de habilitar a proteção contra exclusão para um cluster. Se um usuário tentar excluir um cluster do Amazon DocumentDB com a proteção de exclusão habilitada, a exclusão falhará e o cluster permanecerá inalterado. A proteção contra exclusão, quando ativada, protege contra exclusões acidentais do Amazon DocumentDB Console de gerenciamento da AWS, e. AWS CLI CloudFormation Para obter mais informações sobre como habilitar e desabilitar a proteção contra exclusão para um cluster do Amazon DocumentDB, consulte Deletion protection (Proteção contra exclusão).

Migração e atualização do Amazon DocumentDB

O Amazon DocumentDB é um serviço de banco de dados gerenciado compatível com a API do MongoDB. É possível migrar dados do bancos de dados do MongoDB para o Amazon DocumentDB em execução on-premises ou no Amazon Elastic Compute Cloud (Amazon EC2) usando o processo detalhado nesta seção.

Tópicos

- Migre para o Amazon DocumentDB usando AWS Serviço de migração de banco de dados (DMS): guia de início rápido
- Runbook de migração do Amazon DocumentDB
- Migração do Couchbase Server
- Datas de suporte da versão do mecanismo do Amazon DocumentDB
- Atualização da versão principal implementada do Amazon DocumentDB no local
- Atualização da versão secundária do Amazon DocumentDB
- Atualizando seu cluster Amazon DocumentDB usando AWS Database Migration Service
- Usar um release de de suporte de longo prazo (LTS)
- Suporte estendido do Amazon DocumentDB

Migre para o Amazon DocumentDB usando AWS Serviço de migração de banco de dados (DMS): guia de início rápido

Tópicos

- Preparação do arquivo de origem
- Configuração do DMS
- Habilitação da compactação do DocumentDB
- Criação de uma tarefa de replicação
- Monitorar o andamento
- Mais informações

Preparação do arquivo de origem

Consulte Ativar fluxos de alterações para habilitar fluxos de alterações do DocumentDB ou para habilitar o MongoDB Oplog para oferecer suporte à captura de dados de alteração (CDC) do DMS.

- A fonte do DMS deve reter todas as alterações em andamento até que a carga total do DMS seja concluída para todas as coleções incluídas.
- Os fluxos de alterações do DocumentDB são baseados no tempo. Certifique-se de que sua configuração `change_stream_log_retention_duration` seja grande o suficiente para cobrir o tempo necessário para concluir a carga completa.
- O MongoDB Oplog tem um tamanho fixo. Certifique-se de que ele seja dimensionado para comportar todas as operações durante a carga total.

Configuração do DMS

Crie endpoints de instância, origem e destino do DMS e teste cada endpoint.

Habilitação da compactação do DocumentDB

Amazon DocumentDB 5.0: ative a compactação anexando um grupo de parâmetros personalizado ao seu cluster do Amazon DocumentDB e definindo o `default_collection_compression` parâmetro como. `enabled` Consulte Gerenciamento da compactação de documentos a nível de coleção para obter mais informações.

Amazon DocumentDB 8.0: a compactação Zstd é habilitada por padrão. Consulte Gerenciando a compactação baseada em dicionário no Amazon DocumentDB 8.0 para obter mais informações.

Criação de uma tarefa de replicação

1. No console do DMS, no painel de navegação, escolha Migrar ou replicar e, em seguida, escolha Tarefas.
2. Escolha Criar tarefa.
3. Na página Criar tarefa, na seção Configuração da tarefa:
    - Insira um Identificador de tarefa exclusivo e significativo (por exemplo, “mongodb-docdb-replication”).
    - Escolha o endpoint de origem que você criou anteriormente no menu suspenso Endpoint de banco de dados de origem.

- Escolha o endpoint de destino que você criou anteriormente no menu suspenso Endpoint de banco de dados de destino.
- Em Tipo de tarefa, escolha Migrar e replicar.

4. Na seção Configurações:

   - Para registros de tarefas, marque a caixa Ativar CloudWatch registros.
   - Em Modo de edição (na parte superior da seção), escolha Editor JSON e defina os atributos a seguir:
     - Defina `ParallelApplyThreads` como 5 (em `TargetMetadata`). Isso permite aproximadamente 1000 operações insert/update de exclusão por segundo no CDC.
     - Defina `MaxFullLoadSubTasks` como 16 (em `FullLoadSettings`). Considere aumentar isso dependendo do tamanho da sua instância.
     - Para coleções grandes (acima de 100 GB), habilite a partição automática (em Mapeamento de tabelas e sob o atributo `parallel-load`):
       - "type": "partitions-auto"
       - "number-of-partitions": 16

Monitorar o andamento

Use o AWS DMS console ou crie um painel personalizado (ferramenta de painel) para acompanhar a migração. Concentre-se nas métricas a seguir:

- FullLoadThroughputBandwidthTarget— Mede a largura de banda de rede (in KB/second) usada pelo DMS ao transferir dados para o banco de dados de destino durante a fase de carregamento total da migração.
- CDCLatencyTarget— mede o tempo de atraso (em segundos) entre a ocorrência de uma alteração no banco de dados de origem e o momento em que essa alteração é aplicada ao banco de dados de destino.
- CDCThroughputRowsTarget— mede o número de linhas por segundo que o DMS está aplicando ao banco de dados de destino durante a fase contínua de replicação da migração.

Mais informações

Para obter mais informações sobre o Amazon DocumentDB e AWS DMS, consulte:

- Runbook de migração do Amazon DocumentDB
- Migração do MongoDB para o Amazon DocumentDB

Runbook de migração do Amazon DocumentDB

Este runbook fornece um guia abrangente para migrar um banco de dados MongoDB para o Amazon DocumentDB usando o (DMS). AWS Database Migration Service Ele foi projetado para oferecer suporte a administradores de banco de dados, engenheiros de nuvem e desenvolvedores em toda a jornada de migração de ponta a ponta, desde a descoberta inicial até a validação pós-migração.

Dadas as diferenças na implementação e nos recursos com suporte entre o MongoDB e o Amazon DocumentDB, este runbook enfatiza uma abordagem estruturada e sistemática. Ele descreve as avaliações essenciais de pré-migração, destaca as considerações de compatibilidade e detalha as principais tarefas necessárias para garantir uma migração com êxito e com o mínimo de interrupções.

O runbook está organizado nos tópicos a seguir:

- Compatibilidade: noções básicas dos recursos e tipos de dados do MongoDB com suporte no Amazon DocumentDB e identificação de possíveis incompatibilidades.
- Descoberta de workload— analise as cargas de trabalho existentes do MongoDB, incluindo read/write padrões, volumes de dados e linhas de base de desempenho.
- Migração de índices: análise de estratégias para extrair e transformar índices do MongoDB para obter uma performance ideal no Amazon DocumentDB.
- Migração do usuário: detalhamento da abordagem para a migração de usuários, perfis e controles de acesso do banco de dados para o Amazon DocumentDB.
- Migrações de dados— Aborde vários métodos de uso da migração de dados AWS DMS, incluindo carregamento total e captura de dados de alteração (CDC).
- Monitoramento: detalhamento de várias abordagens de monitoramento na migração usando o DMS ou ferramentas nativas.
- Validação: fornecimento de procedimentos para verificações de integridade de dados, validação funcional e comparação de performance após a migração.

Seguindo as orientações deste runbook, as equipes podem garantir uma transição tranquila, segura e eficiente para o Amazon DocumentDB, preservando a funcionalidade da aplicação e minimizando os riscos.

Compatibilidade

Tópicos

- compatibilidade de recursos principais
- Ferramenta de avaliação de compatibilidade do Amazon DocumentDB

Ao migrar do MongoDB para o Amazon DocumentDB, uma avaliação inicial completa e uma verificação de compatibilidade de recursos são essenciais para uma migração com êxito. Esse processo começa com um inventário abrangente dos recursos do MongoDB, incluindo operadores de pipeline de agregação, padrões de consulta, índices e modelos de dados.

Como o Amazon DocumentDB é compatível com as APIs MongoDB 3.6, 4.0, 5.0 e 8.0, os aplicativos que usam recursos mais novos MongoDB-specific podem exigir refatoração. As áreas críticas a serem avaliadas incluem mecanismos de fragmentação (o Amazon DocumentDB usa uma abordagem diferente), implementações de transações, funcionalidade de fluxos de mudança e tipos de índice (especialmente índices esparsos e parciais).

As características de performance também diferem, com o Amazon DocumentDB otimizado para workloads corporativas com performance previsível. Os testes devem envolver a execução de workloads representativas em ambos os sistemas para identificar padrões de consulta que possam precisar de otimização.

Monitorar os planos de execução para detectar possíveis lacunas de performance é importante durante a fase de avaliação. Isso ajuda a criar um roteiro de migração claro, identificando as mudanças necessárias nas aplicações e estabelecendo cronogramas realistas para uma transição suave.

compatibilidade de recursos principais

Suporte a recursos abrangentes

- Operações CRUD: aproveite o suporte completo a todas as operações básicas de criação, leitura, atualização e exclusão, incluindo operadores em massa e de consulta, oferecendo compatibilidade perfeita com aplicações.
- Recursos avançados de indexação: aproveite o suporte abrangente para índices de campo único, compostos, TTL, parciais, esparsos e 2dsphere, para otimizar a performance de suas consultas e índices de texto (versão 5) para pesquisas baseadas em texto.

- Enterprise-grade replicação — beneficie-se de um mecanismo robusto de failover automático com réplicas de leitura para uma alta disponibilidade superior sem sobrecarga operacional.
- Soluções avançadas de backup — Fique tranquilo com o sistema de backup automatizado com Point-in-Time recuperação (PITR) e instantâneos manuais sob demanda para proteção de dados.

Avançado AWS-recursos integrados

- Agregação simplificada: aproveite os estágios de agregação mais usados (`$match`, `$group`, `$sort`, `$project`, etc.) com performance otimizada para workloads corporativas.
- Suporte a transações: implemente transações com vários documentos e várias coleções, perfeitas para a maioria das necessidades das aplicações de negócios.
- Real-time rastreamento de dados — habilite fluxos de alterações por meio de um comando simples e aumente o período de retenção do fluxo de alterações por meio de uma configuração simples de grupo de parâmetros para monitoramento de alterações de dados em tempo real.
- Location-based serviços — Implemente aplicações geoespaciais com suporte para índices de `$geoNear` operadores e 2dsphere.
- Recursos de pesquisa de texto: utilize a funcionalidade de pesquisa de texto integrada para as necessidades de descoberta de conteúdo.

Vantagens da arquitetura moderna

- Cloud-native design — desfrute AWS de uma arquitetura otimizada que substitui recursos legados, como operações MapReduce de pipeline de agregação mais eficientes.
- Segurança aprimorada — Beneficie-se da autenticação de X.509 certificados AWS Identity and Access Management (IAM) SCRAM-SHA-1 SCRAM-SHA-256,, e da autenticação baseada em senha.
- Performance previsível: experimente uma performance consistente otimizada especificamente para workloads corporativas.

Para obter uma visão geral abrangente dos recursos do Amazon DocumentDB, consulte APIs, operações e tipos de dados do MongoDB compatíveis no Amazon DocumentDB e Diferenças funcionais: Amazon DocumentDB e MongoDB para maximizar o potencial do seu banco de dados.

O Amazon DocumentDB não oferece suporte a todos os índices oferecidos pelo MongoDB. Nós fornecemos uma ferramenta de indexação gratuita GitHub para verificar a compatibilidade.

Execute a ferramenta de indexação para avaliar a incompatibilidade e planejar soluções alternativas adequadamente.

Ferramenta de avaliação de compatibilidade do Amazon DocumentDB

A ferramenta de compatibilidade do Amazon DocumentDB GitHub analisa sua MongoDB carga de trabalho e relata operadores, comandos e recursos que não são suportados ou exigem alterações no Amazon DocumentDB. Ele suporta versões de destino de 3.6 a 8.0.1 (padrão: 8.0.1).

Recomendado: modo URI (MongoDB5.0+)

O modo URI se conecta diretamente à sua instância do MongoDB, faz amostras de operações em tempo real e não requer criação de perfil ou acesso a registros.

Métodos alternativos

- Log-based análise — Analisa os registros do MongoDB perfilador. Captura o comportamento real do tempo de execução, mas exige que a criação de perfil esteja habilitada e cobre apenas o período de registro.
- Análise do código-fonte — Escaneia o código-fonte do aplicativo para uso MongoDB da API. Fornece cobertura abrangente, mas pode sinalizar caminhos de código não utilizados e não pode detectar consultas construídas dinamicamente.

Pré-requisitos

- Python3,7+
- Clone a ferramenta: `git clone https://github.com/awslabs/amazon-documentdb-tools.git`
- Instalar dependências: `pip install -r compat-tool/requirements.txt`

Para obter detalhes completos de uso e todas as opções disponíveis, consulte o README em GitHub.

Descoberta de workload

A migração do MongoDB para o Amazon DocumentDB exige uma compreensão completa da workload do banco de dados existente. A descoberta da workload é o processo de analisar os padrões de uso do banco de dados, as estruturas de dados, a performance das consultas e as dependências operacionais para garantir uma transição perfeita com o mínimo de interrupção.

Esta seção descreve as principais etapas envolvidas na descoberta da workload para facilitar uma migração efetiva do MongoDB para o Amazon DocumentDB.

Tópicos

- Avaliação da implantação existente do MongoDB
- Identificação de diferenças no modelo de dados
- Análise de consultas e de performance
- Verificação de segurança e controle de acesso
- Considerações operacionais e de monitoramento

Avaliação da implantação existente do MongoDB

Antes da migração, é fundamental avaliar o ambiente atual do MongoDB, incluindo:

- Arquitetura do cluster: identifique o número de nós, conjuntos de réplicas e configurações de fragmentação. Ao migrar do MongoDB para o Amazon DocumentDB, entender sua configuração de fragmentação do MongoDB é importante porque o Amazon DocumentDB não oferece suporte à fragmentação controlada pelo usuário. As aplicações projetadas para um ambiente fragmentado do MongoDB precisarão de mudanças arquitetônicas, pois o Amazon DocumentDB usa uma abordagem de escalabilidade diferente com sua arquitetura baseada em armazenamento. Você precisará adaptar sua estratégia de distribuição de dados e possivelmente consolidar coleções fragmentadas ao migrar para o Amazon DocumentDB.
- Armazenamento e volume de dados: meça o tamanho total dos dados e o tamanho do índice do seu cluster. Complemente isso com a ferramenta de revisão Oplog para entender os padrões de gravação e a velocidade de crescimento dos dados. Para obter mais informações sobre como dimensionar seu cluster, consulte Dimensionamento de instância.
- Padrões de workload: analise o throughput de leitura e gravação, a frequência de execução de consultas e a eficiência da indexação.
- Dependências operacionais: documente todas as aplicações, serviços e integrações que dependam do MongoDB.

Identificação de diferenças no modelo de dados

Embora o Amazon DocumentDB seja MongoDB-compatible, existem diferenças nos recursos suportados, como:

- Transações: o Amazon DocumentDB oferece suporte a transações ACID, mas com algumas Limitações.
- Design do esquema: garanta que as estruturas dos documentos, os documentos incorporados e as referências estejam alinhados às práticas recomendadas do Amazon DocumentDB.

Análise de consultas e de performance

Compreender o comportamento das consultas ajuda a otimizar a migração e a performance pós-migração. As principais áreas a serem analisadas incluem:

- Consultas lentas: identifique consultas com alto tempo de execução usando as ferramentas de criação de perfil do MongoDB.
- Padrões de consulta: categorize os tipos de consulta comuns, incluindo operações e agregações CRUD.
- Uso de índices avalie se os índices são utilizados de forma eficaz ou se precisam de otimização no Amazon DocumentDB. Para avaliar o uso de índices e otimizar a performance no Amazon DocumentDB, use o estágio do pipeline de agregação `$indexStats` combinado com o método `explain()` em suas consultas críticas. Comece executando `db.collection.aggregate([{$indexStats{}}])` para identificar quais índices estão sendo usados. É possível fazer uma análise mais detalhada executando suas consultas mais frequentes com `explainPlan`.
- Concorrência e distribuição de workloads: avalie as taxas de leitura e gravação, o pool de conexões e os gargalos de performance.

Verificação de segurança e controle de acesso

Autenticação e autorização

- MongoDB RBAC para Amazon DocumentDB IAM e RBAC — Mapeie os usuários e funções de controle de acesso baseados em funções do MongoDB às políticas (IAM) e aos AWS Identity and Access Management usuários de autenticação SCRAM do Amazon DocumentDB.
- Estratégia de migração de usuários: planeje a migração de usuários do banco de dados, perfis personalizados e privilégios para os mecanismos de autenticação com suporte no Amazon DocumentDB.
- Diferenças de privilégios: identifique os privilégios do MongoDB sem equivalentes diretos do Amazon DocumentDB (por exemplo, perfis de administração de cluster).

- Autenticação de aplicações: atualize strings de conexão e gerenciamento de credenciais para as políticas de senha do Amazon DocumentDB. É possível usar o gerenciador de segredos para armazenar suas credenciais e alternar senhas.
- Gerenciamento da conta de serviço — Estabeleça processos para gerenciar as credenciais da conta de serviço no AWS Secrets Manager.
- Implementação de privilégios mínimos: revise e refine os controles de acesso para implementar os princípios de privilégios mínimos no novo ambiente.

Criptografia

Garanta que a criptografia em repouso e em trânsito esteja alinhada aos requisitos de conformidade.

Configuração de rede

Planeje a configuração da nuvem privada virtual (VPC) e as regras do grupo de segurança.

Considerações operacionais e de monitoramento

Para manter a confiabilidade do sistema, a descoberta de workloads também deve incluir:

- Estratégia de backup e restauração: avalie os métodos de backup existentes e os recursos de backup do Amazon DocumentDB.
- AWS Backup integração — Aproveite o gerenciamento centralizado AWS Backup de backup em todos AWS os serviços, incluindo o Amazon DocumentDB.
- CloudWatch métricas — Mapeie as métricas de monitoramento do MongoDB com as CloudWatch métricas do Amazon DocumentDB para CPU, memória, conexões e armazenamento.
- Insights de performance: implemente os Insights de performance do Amazon DocumentDB para visualizar a carga do banco de dados e analisar problemas de performance com análises detalhadas das consultas.
- Profiler — Configure o Amazon DocumentDB Profiler para capturar operações de execução lenta (semelhante ao profiler do MongoDB, mas com configurações da Amazon). DocumentDB-specific
  - Habilite por meio de grupos de parâmetros com limites apropriados.
  - Analise os dados do profiler para identificar oportunidades de otimização
- CloudWatch Eventos — Configure o monitoramento orientado por eventos para eventos de cluster do Amazon DocumentDB.
  - Configure notificações para eventos de backup, janelas de manutenção e failovers.

  - Integre-se ao Amazon SNS para alertas e respostas AWS Lambda automatizadas.
- Logs de auditoria: planeje a configuração dos logs de auditoria para rastrear a atividade do usuário e eventos relevantes à segurança.
- Monitoramento aprimorado — habilite o monitoramento aprimorado para OS-level métricas granulares em intervalos de 1 segundo.

Criar novo cluster de destino

Crie um cluster Amazon DocumentDB seguindo as etapas emCriar um cluster do Amazon DocumentDB. Ao configurar o cluster para migração, aplique estas recomendações:

- Classe de instância: escolha R8G para obter o melhor desempenho. Se o R8G não estiver disponível na sua região de destino, use o R6G. Escolha a instância mais grande possível para obter a melhor taxa de transferência de carga total. Diminua a escala após a conclusão da migração.
- Versão do motor: mais recente (8.0.1)
- Número de instâncias: escolha 1 instância para minimizar os custos durante a migração. Escale para 3 instâncias para obter alta disponibilidade após a conclusão da migração de carga total.
- Configurações de rede: certifique-se de que seu grupo de segurança do Amazon DocumentDB permita conexões de entrada do grupo de segurança da instância de replicação do DMS (para migrações on-line) ou do host de migração do EC2 (para migrações offline usando mongorestore).

Migração de índices

A migração do MongoDB para o Amazon DocumentDB envolve a transferência não apenas de dados, mas também de índices para manter a performance das consultas e otimizar as operações do banco de dados. Esta seção descreve o processo detalhado de migração de índices do MongoDB para o Amazon DocumentDB e, ao mesmo tempo, garante compatibilidade e eficiência.

Uso da ferramenta de índice do Amazon DocumentDB

Clone a ferramenta de indexação em GitHub

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

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

Exportação de índices do MongoDB (se estiver migrando a partir do MongoDB)

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

Verificação de índices

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

Importação de índices

```
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'
```

Migração do usuário

A migração de usuários do MongoDB para o Amazon DocumentDB é essencial para manter o controle de acesso, a autenticação e a segurança do banco de dados. Esta seção descreve as etapas detalhadas para uma migração com êxito dos usuários do MongoDB, preservando seus perfis e permissões, por meio da ferramenta de exportação de usuários do Amazon DocumentDB.

Uso da ferramenta de exportação de usuários do Amazon DocumentDB

Ele `Export Users tool` exporta usuários e funções do MongoDB ou do Amazon DocumentDB para JavaScript arquivos, que podem então ser usados para recriá-los em outro cluster.

Pré-requisitos

```
# 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
```

Etapa 1: exportação de usuários e perfis

```
# 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"
```

Etapa 2: edição do arquivo do usuário

As senhas não são exportadas por motivos de segurança. Abra o `mongodb-users.js` arquivo e adicione uma senha para cada usuário substituindo o pwd valor em cada `createUser` instrução:

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

Etapa 3: restauração de perfis personalizados no Amazon DocumentDB

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

Etapa 4: restauração de usuários no 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
```

Observações importantes

- As senhas não são exportadas por motivos de segurança e deverão ser adicionadas manualmente ao arquivo users.js.
- Os perfis devem ser importados antes dos usuários para garantir a atribuição adequada dos perfis.
- A ferramenta gera JavaScript arquivos que podem ser executados diretamente com o shell mongosh.
- Os perfis personalizados e seus privilégios são preservados durante a migração.
- Essa abordagem permite revisar e modificar as permissões do usuário antes da importação.

Esse método fornece uma abordagem segura e flexível para migrar usuários e perfis do MongoDB para o Amazon DocumentDB, permitindo redefinições de senha durante o processo de migração.

Migrações de dados

Tópicos

- migração online
- Migração offline

migração online

Esta seção fornece etapas detalhadas para realizar uma migração on-line do MongoDB para o Amazon DocumentDB usando para permitir o mínimo de tempo de inatividade e AWS DMS a replicação contínua. Para começar, você configura um cluster do Amazon DocumentDB como destino e garante que sua instância do MongoDB esteja configurada corretamente como origem, normalmente exigindo o modo de conjunto de réplicas para a captura de dados de alteração. Em seguida, você cria uma instância de replicação do DMS e define os endpoints de origem e de destino com os detalhes de conexão necessários. Depois de validar os endpoints, você configura e inicia uma tarefa de migração que pode incluir carga total de dados, replicação contínua ou ambas.

Criar novo cluster de destino

Consulte Criar novo cluster de destino.

Configuração da fonte

O MongoDB e o Amazon DocumentDB podem servir como origens de migração, dependendo do seu cenário:

- MongoDB como fonte — Comum ao migrar de um MongoDB local ou autogerenciado para um Amazon DocumentDB ou outros serviços de banco de dados. AWS Requer execução no modo de conjunto de réplicas com um oplog de tamanho adequado (certifique-se de que seja dimensionado para conter todas as operações durante a carga total) para oferecer suporte à captura de dados de alterações durante a migração.
- Amazon DocumentDB como origem — normalmente usado para replicação entre regiões, atualizações de versão ou migração para outros serviços de banco de dados, como o MongoDB Atlas. Requer Ativar fluxos de alterações a partir da configuração do parâmetro `change_stream_log_retention_duration` no grupo de parâmetros do cluster para capturar as alterações em andamento durante a migração. Certifique-se de que sua configuração `change_stream_log_retention_duration` seja grande o suficiente para cobrir o tempo necessário para concluir a carga completa.

Antes de iniciar a migração, configure sua fonte para permitir o AWS DMS acesso.

Crie um usuário do MongoDB com as permissões adequadas:

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

Configure a rede e a autenticação.

Ao configurar a conectividade de rede para a migração do MongoDB para o DMS:

EC2-hosted Origem do MongoDB

- Modifique o grupo de segurança do EC2 para permitir o tráfego de entrada do grupo de segurança da instância de replicação do DMS.

- Adicione uma regra para a porta 27017 de TCP (ou sua porta personalizada do MongoDB).
- Use o ID do grupo de segurança da instância de replicação do DMS como fonte para um controle de acesso preciso.
- Certifique-se de que a sub-rede da instância do EC2 tenha uma rota para a sub-rede da instância de replicação do DMS.

On-premises Origem do MongoDB

- Configure seu firewall para permitir conexões de entrada dos endereços IP públicos da instância de replicação do DMS.
- Se estiver usando Direct Connect uma VPN, garanta o roteamento adequado entre sua rede e a VPC que contém a instância DMS.
- Teste a conectividade usando comandos telnet ou nc da sub-rede do DMS para seu servidor do MongoDB.

Origem no MongoDB Atlas

- Adicione os endereços IP de uma instância de replicação do DMS à lista de permissões de IP do MongoDB Atlas.
- Configure o emparelhamento de VPC entre AWS VPC e MongoDB Atlas VPC se o Atlas estiver em execução. AWS
- Configure AWS PrivateLink para conectividade privada (nível Enterprise), se estiver sendo executado em outro provedor de nuvem.
- Crie um usuário dedicado com read/write as permissões apropriadas.
- Use uma string de conexão do MongoDB Atlas com o modo de SSL definido como “verify-full”.
- Garanta um tamanho de oplog suficiente para a duração da migração.

Origem no Amazon DocumentDB

Configure seu grupo de segurança do Amazon DocumentDB de origem para permitir o tráfego de entrada do grupo de segurança da instância de replicação do DMS.

Criação de uma instância de replicação do DMS.

Use o DMS Buddy on GitHub para criar uma infraestrutura de migração ideal com configurações e tamanhos de instância de DMS ideais. Se preferir configurar manualmente, siga estas etapas:

1. No AWS DMS console, no painel de navegação, escolha Migrar ou replicar e, em seguida, escolha Instâncias provisionadas.
2. Insira os detalhes da instância de replicação:
   - Nome da instância: escolha um nome exclusivo.
   - Classe da instância: selecione com base na workload. Exemplo: dms.r7i.large (cargas de trabalho pequenas), dms.r7i.4xlarge (cargas de trabalho grandes).
   - Versão do motor: 3.5.4
   - Armazenamento alocado: o padrão é 50 GB (aumente se necessário). Isso é determinado pelo tamanho do documento updates/second e pela duração total do carregamento.
   - Multi-AZ Implantação: habilite a alta disponibilidade, se necessário.
   - Escolha a mesma VPC do Amazon DocumentDB.
   - Garanta que os grupos de segurança permitam tráfego de entrada da origem e do Amazon DocumentDB.
3. Escolha Criar instância de replicação e aguarde até que o status esteja disponível.

Criação de endpoints da VPC

Criação de um endpoint de origem

Para uma origem do MongoDB

1. No console do DMS, no painel de navegação, escolha Migrar ou replicar e, em seguida, escolha Endpoints.
2. Escolha Criar endpoint.
3. Na página Criar endpoint, escolha Endpoint de origem.
4. Na seção Configuração do endpoint:
   - Insira um identificador de endpoint exclusivo e significativo (por exemplo, “mongodb-source”).
   - Escolha o MongoDB como Mecanismo de origem.
   - Em Acesso ao banco de dados de endpoint, escolha Fornecer informações de acesso manualmente.
   - Em Nome do servidor, insira seu*MongoDB server DNS name/IP address*.
   - Em Porta, insira 27017 (porta padrão do MongoDB).

- Para o modo de autenticação, escolha o modo apropriado para seu aplicativo (password/SSL) (o padrão é gerenciador de segredos).
- Se o Modo de autenticação for Senha, forneça:
  - Nome de usuário e Senha: insira as credenciais do MongoDB.
  - Nome do banco de dados: o nome do seu banco de dados de origem.
  - Mecanismo de autenticação: SCRAM-SHA-1 (padrão) ou mecanismo apropriado

5. Em Modo de metadados, deixe a configuração padrão do documento.
6. Atributos adicionais de conexão:

- authSource=admin (se o banco de dados de autenticação for diferente)
- replicaSet=<nome-do-seu-conjunto-de-réplicas> (necessário para o CDC)

Para uma origem do Amazon DocumentDB

1. No console do DMS, no painel de navegação, escolha Migrar ou replicar e, em seguida, escolha Endpoints.
2. Escolha Criar endpoint.
3. Na página Criar endpoint, escolha Endpoint de origem.
4. Na seção Configuração do endpoint:

- Insira um identificador de endpoint exclusivo e significativo (por exemplo, “docdb-source”).
- Escolha Amazon DocumentDB como Mecanismo de origem.
- Em Acesso ao banco de dados de endpoint, escolha Fornecer informações de acesso manualmente.
- Em Nome do servidor, insira seu*source Amazon DocumentDB cluster endpoint*.
- Em Porta, insira 27017 (porta padrão para o Amazon DocumentDB).
- Em Modo de SSL, escolha verify-full (recomendado para Amazon DocumentDB).
- Em Certificado CA, escolha o certificado CA raiz do Amazon DocumentDB,
- Para o modo de autenticação, escolha o modo apropriado para seu aplicativo (password/SSL) (o padrão é gerenciador de segredos).
- Se o Modo de autenticação for Senha, forneça:
  - Nome de usuário e senha: insira as credenciais do Amazon DocumentDB.
  - Nome do banco de dados: o nome do seu banco de dados de origem.

- Mecanismo de autenticação: SCRAM-SHA-1 (padrão) ou mecanismo apropriado

5. Em Modo de metadados, deixe a configuração padrão do documento.

Criação de um endpoint de destino do Amazon DocumentDB.

1. No console do DMS, no painel de navegação, escolha Migrar ou replicar e, em seguida, escolha Endpoints.
2. Escolha Criar endpoint.
3. Na página Criar endpoint, escolha Endpoint de destino.
4. Na seção Configuração do endpoint:
   - Insira um identificador de endpoint exclusivo e significativo (por exemplo, “docdb-target”).
   - Escolha Amazon DocumentDB como Mecanismo de destino.
   - Para Acesso ao banco de dados do endpoint, escolha o método que você deseja usar para autenticar o acesso ao banco de dados:
     - Se você escolher AWS Secrets Manager, escolha o segredo em que você armazena suas credenciais do Amazon DocumentDB no campo Segredo.
     - Se você escolher Fornecer informações de acesso manualmente:
       - Em Nome do servidor, insira seu*target Amazon DocumentDB cluster endpoint*.
       - Em Porta, insira 27017 (porta padrão para o Amazon DocumentDB).
       - Em Modo de SSL, escolha verify-full (recomendado para Amazon DocumentDB).
       - Em Certificado CA, baixe e especifique o pacote de certificados CA para verificação de SSL.
       - Para o modo de autenticação, escolha o modo apropriado para seu aplicativo (password/SSL) (o padrão é gerenciador de segredos).
       - Se o Modo de autenticação for Senha, forneça:
         - Nome de usuário e senha: insira as credenciais do Amazon DocumentDB.
         - Nome do banco de dados: o nome do seu banco de dados de origem.
         - Mecanismo de autenticação: SCRAM-SHA-1 (padrão) ou mecanismo apropriado
5. Em Modo de metadados, deixe a configuração padrão do documento.

Criação de tarefas de replicação

1. No console do DMS, no painel de navegação, escolha Migrar ou replicar e, em seguida, escolha Tarefas.
2. Escolha Criar tarefa.
3. Na página Criar tarefa, na seção Configuração da tarefa:
   - Insira um Identificador de tarefa exclusivo e significativo (por exemplo, “mongodb-docdb-replication”).
   - Escolha o endpoint de origem que você criou anteriormente no menu suspenso Endpoint de banco de dados de origem.
   - Escolha o endpoint de destino que você criou anteriormente no menu suspenso Endpoint de banco de dados de destino.
   - Escolha a instância de replicação que você criou anteriormente no menu suspenso Instância provisionada.
   - Em Tipo de tarefa, escolha Migrar e replicar.
   - Por quanto tempo você pretende continuar replicando da origem para o destino? , escolha Por tempo limitado.
4. Na seção Configurações:
   - No modo de preparação da tabela Target, escolha Não fazer nada. Caso contrário, as coleções e índices que já foram criados serão excluídos.
   - Para registros de tarefas, marque a caixa Ativar CloudWatch registros.
   - Deixe os valores padrão para todas as outras configurações.
5. Na parte superior da seção Configurações, no modo de edição, escolha o editor JSON e defina os seguintes atributos:

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

Defina `MaxFullLoadSubTasks` com base na contagem de coleções e na capacidade da instância. Use valores mais baixos (2—4) para coleções menores ou menores e valores mais altos (8—16) para muitas coleções grandes. Para dimensionamento automatizado, consulte DMS Buddy on. GitHub

6. Na seção Mapeamentos de tabela, adicione uma nova regra de seleção:
   - Em Nome do esquema, adicione o banco de dados de origem a ser migrado. Use % para especificar vários bancos de dados.
   - No nome da Tabela do esquema, adicione o banco de dados de origem a ser migrado. Use % para especificar várias coleções.
   - Em Ação, deixe a configuração padrão de Incluir
7. Para coleções grandes (acima de 100 GB), adicione a Regra de configurações de tabela:
   - Em Nome do esquema, adicione o banco de dados de origem a ser migrado. Use % para especificar vários bancos de dados.
   - No nome da Tabela do esquema, adicione o banco de dados de origem a ser migrado. Use % para especificar várias coleções.
   - Para Número de partições, defina com base na sua carga de trabalho. Para um pequeno número de coleções grandes, defina partições ≥ `MaxFullLoadSubTasks` para manter todos os trabalhadores ocupados. Para muitas coleções carregadas em paralelo, defina partições ≤, `MaxFullLoadSubTasks` pois a própria contagem de coleções fornece paralelismo suficiente.
8. Na seção Avaliação de pré-migração, certifique-se de que ela esteja desativada. A avaliação de pré-migração não é suportada pelo Amazon DocumentDB como destino e deve ser desativada.

Migração offline

Esta seção descreve o processo para realizar uma migração offline de uma instância autogerenciada do MongoDB para o Amazon DocumentDB usando ferramentas nativas do MongoDB: `mongodump` e `mongorestore`.

Pré-requisitos

Requisitos do MongoDB de origem

- Acesso à instância de origem do MongoDB com as permissões apropriadas.
- Instale o `mongodump`, se necessário (ele é instalado durante a instalação do MongoDB).

- Verifique se há espaço em disco suficiente para os arquivos de despejo.

Requisitos do Amazon DocumentDB de destino

- Certifique-se de ter um cluster do Amazon DocumentDB provisionado.
- Certifique-se de que haja uma instância do EC2 na mesma VPC do Amazon DocumentDB para facilitar a migração.
- A conectividade de rede deve estar disponível entre seu ambiente de origem e o Amazon DocumentDB.
- O mongorestore deve estar instalado na instância EC2 de migração.
- As permissões apropriadas do IAM devem ser configuradas para acessar o Amazon DocumentDB,

Requisitos gerais

- AWS CLI deve ser configurado (se estiver usando AWS serviços para armazenamento intermediário)
- Uma largura de banda suficiente deve estar disponível para a transferência de dados.
- A janela de tempo de inatividade deve ser aprovada (se estiver fazendo uma migração ao vivo, considere outras abordagens)

Criação de um cluster do Amazon DocumentDB

Consulte Criar novo cluster de destino.

Execute o despejo de dados (mongodump)

Escolha uma das opções a seguir para criar um arquivo de despejo:

- Opção 1: básica

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

- Opção 2: melhor controle e performance

  Comprime a saída com gzip, despeja as coleções em paralelo, ativa o TLS e lê a partir de uma réplica secundária, se disponível:

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

- Opção 3: bancos de dados grandes

  Despeja um banco de dados e uma coleção específicos, filtra documentos por consulta e envia para um único arquivo compactado:

```
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
```

Transferência dos arquivos de despejo para o ambiente de restauração

Escolha um método apropriado com base no tamanho do seu despejo:

- Pequeno: copie diretamente para sua máquina de migração (a instância EC2 que você criou anteriormente):

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

- Médio: use o Amazon S3 como armazenamento intermediário:

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

- Grande — Para bancos de dados muito grandes, considere AWS DataSync uma transferência física.

Restauração dos dados no Amazon DocumentDB (mongorestore)

Antes de iniciar o processo de restauração, crie os índices no Amazon DocumentDB. É possível utilizar a ferramenta de índice do Amazon DocumentDB para exportar e importar índices.

Escolha uma das opções a seguir para restaurar os dados:

- Opção 1: restauração básica

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

- Opção 2: melhor controle e performance

  Restaura despejos compactados com gzip com coleta paralela e inserção de documentos. Ignora a restauração do índice, pois os índices são pré-criados:

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

- Opção 3: bancos de dados grandes ou controles específicos

  Restaura a partir de um arquivo compactado com filtragem de namespace para incluir ou excluir bancos de dados e coleções específicos. Ignora a restauração do índice, pois os índices são pré-criados:

```
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
```

Monitoramento

Esta seção fornece um processo de monitoramento detalhado para acompanhar o progresso, o desempenho e a integridade de uma migração contínua.

As etapas de monitoramento se aplicam independentemente do método de migração (AWS DMS mongodump/mongorestore, ou de outras ferramentas).

AWS DMS Monitoramento da migração (se aplicável)

Monitore as seguintes CloudWatch métricas principais:

Métricas da fase de carga total

- FullLoadThroughputBandwidthTarget— Largura de banda de rede (KB/second) durante a carga total
- FullLoadThroughputRowsTarget— Número de rows/documents carregamentos por segundo
- FullLoadThroughputTablesTarget— Número de tables/collections concluídos por minuto
- FullLoadProgressPercent— Porcentagem de carga completa concluída
- TablesLoaded— Número de carregados tables/collections com sucesso
- TablesLoading— Número de tables/collections carregamentos atuais
- TablesQueued— Número de tables/collections espera para serem carregados
- TablesErrored— Número do tables/collections que falhou ao carregar

Métricas da fase do CDC

- CDCLatencyTarget— Tempo de atraso (segundos) entre a alteração da fonte e o aplicativo de destino

- CDCLatencySource— Tempo de atraso (segundos) entre a alteração na fonte e a leitura do DMS
- CDCThroughputRowsTarget— Linhas por segundo aplicadas durante a replicação contínua
- CDCThroughputBandwidthTarget— Largura de banda de rede (KB/second) durante o CDC
- CDCIncomingChanges— Número de eventos de mudança recebidos da fonte
- CDCChangesMemoryTarget— Memória usada (MB) para armazenar alterações no lado do alvo

Métricas de recursos

- CPUUtilization: uso da CPU da instância de replicação
- FreeableMemory— Memória disponível na instância de replicação
- FreeStorageSpace— Armazenamento disponível na instância de replicação
- NetworkTransmitThroughput— Taxa de transferência de rede para a instância de replicação
- NetworkReceiveThroughput— Taxa de transferência de rede para a instância de replicação

Métricas de erros

- ErrorsCount— Número total de erros durante a migração
- TableErrorsCount— Número de erros específicos da tabela
- RecordsErrorsCount— Número de erros específicos do registro

Crie CloudWatch alarmes para métricas críticas, como `CDCLatencyTarget` e `CPUUtilization` para receber notificações se o desempenho da migração diminuir.

Registros do DMS (CloudWatch registros)

1. Acesse o console do Amazon CloudWatch Logs.
2. Encontre e escolha em seu grupo de logs. Ele se parece com “dms-tasks –”.
3. Procure por fluxos de log que possam conter informações de erro:
    - Fluxos com “error” no nome
    - Fluxos com IDs de tarefas ou nomes de endpoints
    - Os fluxos de log mais recentes durante o período de sua migração
4. Dentro desses fluxos, pesquise por palavras-chave como:

- "error"
- "exception"
- "failed"
- "warning"

Status da tarefa do DMS (usando AWS CLI)

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

Fluxo de status esperado:

em criação → pronto → em execução → parando → parado (ou com falha

Monitore usando **`docdb-dashboarder`**

A `docdb-dashboarder` ferramenta fornece monitoramento abrangente para clusters do Amazon DocumentDB ao gerar automaticamente CloudWatch painéis com métricas de desempenho essenciais. Esses painéis exibem métricas críticas em nível de cluster (atraso de réplica, contadores de operação), métricas em nível de instância (CPU, memória, conexões) e métricas de armazenamento (uso de volume, armazenamento de backup). Para cenários de migração, a ferramenta oferece painéis especializados que monitoram o andamento da migração com métricas como atraso na replicação do CDC e taxas de operação. Os painéis podem monitorar vários clusters simultaneamente e incluir suporte para NVMe-backed instâncias. Ao visualizar essas métricas, as equipes podem identificar proativamente gargalos de performance, otimizar a alocação de recursos e garantir o bom funcionamento de suas implantações do Amazon DocumentDB. A ferramenta elimina a necessidade de criação manual de painéis e, ao mesmo tempo, fornece monitoramento consistente em todos os ambientes. Para obter instruções e opções avançadas de configuração, consulte o repositório Amazon DocumentDB Dashboarder Tool. GitHub

Validação

Tópicos

- Lista de verificação de validação
- Validação de esquema e índices

- Amostragem de dados e validação em nível de campo
- Validação usando a DataDiffer ferramenta

Esta seção fornece um processo de validação detalhado para garantir a consistência dos dados, a integridade e a compatibilidade dos aplicativos após a migração.

As etapas de validação se aplicam independentemente do método de migração (AWS DMS mongodump/mongorestore, ou de outras ferramentas).

Lista de verificação de validação

Verifique se o número de documentos em cada coleção corresponde entre a origem e o destino:

Origem do MongoDB

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

Destino do Amazon DocumentDB

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

Validação de esquema e índices

Verifique se:

- todas as coleções existem no destino.
- os índices estão replicados corretamente.
- as definições do esquema (se aplicadas) são idênticas.

Verifique as coleções (origem versus 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();"
```

Verifique os índices (origem versus 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 a lista de coleções para garantir que não haja coleções ausentes ou extras.

Verifique os índices verificando os nomes dos índices, as definições de chave, as restrições exclusivas e os índices de TTL (se houver).

Verifique as regras de validação do esquema (se estiver usando a validação de esquema do 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"
```

Amostragem de dados e validação em nível de campo

É possível amostrar documentos aleatoriamente e comparar campos entre origem e destino.

Amostragem manual

Obtenha cinco documentos aleatórios (origem):

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

Obtenha os mesmos IDs 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>] } })"
```

Amostragem automática

```
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']}")
```

Validação usando a DataDiffer ferramenta

A DataDiffer ferramenta on GitHub fornece uma maneira confiável de comparar dados entre bancos de dados de origem e de destino.

Pré-requisitos

Os seguintes pré-requisitos devem ser atendidos antes da instalação da DataDiffer ferramenta:

- Python3,7+
- PyMongo biblioteca
- Conectividade de rede com os clusters de origem do MongoDB e de destino do Amazon DocumentDB

Configuração e instalação

Clone o repositório e navegue até o diretório DataDiffer

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

Instale as dependências necessárias

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

Execução da validação de dados

Crie um arquivo de configuração (por exemplo, config.json) com detalhes da conexão

```
{
  "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
  }
}
```

Execute a DataDiffer ferramenta

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

Para coleções grandes, use a amostragem para validar um subconjunto de dados

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

Para validar várias coleções, crie arquivos de configuração separados ou use o modo de lote

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

Como interpretar os resultados do

A saída da ferramenta será:

- Total de documentos na origem e no destino
- Número de documentos correspondentes
- Número de documentos ausentes
- Número de documentos com diferenças

- Relatório detalhado das diferenças (se houver)

Práticas recomendadas

A seguir estão as melhores práticas ao usar a DataDiffer ferramenta:

- Execute em fases: primeiro valide a contagem de documentos, depois experimente os principais documentos e, por fim, faça uma comparação completa, se necessário.
- Verifique as diferenças de esquema: o Amazon DocumentDB tem algumas limitações em comparação com o MongoDB. A ferramenta destacará tipos ou estruturas de dados incompatíveis.
- Valide durante períodos de silêncio: execute a validação quando as operações de gravação forem mínimas para garantir a consistência.
- Monitore o uso de recursos: o processo de comparação pode consumir muitos recursos. Ajuste o tamanho dos lotes e a contagem de linhas adequadamente.
- Valide os índices: após a validação dos dados, certifique-se de que todos os índices necessários tenham sido criados no cluster de destino do Amazon DocumentDB.
- Resultados da validação de documentos: mantenha um registro dos resultados da validação de cada coleção como parte da documentação de migração.

Migração do Couchbase Server

Tópicos

- Introdução
- Comparação com o Amazon DocumentDB
- Descoberta
- Planejamento
- Migração
- Validação

Introdução

Este guia apresenta os principais pontos a serem considerados ao migrar do Couchbase Server para o Amazon DocumentDB. Ele explica as considerações sobre as fases de descoberta, planejamento, execução e validação de sua migração. Também explica como realizar migrações offline e online.

Comparação com o Amazon DocumentDB

	Servidor Couchbase	Amazon DocumentDB
Organização de dados	Nas versões 7.0 e posteriores, os dados são organizados em intervalos, escopos e coleções. Nas versões anteriores, os dados são organizados em compartim entos.	Os dados são organizados em bancos de dados e coleções.
Compatibilidade	Há APIs separadas para cada serviço (por exemplo, dados, índice, pesquisa etc.). As pesquisas secundárias usam SQL++ (anteriormente conhecido como N1QL); uma linguagem de consulta baseada em ANSI-stan dard SQL, por isso é familiar para muitos desenvolvedores.	O Amazon DocumentDB é compatível com a API do MongoDB.
Arquitetura	O armazenamento é anexado a cada instância do cluster. Você não pode escalar a computaçã o independentemente do armazenamento.	O Amazon DocumentDB foi projetado para a nuvem e para evitar as limitações das arquiteturas tradicionais de banco de dados. As camadas de computação e armazenam ento são separadas no Amazon DocumentDB e a camada de computação pode ser escalada independentemente do armazenamento.
Adicione capacidad e de leitura sob demanda	Os clusters podem ser ampliados com a adição de instâncias. Como o armazenamento está conectado à instância em que	Você pode obter escalabil idade de leitura para seu cluster Amazon DocumentDB criando até 15 réplicas do Amazon

	Servidor Couchbase	Amazon DocumentDB
	o serviço está em execução, o tempo necessário para escalar depende da quantidade de dados que precisam ser movidos para a nova instância ou rebalanceados.	DocumentDB no cluster. Não há impacto na camada de armazenamento.
Recupere-se rapidamente da falha do nó	Os clusters têm recursos de failover automático, mas o tempo necessário para recuperar a força total do cluster depende da quantidade de dados que precisam ser movidos para a nova instância.	O Amazon DocumentDB pode fazer o failover do primário normalmente em 30 segundos e restaurar a força total do cluster em 8 a 10 minutos, independentemente da quantidade de dados no cluster.
Dimensione o armazenamento à medida que os dados crescem	Para clusters autogerenciados, o armazenamento e o iOS não são escalados automaticamente.	O armazenamento do Amazon DocumentDB e o iOS escalam automaticamente.
Faça backup de dados sem afetar o desempenho	Os backups são executados pelo serviço de backup e não são ativados por padrão. Como o armazenamento e a computação não estão separados, pode haver um impacto no desempenho.	Os backups do Amazon DocumentDB são habilitados por padrão e não podem ser desativados. Os backups são gerenciados pela camada de armazenamento, portanto, não causam impacto na camada de computação. O Amazon DocumentDB oferece suporte à restauração a partir de um snapshot de cluster e à restauração em um ponto no tempo.

	Servidor Couchbase	Amazon DocumentDB
Durabilidade dos dados	Pode haver no máximo 3 cópias de réplica de dados em um cluster, totalizando 4 cópias. Cada instância em que o serviço de dados estiver em execução terá 1, 2 ou 3 cópias ativas dos dados.	O Amazon DocumentDB mantém 6 cópias de dados, independentemente de quantas instâncias de computação existam, com um quórum de gravação de 4 e persistem como verdadeiras. Os clientes recebem uma confirmação após a camada de armazenamento persistir em 4 cópias dos dados.
Consistência	A consistência imediata K/V das operações é suportada. O SDK do Couchbase encaminha as K/V solicitações para a instância específica que contém a cópia ativa dos dados. Assim que uma atualização for confirmada, o cliente terá a garantia de lê-la. A replicação de atualizações para outros serviços (indexação, pesquisa, análise, eventos) acaba sendo consistente.	Eventualmente, as réplicas do Amazon DocumentDB são consistentes. Se forem necessárias leituras de consistência imediata, o cliente poderá ler da instância primária.
Replicação	Cross-Data O Center Replication (XDCR) fornece replicação filtrada e ativa de dados passive/active em muitas topologias.	Os clusters globais do Amazon DocumentDB fornecem replicação ativa-passiva em topologias 1:muitas (até 10).

Descoberta

A migração para o Amazon DocumentDB exige uma compreensão completa da carga de trabalho existente do banco de dados. A descoberta da carga de trabalho é o processo de analisar a

configuração e as características operacionais do cluster Couchbase — conjunto de dados, índices e carga de trabalho — para ajudar a garantir uma transição perfeita com o mínimo de interrupção.

Configuração do cluster

O Couchbase usa uma arquitetura centrada no serviço em que cada recurso corresponde a um serviço. Execute o seguinte comando em seu cluster Couchbase para determinar quais serviços estão sendo usados (consulte Obtenção de informações sobre nós):

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

Exemplo de saída:

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

Os serviços do Couchbase incluem o seguinte:

Serviço de dados (kv)

O serviço de dados fornece read/write acesso aos dados na memória e no disco.

O Amazon DocumentDB oferece suporte a K/V operações em dados JSON por meio da API MongoDB.

Serviço de consulta (n1ql)

O serviço de consulta oferece suporte à consulta de dados JSON via SQL++.

O Amazon DocumentDB oferece suporte à consulta de dados JSON por meio da API do MongoDB.

Serviço de indexação (índice)

O serviço de indexação cria e mantém índices nos dados, permitindo consultas mais rápidas.

O Amazon DocumentDB oferece suporte a um índice primário padrão e à criação de índices secundários em dados JSON por meio da API do MongoDB.

Serviço de pesquisa (fts)

O serviço de pesquisa oferece suporte à criação de índices para pesquisa de texto completo.

O recurso nativo de pesquisa de texto completo do Amazon DocumentDB permite que você realize consultas de texto básicas em grandes conjuntos de dados textuais usando índices de texto para fins especiais usando a API. MongoDB Para recursos avançados, como pesquisa difusa, pesquisa multilíngue e pesquisa entre coleções, use a integração Zero-ETL do Amazon DocumentDB com o Amazon Service. OpenSearch

Serviço de análise (cbas)

O serviço de análise oferece suporte à análise de dados JSON quase em tempo real.

O Amazon DocumentDB oferece suporte a consultas ad-hoc em dados JSON por meio da API do MongoDB. Você também pode executar consultas complexas em seus dados JSON no Amazon DocumentDB usando o Apache Spark em execução no Amazon EMR.

Serviço de eventos (eventos)

O serviço de eventos executa a lógica de negócios definida pelo usuário em resposta às mudanças nos dados.

O Amazon DocumentDB automatiza cargas de trabalho orientadas por eventos invocando AWS Lambda funções sempre que os dados são alterados com seu cluster Amazon DocumentDB.

Serviço de backup (backup)

O serviço de backup agenda backups de dados completos e incrementais e mescla de backups de dados anteriores.

O Amazon DocumentDB faz backup contínuo de seus dados no Amazon S3 com um período de retenção de 1 a 35 dias para que você possa restaurar rapidamente em qualquer ponto dentro do período de retenção de backup. O Amazon DocumentDB também tira snapshots automáticos dos seus dados como parte desse processo de backup contínuo. Você também pode gerenciar o backup e a restauração do Amazon DocumentDB com AWS Backup. .

Características operacionais

Use a ferramenta Discovery para Couchbase para obter as seguintes informações sobre seu conjunto de dados, índices e carga de trabalho. Essas informações ajudarão você a dimensionar seu cluster Amazon DocumentDB.

Conjunto de dados

A ferramenta recupera as seguintes informações de bucket, escopo e coleta:

1. nome do bucket
2. tipo de bucket
3. nome do escopo
4. nome da coleção
5. tamanho total (bytes)
6. total de itens
7. tamanho do item (bytes)

Índices

A ferramenta recupera as seguintes estatísticas de índice e todas as definições de índice para todos os compartimentos. Observe que os índices primários são excluídos, pois o Amazon DocumentDB cria automaticamente um índice primário para cada coleção.

1. nome do bucket
2. nome do escopo
3. nome da coleção
4. nome do índice
5. tamanho do índice (bytes)

Workload

A ferramenta K/V recupera as métricas de consulta N1QL. K/V os valores das métricas são coletados no nível do bucket e as métricas do SQL++ são coletadas no nível do cluster.

As opções da linha de comando da ferramenta são as seguintes:

```
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>
```

Veja a seguir um exemplo de comando:

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

K/V os valores métricos serão baseados em amostras a cada 10 minutos na semana passada (consulte Método HTTP e URI). Os valores métricos do SQL++ serão baseados em amostras a cada 1 segundo nos últimos 60 segundos (consulte Rótulos https://docs.couchbase.com/server/current/rest-api/rest-statistics-single.html#general-labels gerais). A saída do comando estará nos seguintes arquivos:

collection-stats.csv — informações de bucket, escopo e coleta

```
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 — nomes e tamanhos 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 — obtenha, defina e exclua métricas para todos os 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 — Métricas de seleção, exclusão e inserção do SQL ++ para o cluster

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

indexes- <bucket-name>.txt — definições de índice de todos os índices no bucket. Observe que os índices primários são excluídos, pois o Amazon DocumentDB cria automaticamente um índice primário para cada coleção.

```
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`)
...
```

Planejamento

Na fase de planejamento, você determinará os requisitos de cluster do Amazon DocumentDB e o mapeamento dos buckets, escopos e coleções do Couchbase para bancos de dados e coleções do Amazon DocumentDB.

Requisitos de cluster do Amazon DocumentDB

Use os dados coletados na fase de descoberta para dimensionar seu cluster Amazon DocumentDB. Consulte Dimensionamento de instâncias para obter mais informações sobre como dimensionar seu cluster Amazon DocumentDB.

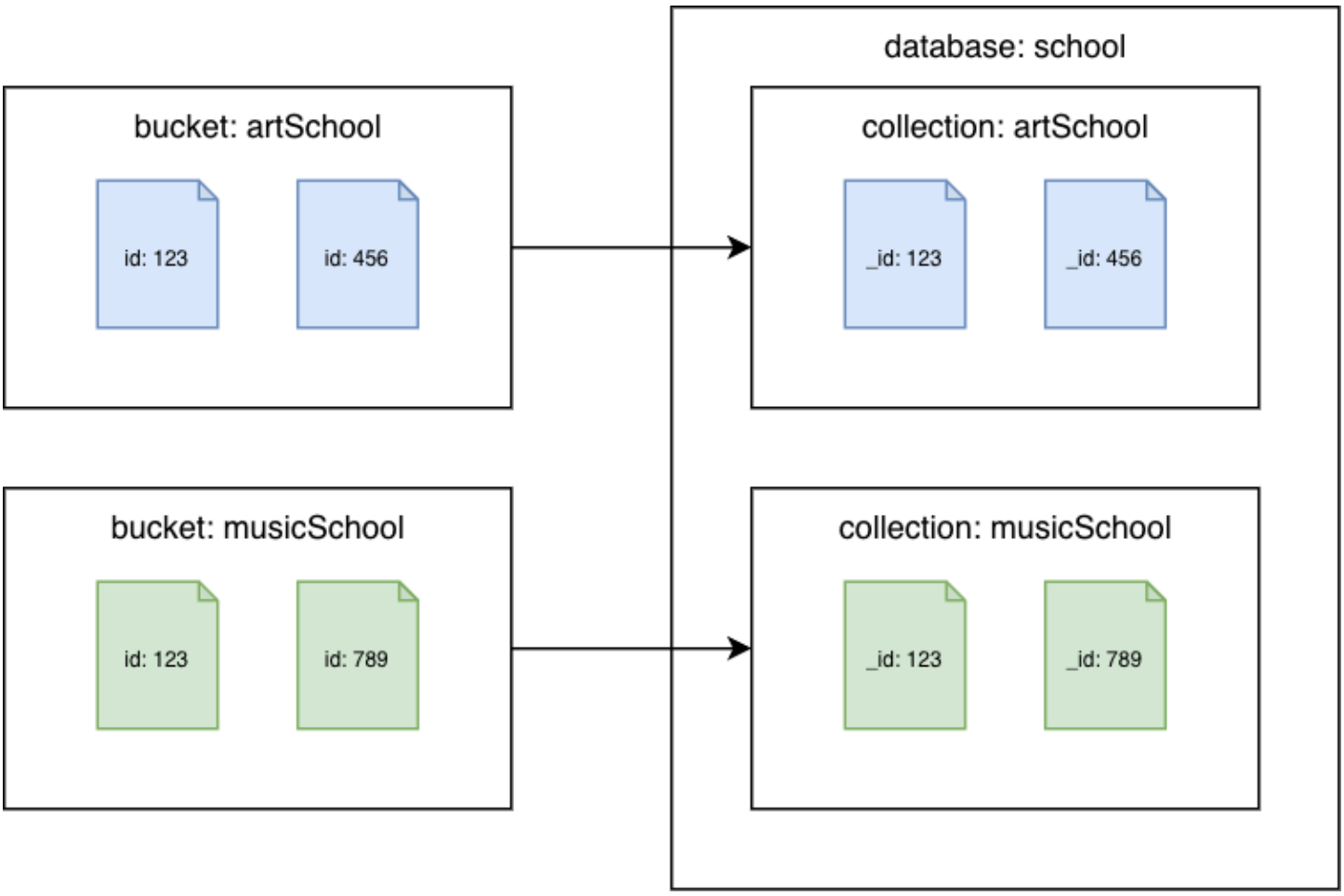
Mapeando intervalos, escopos e coleções para bancos de dados e coleções

Determine os bancos de dados e as coleções que existirão em seus clusters do Amazon DocumentDB. Considere as seguintes opções, dependendo de como os dados são organizados em seu cluster Couchbase. Essas não são as únicas opções, mas fornecem pontos de partida para você considerar.

Couchbase Server 6.x ou anterior

Buckets do Couchbase para coleções do Amazon DocumentDB

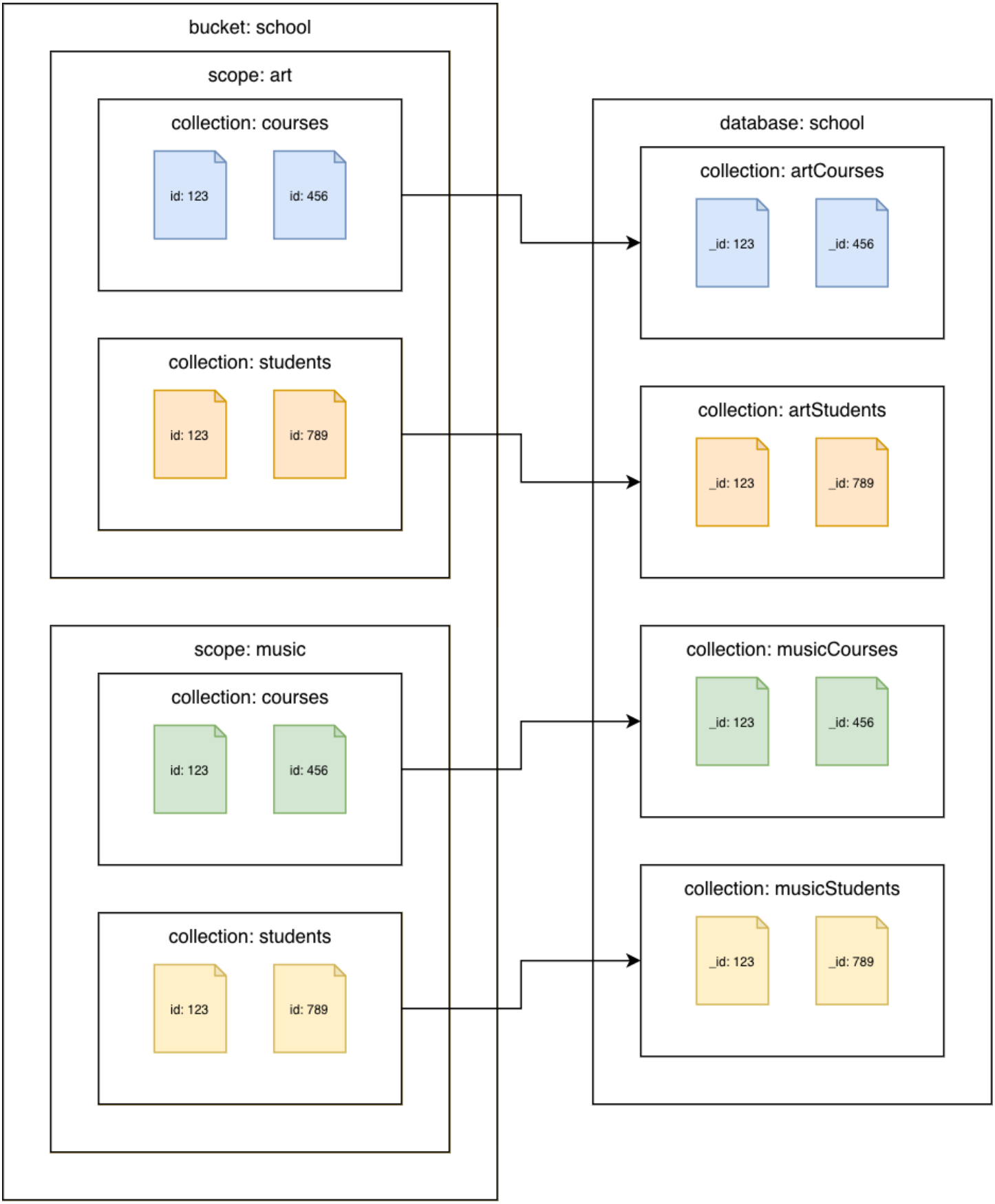
Migre cada bucket para uma coleção diferente do Amazon DocumentDB. Nesse cenário, o valor do documento Couchbase será usado como o `id` valor do Amazon `_id` DocumentDB.



Couchbase Server 7.0 ou posterior

Coleções do Couchbase para coleções do Amazon DocumentDB

Migre cada coleção para uma coleção diferente do Amazon DocumentDB. Nesse cenário, o valor do documento Couchbase será usado como o `id` valor do Amazon `_id` DocumentDB.


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

Migração

Migração de índices

A migração para o Amazon DocumentDB envolve a transferência não apenas de dados, mas também de índices para manter o desempenho das consultas e otimizar as operações do banco de dados. Esta seção descreve o processo detalhado passo a passo para migrar índices para o Amazon DocumentDB e, ao mesmo tempo, garantir a compatibilidade e a eficiência.

Use o Amazon Q (disponível no AWS Management Console ou no seu IDE) para converter instruções SQL++ em `CREATE INDEX` comandos do Amazon `createIndex()` DocumentDB.

1. Faça o upload do (s) <bucket name>arquivo (s) indexes- .txt criados pela Discovery Tool for Couchbase.
2. Digite o seguinte prompt:

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

O Amazon Q gerará `createIndex()` comandos equivalentes do Amazon DocumentDB. Observe que talvez seja necessário atualizar os nomes das coleções com base em como você mapeou os buckets, escopos e coleções do Couchbase para as coleções do Amazon DocumentDB.

Por exemplo:

indexes-beer-sample.txt

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

Exemplo de saída do Amazon Q (trecho):

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

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

Para quaisquer índices que o Amazon Q não consiga converter, consulte Gerenciando índices e índices do Amazon DocumentDB e propriedades do índice para obter mais informações.

Refatore o código para usar as APIs do MongoDB

Os clientes usam os SDKs do Couchbase para se conectar ao Couchbase Server. Os clientes do Amazon DocumentDB usam drivers do MongoDB para se conectar ao Amazon DocumentDB. Todos os idiomas suportados pelos SDKs do Couchbase também são suportados pelos drivers do MongoDB. Consulte MongoDB Drivers para obter mais informações sobre o driver para seu idioma.

Como as APIs são diferentes entre o Couchbase Server e o Amazon DocumentDB, você precisará refatorar seu código para usar as APIs apropriadas do MongoDB. Você pode usar o Amazon Q para converter as chamadas de K/V API e as consultas SQL++ nas APIs equivalentes do MongoDB:

1. Faça upload do (s) arquivo (s) do código-fonte.
2. Digite o seguinte prompt:

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

Usando a amostra de código do Hello Couchbase Python, o Amazon Q gera o seguinte (trecho):

```
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 Conectando-se programaticamente ao Amazon DocumentDB para ver exemplos de conexão com o Amazon DocumentDB em Python, PHP, Go, Java Node.js, C#/.NET, R e Ruby.

Selecione a abordagem de migração

Ao migrar dados para o Amazon DocumentDB, há duas opções:

1. migração off-line
2. migração on-line

> **Note**
>
> O Amazon DocumentDB usa o `_id` campo como chave primária para cada documento. As abordagens de migração neste guia armazenam a chave do documento do Couchbase no `_id` campo. Para migração offline, isso é feito por meio da `--include-key _id` opção em`cbexport`. Para a migração on-line, isso é feito por meio do conector Kafka. `ProvidedInKeyStrategy`

Migração offline

Considere uma migração off-line quando:

- O tempo de inatividade é aceitável: a migração off-line envolve interromper as operações de gravação no banco de dados de origem, exportar os dados e importá-los para o Amazon DocumentDB. Esse processo gera tempo de inatividade para seu aplicativo. Se seu aplicativo ou carga de trabalho tolerar esse período de indisponibilidade, a migração off-line é uma opção viável.

- Migração de conjuntos de dados menores ou realização de provas de conceito: para conjuntos de dados menores, o tempo necessário para o processo de exportação e importação é relativamente curto, tornando a migração off-line um método rápido e simples. Também é adequado para ambientes de desenvolvimento, teste e prova de conceito, nos quais o tempo de inatividade é menos crítico.
- Simplicidade é uma prioridade: o método offline, usando cbexport e mongoimport, geralmente é a abordagem mais direta para migrar dados. Ele evita as complexidades da captura de dados de mudança (CDC) envolvida nos métodos de migração on-line.
- Nenhuma alteração contínua precisa ser replicada: se o banco de dados de origem não estiver recebendo mudanças ativamente durante a migração, ou se essas alterações não forem essenciais para serem capturadas e aplicadas ao destino durante o processo de migração, uma abordagem off-line é apropriada.

Tópicos

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

Couchbase Server 6.x ou anterior

Bucket Couchbase para a coleção Amazon DocumentDB

Exporte dados usando cbexport json para criar um despejo JSON de todos os dados no bucket. Para a `--format` opção, você pode usar `lines` ou`list`.

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

Importe os dados para uma coleção do Amazon DocumentDB usando mongoimport com a opção apropriada para importar as linhas ou a lista:

linhas:

```
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 ou posterior

Para realizar uma migração offline, use as ferramentas cbexport e mongoimport:

Bucket Couchbase com escopo padrão e coleção padrão

Exporte dados usando cbexport json para criar um dump JSON de todas as coleções no bucket. Para a `--format` opção, você pode usar `lines` ou `list`.

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

Importe os dados para uma coleção do Amazon DocumentDB usando mongoimport com a opção apropriada para importar as linhas ou a lista:

linhas:

```
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
```

Coleções do Couchbase para coleções do Amazon DocumentDB

Exporte dados usando cbexport json para criar um dump JSON de cada coleção. Use a `--include-data` opção de exportar cada coleção. Para a `--format` opção, você pode usar `lines` ou`list`. Use as `--collection-field` opções `--scope-field` e para armazenar o nome do escopo e da coleção nos campos especificados em 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 adicionou os `_collection` campos `_scope` e a cada documento exportado, você pode removê-los de cada documento no arquivo de exportação por meio de pesquisa e substituição`sed`, ou qualquer método de sua preferência.

Importe os dados de cada coleção para uma coleção do Amazon DocumentDB usando mongoimport com a opção apropriada para importar as linhas ou a lista:

linhas:

```
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
```

migração online

Considere uma migração on-line quando precisar minimizar o tempo de inatividade e as mudanças contínuas precisarem ser replicadas para o Amazon DocumentDB quase em tempo real.

Consulte Como realizar uma migração ao vivo do Couchbase para o Amazon DocumentDB para saber como realizar uma migração ao vivo para o Amazon DocumentDB. A documentação orienta você na implantação da solução e na execução de uma migração ao vivo de um bucket para um cluster do Amazon DocumentDB.

Tópicos

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

Couchbase Server 6.x ou anterior

Bucket Couchbase para a coleção Amazon DocumentDB

O utilitário de migração do Couchbase é pré-configurado para realizar uma migração on-line de um bucket do Couchbase para uma coleção do Amazon DocumentDB. Observando a configuração do conector de coletor, o `document.id.strategy` parâmetro é configurado para usar o valor da chave da mensagem como o valor do `_id` campo (consulte Propriedades da estratégia de ID do conector de coletor):

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

Couchbase Server 7.0 ou posterior

Bucket Couchbase com escopo padrão e coleção padrão

O utilitário de migração do Couchbase é pré-configurado para realizar uma migração on-line de um bucket do Couchbase para uma coleção do Amazon DocumentDB. Observando a configuração do conector de coletor, o `document.id.strategy` parâmetro é configurado para usar o valor da chave da mensagem como o valor do `_id` campo (consulte Propriedades da estratégia de ID do conector de coletor):

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

Coleções do Couchbase para coleções do Amazon DocumentDB

Configure o conector de origem para transmitir cada coleção do Couchbase em cada escopo para um tópico separado (consulte Opções de configuração de origem). Por exemplo:

```
ConnectorConfiguration:
  # add couchbase.collections configuration
```

```
  couchbase.collections: '<scope 1>.<collection 1>, <scope 1>.<collection 2>, ...'
```

Configure o conector de coletor para transmitir de cada tópico para uma coleção separada do Amazon DocumentDB (consulte Propriedades de configuração do conector de coletor). Por exemplo:

```
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>'
```

Validação

Esta seção fornece um processo de validação detalhado para verificar a consistência e a integridade dos dados após a migração para o Amazon DocumentDB. As etapas de validação se aplicam independentemente do método de migração.

Tópicos

- Verifique se todas as coleções existem no destino
- Verifique a contagem de documentos entre os clusters de origem e de destino
- Compare documentos entre clusters de origem e de destino

Verifique se todas as coleções existem no destino

Fonte do Couchbase

opção 1: banco de trabalho de consulta

```
SELECT RAW `path`
```

```
  FROM system:keyspaces
  WHERE `bucket` = '<bucket>'
```

opção 2: ferramenta cbq

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

Destino do Amazon DocumentDB

mongosh (consulte Conecte-se ao seu cluster Amazon DocumentDB):

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

Verifique a contagem de documentos entre os clusters de origem e de destino

Fonte do Couchbase

Couchbase Server 6.x ou anterior

opção 1: banco de trabalho de consulta

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

opção 2: cbq

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

Couchbase Server 7.0 ou posterior

opção 1: banco de trabalho de consulta

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

opção 2: cbq

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

Destino do Amazon DocumentDB

mongosh (consulte Conecte-se ao seu cluster Amazon DocumentDB):

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

Compare documentos entre clusters de origem e de destino

Fonte do Couchbase

Couchbase Server 6.x ou anterior

opção 1: banco de trabalho de consulta

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

opção 2: cbq

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

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

Couchbase Server 7.0 ou posterior

opção 1: banco de trabalho de consulta

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

opção 2: cbq

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

Destino do Amazon DocumentDB

mongosh (consulte Conecte-se ao seu cluster Amazon DocumentDB):

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

Datas de suporte da versão do mecanismo do Amazon DocumentDB

É possível usar as seguintes datas para planejar seus ciclos de teste e atualização.

Calendário de lançamento para as principais versões do Amazon DocumentDB

Atualmente, o Amazon DocumentDB oferece suporte às seguintes versões principais. Para atualizar a versão do seu motor, consulte Atualizando uma versão principal do motor ouAtualizando usando AWS DMS.

Versão do mecanismo	Data de lançamento	Término do suporte padrão	Início do Suporte estendido (preços do anos 1)	Início do Suporte estendido (preços do anos 3)	Término do Suporte estendido
Versão 3.6	9 de janeiro de 2019	30 de março de 2026	31 de março de 2026	31 de março de 2028	30 de março de 2029
Versão 4.0	9 de novembro de 2020	N/A	N/A	N/A	N/A
Versão 5.0	1 de março de 2023	N/A	N/A	N/A	N/A
Versão 8.0	14 de novembro de 2025	N/A	N/A	N/A	N/A

O Amazon DocumentDB oferece suporte a várias versões de mecanismos sob suporte padrão. É possível continuar executando uma versão principal após a data de término do seu suporte padrão mediante uma taxa de Suporte estendido. Para obter mais informações, consulte Suporte estendido do Amazon DocumentDB e Definição de preço do Amazon DocumentDB.

Calendário de lançamento para versões secundárias do Amazon DocumentDB

Atualmente, o Amazon DocumentDB oferece suporte às seguintes versões secundárias. O cronograma de lançamento variará dependendo dos recursos ou correções adicionais. As versões

secundárias podem chegar ao fim do suporte padrão antes das versões principais correspondentes. Para atualizar a versão do seu motor, consulteAtualizando uma versão secundária do motor.

> **Note**
>
> Versões secundárias estão disponíveis a partir da versão principal 5.0.

Versão do mecanismo	Data de lançamento
Versão 5.0.0 (LTS)	16 de fevereiro de 2026
Versão 5.0.1	8 de junho de 2026

O Amazon DocumentDB designa determinadas versões como versões do Long-Term Support (LTS). Para obter mais informações, consulte Usar um release de de suporte de longo prazo (LTS).

Atualização da versão principal implementada do Amazon DocumentDB no local

Você pode realizar uma atualização de versão principal (MVU) no local do seu cluster Amazon DocumentDB, mantendo os mesmos endpoints, armazenamento e tags. Seus aplicativos continuam funcionando sem modificações. Esse recurso está disponível sem custo adicional em todas as áreas em Regiões da AWS que o Amazon DocumentDB está disponível.

> **Important**
>
> Seu cluster não está disponível durante a atualização e é reinicializado várias vezes. Não se conecte, leia ou grave no cluster depois de iniciar a atualização. O tempo de inatividade varia de acordo com o número de coleções, índices, bancos de dados e instâncias. Execute a atualização durante sua janela de manutenção ou horas de baixa utilização.
> Depois de fazer o upgrade, você não pode fazer o downgrade para uma versão anterior. Você pode restaurar seu snapshot de pré-upgrade em um novo cluster, se necessário.

Tópicos

- Caminhos de atualização com suporte
- Pré-requisitos
- Práticas recomendadas
- O que muda após a atualização para o Amazon DocumentDB 8.0
- Post-upgrade considerações para clusters atualizados da versão 3.6 ou 4.0
- Executar a atualização
- Solução de problemas

Caminhos de atualização com suporte

A tabela a seguir lista as versões do mecanismo de destino que você pode selecionar para cada versão de origem. Não há necessidade de primeiro atualizar para a versão `.0` secundária e depois aplicar uma atualização de versão secundária.

Versão de origem	Versões de destino compatíveis
3.6	Qualquer versão secundária 5.0 ou 8.0 publicada
4,0	Qualquer versão secundária 5.0 ou 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	8.0.2

Para considerações pós-upgrade, consulte e. the section called “Post-upgrade considerações para clusters atualizados da versão 3.6 ou 4.0” the section called “O que muda após a atualização para o Amazon DocumentDB 8.0”

Para ver as versões secundárias disponíveis em sua AWS região, use o AWS CLI comando`aws docdb describe-db-engine-versions`. Para obter uma lista das versões secundárias lançadas, consulteNotas da versão.

> **Note**
>
> In-place O MVU não é compatível com clusters globais ou elásticos. Para atualizar um cluster global, remova os clusters secundários, converta o cluster primário em regional, execute a MVU e recrie o cluster global adicionando clusters secundários usando os mesmos nomes para reter seus endpoints. Você incorrerá em I/O cobranças enquanto o primário atualizado replica os dados para os novos secundários. Para obter detalhes das etapas, consulte, Remover um cluster de um cluster global do Amazon DocumentDB.

Pré-requisitos

> **Important**
>
> Amplie as instâncias com capacidade de intermitência antes de fazer o upgrade. Se seu cluster usa tipos de instância com capacidade de intermitência (por exemplo, `db.t3.medium` ou`db.t4g.medium`), expanda a instância primária para pelo menos `db.r5.large` ou `db.r6g.large` antes de iniciar a atualização. As instâncias com capacidade de intermitência podem não ter CPU e memória suficientes para concluir o processo de upgrade, o que pode resultar em falhas de upgrade e indisponibilidade prolongada do cluster. Você pode reduzir a escala após a conclusão da atualização.
> Amplie a instância do gravador sem servidor antes da atualização. Se seu cluster tiver uma instância de gravador sem servidor, o DCU máximo no `ServerlessV2ScalingConfiguration` deve ser definido como 3 ou superior antes de iniciar a atualização. Clusters com um máximo de DCU abaixo de 3 não podem ser atualizados porque não têm CPU ou memória suficientes para concluir o upgrade.
> Verifique seus índices parciais antes de fazer o upgrade (atualize do Amazon DocumentDB 5.0 para 8.0). Antes de iniciar a atualização, revise seus índices parciais para encontrar um `partialFilterExpression` que usa o `$type` operador para corresponder a determinados tipos de BSON: especificamente os aliases `binData`a`bool`,`date`, `double`e`decimal`, e de `array` tipo, ou um código de tipo numérico, como. `{ "$type" : 1 }` Esses índices parciais fazem com que a atualização falhe, mesmo que o índice funcione normalmente para leituras e gravações. Use `db.collection.getIndexes()` para revisar seus índices parciais. `partialFilterExpression` Elimine o índice parcial afetado antes de iniciar a atualização e recrie-o após a conclusão da atualização. Se você não tiver certeza se um índice foi afetado, entre em contato com o AWS suporte antes de fazer o upgrade.

- Tipo de instância — O Amazon DocumentDB 4.0+ não oferece suporte a instâncias db.r4. Modifique todas `db.r4.*` as instâncias para `db.r5.*` instâncias ou mais recentes antes de fazer o upgrade. Consulte Modificar uma instância do Amazon DocumentDB e Classes de instância compatíveis por região.
- Patches do sistema operacional — aplique todas as ações pendentes de manutenção do sistema operacional em todas as instâncias antes da atualização. Consulte Atualizações do sistema operacional do Amazon DocumentDB.

> **Note**
>
> Os patches pendentes do mecanismo no nível do cluster podem ocultar os patches do sistema operacional da instância. Aplique os patches do motor primeiro, se necessário. Consulte Executar uma atualização de patch para a versão do mecanismo de um cluster.

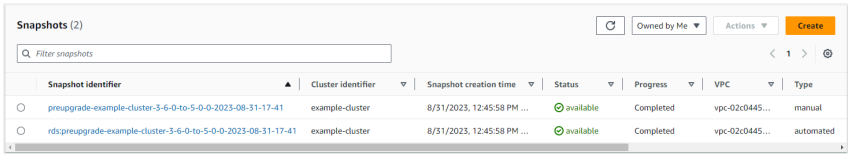
- Limites de índice em instâncias intermitentes (instâncias da família t) — Se você tiver mais de 3.000 índices em instâncias intermitentes, amplie o primário para pelo menos db.r5.large antes de fazer o upgrade. Você pode reduzir a escala após a conclusão da atualização.

Instância	Índices máximos para MVU
db.t4g.medium	3 mil
db.t3.medium	10 mil

- Grupo de parâmetros — Tenha um grupo de parâmetros de cluster personalizado para a versão de destino pronto antes da atualização. Se um não for especificado, o grupo de parâmetros padrão para a versão de destino será usado (por exemplo, `default.docdb5.0` ou`default.docdb8.0`).
- Instantâneo manual — Crie um instantâneo manual antes da atualização. O processo de upgrade cria um instantâneo automático chamado`preupgrade-<name>-<version>-<timestamp>`, mas sempre cria seu próprio backup. Consulte Criação de um snapshot manual de cluster.

> **Note**
>
> O snapshot automático criado pelo processo de atualização não será excluído automaticamente após a conclusão da atualização da versão principal implementada. Esse snapshot não incorrerá em nenhuma cobrança, desde que esteja dentro do período de

retenção. É possível optar por excluir esse snapshot depois de verificar uma atualização com êxito do seu cluster.



Práticas recomendadas

Antes da atualização — testando com um clone

1. Use Clonar um volume para um cluster Amazon DocumentDB para criar um clone do seu cluster. Você não incorrerá em custos de armazenamento, a menos que modifique os dados no clone.
2. Combine a contagem de instâncias do clone com o cluster de destino para obter uma estimativa de tempo realista.
3. Execute o MVU no clone e teste totalmente as diferenças funcionais.
4. Verifique se uma atualização já está agendada executando `describe-db-clusters` e procurando`PendingModifiedValues.EngineVersion`. Se você modificou o cluster e optou por aplicá-lo na próxima janela de manutenção, a programação não estará visível no console, mas você poderá visualizá-la em AWS CLI:

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

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

5. Se o teste for bem-sucedido, continue com a atualização em seu cluster de produção.

Durante a atualização

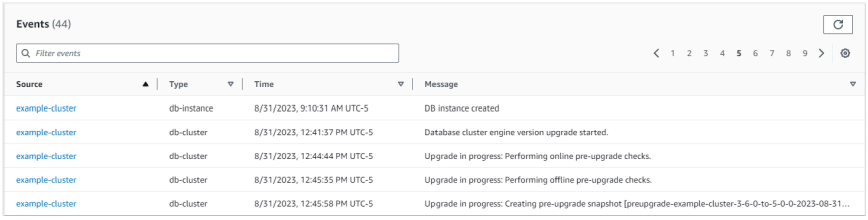
É possível monitorar o andamento da atualização da versão principal implementada assinando os eventos de manutenção do cluster. Quando a atualização for concluída, você receberá o evento

“A versão principal do cluster de banco de dados foi atualizada”. Esses e outros eventos que ocorrem durante a atualização aparecem na seção Eventos e tags da página de detalhes do cluster no console do Amazon DocumentDB. Em seguida, o status do cluster muda de `upgrading` para`available`.

Os seguintes eventos são gerados durante a atualização:

1. Foi iniciada a atualização da versão principal do mecanismo de cluster de banco de dados. O cluster permanece online.
2. Preparação da atualização em andamento: iniciando pré-verificações de atualização on-line.
3. Preparação da atualização em andamento: pré-verificações de atualização on-line concluídas.
4. Colocar o cluster de banco de dados off-line enquanto a instância primária conclui o patch/ upgrade processo.
5. Preparação da atualização em andamento: iniciando as pré-verificações de atualização off-line.
6. Preparação da atualização em andamento: pré-verificações de atualização off-line concluídas.
7. <version-to><timestamp>Atualização em andamento: Criação de um instantâneo de pré-atualização [preupgrade- <cluster-name>- <version-from>-to- -].
8. Atualização em andamento: volume de clonagem.
9. Atualização em andamento: atualizando gravador.
10. Atualização em andamento: atualizando leitores.
11. A versão principal do mecanismo do cluster de banco de dados foi atualizada.

Os eventos são visíveis no console na página Eventos:



A partir do AWS CLI, você pode executar `aws docdb describe-events` para monitorar o progresso da atualização. Para receber notificações automaticamente, use `aws docdb create-event-subscription` para assinar eventos e encaminhá-los ao Amazon SNS para receber e-mails, mensagens push ou outros métodos de entrega. Para obter mais informações, consulte Tornar-se assinante de eventos do Amazon DocumentDB.

```
aws docdb describe-events
```

```
  --source-identifier mydocdbcluster
  --source-type db-cluster
```

Esse comando retorna uma saída semelhante à seguinte:

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

Após a atualização

Warning

Imediatamente após a atualização local da versão principal, seu cluster Amazon DocumentDB repreenche os metadados de índice que o mecanismo de banco de dados usa para otimizar os planos de execução de consultas. O desempenho da consulta retorna aos níveis esperados quando esse processo é concluído. Normalmente, ele termina em alguns minutos, mas pode levar até duas horas, dependendo do número de índices em seu cluster. Não reinicialize, faça failover ou escale up/down sua instância de gravador durante esse período, pois isso pode interromper o recálculo dos metadados do índice. Espere até observar o desempenho esperado da consulta antes de fazer essas alterações. Acompanhe o progresso por meio dos seguintes eventos de cluster:

- Post-upgrade status do cluster: processo de atualização de metadados do índice iniciado
- Post-upgrade status do cluster: processo de atualização de metadados do índice concluído em X segundos

Entre em contato com o AWS suporte se o processo de atualização dos metadados do índice não for concluído em três horas ou se você continuar enfrentando problemas de desempenho após a conclusão do processo.

1. Faça um instantâneo manual do cluster atualizado caso precise restaurar para o estado pós-upgrade. O processo automático de captura instantânea será retomado assim que a atualização local da versão principal for concluída.
2. Clusters de tags atualizados da versão 3.6. Adicione uma tag para diferenciar os clusters atualizados da versão 3.6. (vejathe section called “Post-upgrade considerações para clusters atualizados da versão 3.6 ou 4.0”).
3. Atualize seu driver. Para usar novos recursos (por exemplo, agrupamento, visualizações ou compactação Zstd na versão 8.0), atualize para a versão correspondente da API MongoDB. Para obter mais informações, consulte O que há de novo no Amazon DocumentDB 8.0.
4. Teste minuciosamente. Valide seu aplicativo em relação ao cluster atualizado.

O que muda após a atualização para o Amazon DocumentDB 8.0

Depois de realizar uma atualização de versão principal para o Amazon DocumentDB 8.0 (do Amazon DocumentDB 5.0 ou diretamente da 3.6 ou 4.0), os seguintes recursos são habilitados ou alterados:

- Cotejamento. O Amazon DocumentDB 8.0 oferece suporte ao ??? agrupamento. Após a atualização, novas coleções e seus índices, e novos índices em coleções existentes, têm o agrupamento ativado por padrão.
- Índice de texto. Novos índices de texto são criados usando o Text Index V2, que usa um analisador de pesquisa de texto atualizado para melhorar a compatibilidade com o MongoDB. Os índices de texto existentes não são afetados.
- Versão do planejador de consultas. Se você não tiver um grupo de parâmetros personalizado, um novo grupo de parâmetros padrão será criado para o Amazon DocumentDB 8.0 com o Planner versão 3 selecionado automaticamente. Com a versão 3 do planejador de consultas, as visualizações também estão disponíveis.
- Compressão. O Amazon DocumentDB 8.0 oferece suporte à compactação de documentos baseada em dicionário usando o algoritmo Zstd. Após a atualização, novas coleções são criadas com a compactação Zstd ativada por padrão. As coleções existentes da versão 5.0 mantêm suas

configurações de compactação. Para aproveitar a compactação Zstd em coleções existentes, você pode modificar suas configurações de compactação. Para obter mais informações, consulte Gerenciando a compactação baseada em dicionário no Amazon DocumentDB 8.0.

- Reconstrução do índice. Se você estiver fazendo o upgrade do Amazon DocumentDB 5.0 para o Amazon DocumentDB 8.0, nenhuma reconstrução do índice será necessária. Se você estiver atualizando diretamente do Amazon DocumentDB 3.6 ou 4.0 para 8.0, reconstrua seus índices conforme descrito em. the section called “Post-upgrade considerações para clusters atualizados da versão 3.6 ou 4.0”

> **Important**
>
> O Amazon DocumentDB 8.0 exige TLS 1.2 ou superior. O TLS 1.0 e o TLS 1.1 não são mais suportados.

> **Note**
>
> Para obter uma lista completa das diferenças funcionais, consulteCompatibilidade do Amazon DocumentDB com MongoDB.

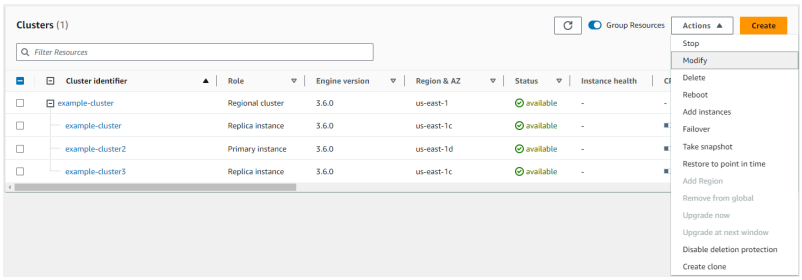
Post-upgrade considerações para clusters atualizados da versão 3.6 ou 4.0

- Reconstrução do índice. Uma MVU retém os índices originais. O Amazon DocumentDB 5.0 e versões posteriores aprimoraram a manutenção do índice e a coleta de lixo, especialmente para índices de baixa cardinalidade. Depois de atualizar da versão 3.6 ou 4.0 (para 5.0 ou 8.0), reconstrua seus índices para garantir o desempenho ideal da consulta (opcional, envolve mais). I/O Consulte Manutenção do índice usando `reIndex`.
- Comparação numérica de subdocumentos (somente 3.6). Os clusters atualizados da versão 3.6 herdam o comportamento da versão 3.6, em que os tipos numéricos nos subdocumentos não são comparados entre os tipos. Por exemplo, `{a: {b: NumberLong(1)}}` não é igual `{a: {b: 1}}` em 3.6, mas eles são iguais em 4.0 e posteriores. Esse comportamento afeta todos os clusters atualizados da versão 3.6.

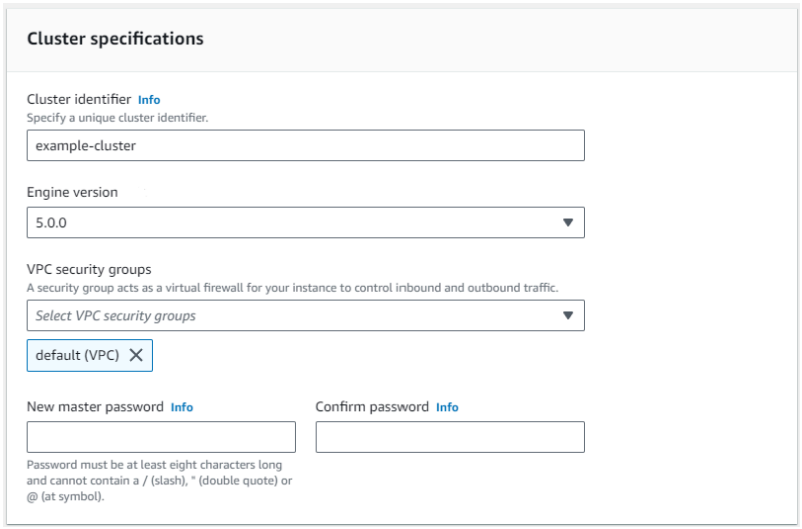
Executar a atualização

Using the Console de gerenciamento da AWS

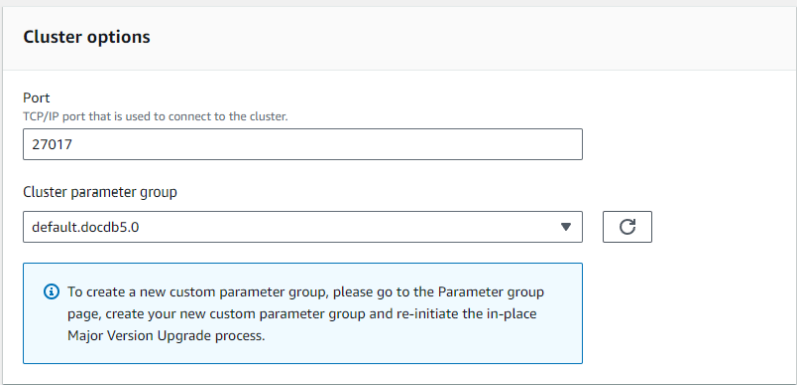
1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon DocumentDB.
2. Na tabela Clusters, selecione o cluster de origem, escolha Ações e, em seguida, Modificar.



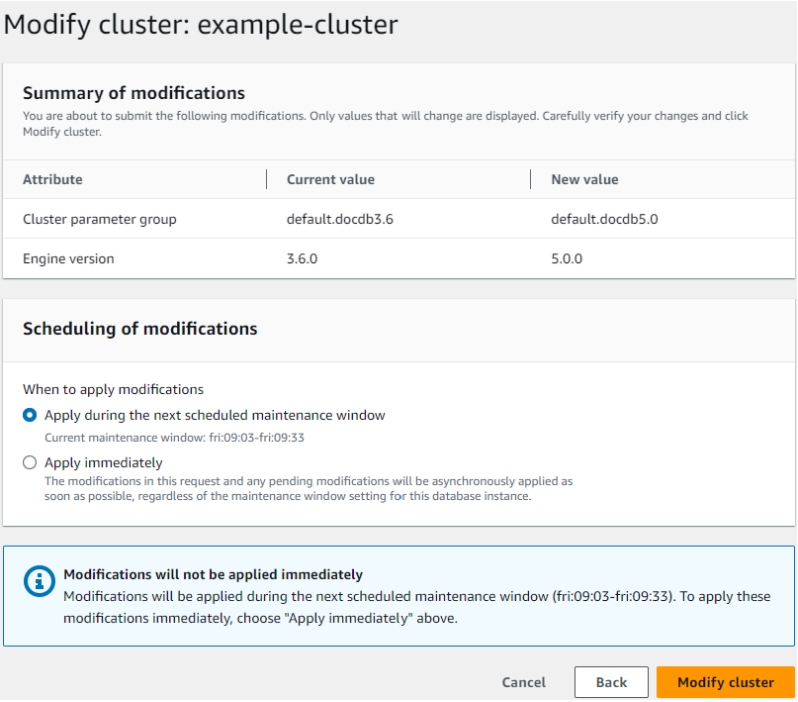
3. Em Especificações do cluster, escolha a versão de destino (por exemplo, 5.0.0 ou 8.0.0) no menu suspenso Versão do mecanismo.



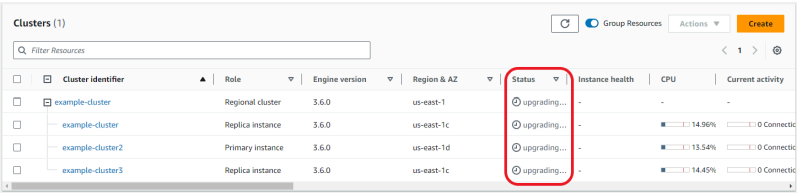
4. Em Opções de cluster, selecione seu grupo de parâmetros de cluster para a versão do mecanismo de destino. Você pode usar o padrão (por exemplo, default.docdb5.0 ou default.docdb8.0) ou um grupo de parâmetros personalizado que você criou.



5. Escolha Continuar, selecione sua preferência de agendamento (aplicar imediatamente ou na próxima janela de manutenção) e escolha Modificar cluster.



6. Monitore o status do cluster na tabela de clusters à medida que ele muda para a atualização:



Using the AWS CLI

Use `modify-db-cluster` com a `--allow-major-version-upgrade` bandeira:

```
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
```

Substitua cada um *placeholder* pelas informações do seu cluster.

Solução de problemas

- Pre-upgrade falha na verificação. Antes do início da atualização, o Amazon DocumentDB executa verificações de validação pré-atualização. A seguir estão as causas comuns de falhas na pré-verificação:
  - A atualização não pôde continuar porque as coleções têm nomes com 58 ou mais caracteres — Renomeie as coleções afetadas para nomes mais curtos antes de tentar a atualização novamente.
  - A atualização não pôde continuar porque a contagem do índice excede o limite do tipo de instância — Atualize para um tipo de instância maior antes de tentar a atualização novamente. Para limites de índice por tipo de instância, consultePré-requisitos.
- Falha na atualização e reversão. Se a atualização falhar, ela tentará automaticamente uma reversão. Uma reversão bem-sucedida gera o evento: “O cluster de banco de dados está em um estado que não pode ser atualizado”. Seu cluster retorna ao estado de pré-atualização e você pode continuar usando-o. Entre em contato com o AWS suporte para solucionar o problema antes de tentar novamente.
- Post-upgrade desempenho. A degradação temporária do desempenho e a alta utilização da CPU podem ocorrer enquanto a atualização dos metadados do índice é executada. Se a degradação persistir por mais de 3 horas, entre em contato com AWS o suporte.

Para obter assistência adicional, entre em contato com AWS Support.

Atualização da versão secundária do Amazon DocumentDB

> **Note**
>
> Este tópico aborda atualizações de versões secundárias dentro da mesma versão principal (por exemplo, atualização da 5.0.0 para a 5.0.1). Para obter informações sobre a atualização para uma versão principal diferente, consulteAtualização da versão principal implementada do Amazon DocumentDB no local.

Você pode atualizar a versão secundária do mecanismo do seu cluster Amazon DocumentDB (por exemplo, de 5.0.0 para 5.0.1) modificando o cluster. Atualizações de versões menores podem ser aplicadas imediatamente ou durante a próxima janela de manutenção.

Antes de realizar um upgrade da versão secundária

Antes de realizar uma pequena atualização de versão, as seguintes ações podem ser úteis:

- Execute a atualização durante um período de baixo tráfego para reduzir o impacto de qualquer breve tempo de inatividade.
- Verifique se a versão secundária de destino é uma meta de atualização válida para sua versão atual usando o `describe-db-engine-versions` comando.
- Para clusters globais, os clusters secundários devem ser atualizados antes do cluster primário. Para obter mais informações, consulte Aplicação de patches em clusters globais.

> **Important**
>
> - Os upgrades são operações unidirecionais — você não pode fazer o downgrade após o upgrade.
> - Seu cluster passará por um breve período de inatividade durante o processo de atualização.

Realizar uma atualização da versão secundária

Using the Console de gerenciamento da AWS

Para realizar uma atualização de versão secundária usando o Console de gerenciamento da AWS:

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon DocumentDB.
2. Na tabela Clusters, selecione o cluster que você deseja atualizar, escolha Ações e, em seguida, escolha Modificar.
3. Na caixa de diálogo Modificar cluster, na seção Especificações do cluster, escolha a versão secundária de destino no menu suspenso Versão do mecanismo.
4. Role para baixo e escolha Continuar.
5. Na seção Programação de modificações, escolha aplicar a alteração imediatamente ou durante a próxima janela de manutenção.

   Depois, selecione Modify Cluster (Modificar cluster).

Using the AWS CLI

Use o comando `modify-db-cluster` com o parâmetro `--engine-version`:

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

No exemplo acima, substitua cada um *`user input placeholder`* pelas informações do seu cluster.

> **Note**
>
> Para aplicar a atualização durante a próxima janela de manutenção, em vez de imediatamente, use `--no-apply-immediately` no lugar de`--apply-immediately`.

Verificando a atualização da versão secundária

Depois que a atualização for concluída, verifique a versão do mecanismo usando: AWS CLI

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

A saída dessa operação é semelhante à seguinte:

```
"5.0.1"
```

Atualizando seu cluster Amazon DocumentDB usando AWS Database Migration Service

> **Important**
>
> O Amazon DocumentDB não segue os mesmos ciclos de vida de suporte do MongoDB e o cronograma de fim de vida do MongoDB não se aplica ao Amazon DocumentDB.

Você pode atualizar seu cluster Amazon DocumentDB para uma versão superior com o mínimo de tempo de inatividade usando. AWS DMS AWS DMS é um serviço totalmente gerenciado que facilita a migração de versões mais antigas do Amazon DocumentDB, bancos de dados relacionais e bancos de dados não relacionais para seu cluster de destino do Amazon DocumentDB.

Tópicos

- Etapa 1: ativar fluxos de alterações
- Etapa 2: modificar a duração da retenção dos fluxos de alterações
- Etapa 3: migrar seus índices
- Etapa 4: criar um AWS DMS instância de replicação
- Etapa 5: criar um AWS DMS endpoint de origem
- Etapa 6: criar um AWS DMS endpoint alvo
- Etapa 7: criar e executar uma tarefa de migração

- Etapa 8: alterar o endpoint da aplicação para o cluster Amazon DocumentDB de destino

Etapa 1: ativar fluxos de alterações

Para realizar uma migração com o mínimo de tempo de inatividade, é AWS DMS necessário acesso aos fluxos de alteração do cluster. Os fluxos de alterações do Amazon DocumentDB fornecem uma sequência ordenada pelo tempo dos eventos atualizados que ocorrem dentro das coleções e bancos de dados do seu cluster. A leitura do stream de alterações permite realizar AWS DMS a captura de dados de alteração (CDC) e aplicar atualizações incrementais ao cluster Amazon DocumentDB de destino.

Para habilitar fluxos de alteração para todas as coleções em um banco de dados específico, autentique-se em seu cluster Amazon DocumentDB usando o shell mongo e execute os seguintes comandos:

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

Etapa 2: modificar a duração da retenção dos fluxos de alterações

Em seguida, modifique o período de retenção do fluxo de alterações com base em quanto tempo você gostaria de manter os eventos de alteração no fluxo de alterações. Por exemplo, se você espera que a migração do cluster do Amazon DocumentDB leve 12 horas, você deve definir a retenção do fluxo de alterações para um valor maior que 12 horas. AWS DMS O período de retenção padrão para o cluster do Amazon DocumentDB é de três horas. Você pode modificar a duração da retenção do log do stream de alterações do seu cluster Amazon DocumentDB para que fique entre uma hora e sete dias usando o Console de gerenciamento da AWS ou o. AWS CLI Para obter mais informações, consulte Modificar a duração da retenção do log do fluxo de alterações.

Etapa 3: migrar seus índices

Crie os mesmos índices em seu cluster Amazon DocumentDB de destino que você tem em seu cluster Amazon DocumentDB de origem. Embora AWS DMS gerencie a migração de dados, ele não migra índices. Para migrar os índices, use a ferramenta de índice do Amazon DocumentDB para exportar índices do cluster de origem do Amazon DocumentDB. Você pode obter a ferramenta criando um clone do repositório de GitHub ferramentas do Amazon DocumentDB e seguindo as

instruções em. README.md Você pode executar a ferramenta a partir de uma instância do Amazon EC2 ou de um AWS Cloud9 ambiente em execução na mesma Amazon VPC do seu cluster Amazon DocumentDB.

No exemplo a seguir, substitua cada *user input placeholder* por suas próprias informações.

O código a seguir despeja índices do seu cluster Amazon DocumentDB de origem:

```
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/
```

Depois que seus índices forem exportados com sucesso, restaure esses índices em seu cluster Amazon DocumentDB de destino. Para restaurar os índices que você exportou na etapa anterior, use a ferramenta de índice Amazon DocumentDB. O comando a seguir restaura os índices em seu cluster Amazon DocumentDB de destino a partir do diretório 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 você restaurou os índices corretamente, conecte-se ao seu cluster Amazon DocumentDB de destino com o shell mongo e liste os índices de uma determinada coleção. Use o seguinte 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

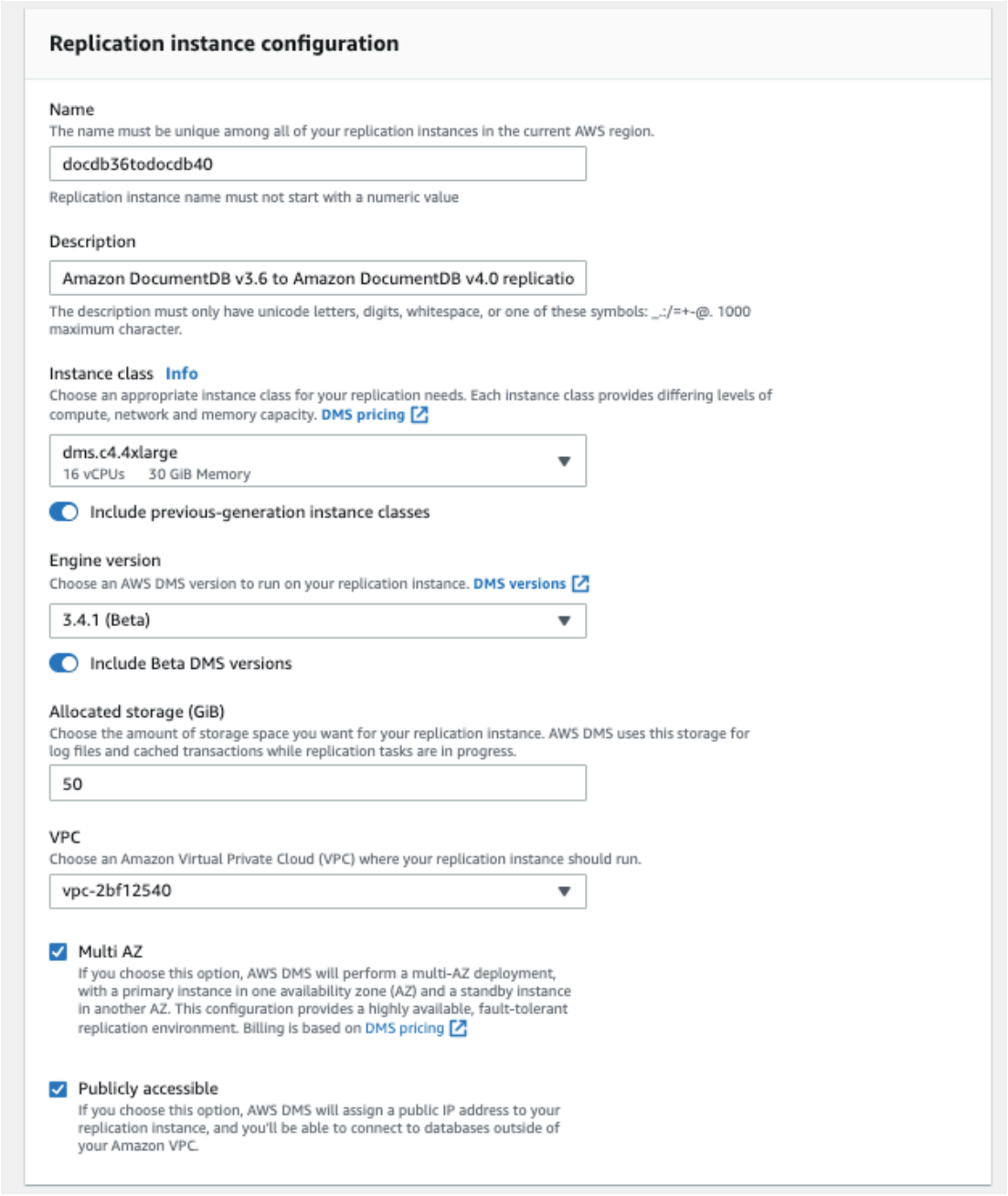
db.coll.getIndexes()
```

Etapa 4: criar um AWS DMS instância de replicação

Uma instância AWS DMS de replicação conecta e lê dados do seu cluster Amazon DocumentDB de origem e os grava no cluster Amazon DocumentDB de destino. A instância AWS DMS de replicação pode realizar operações de carga em massa e CDC. A maior parte desse processo ocorre na memória. No entanto, operações grandes podem exigir buffer no disco. Transações armazenadas em cache e arquivos de log também são gravados no disco. Depois que os dados são migrados, a instância de replicação também transmite quaisquer eventos de alteração para garantir que a origem e o destino estejam sincronizados.

Para criar uma instância de AWS DMS replicação:

1. Abra o AWS DMS console.
2. No painel de navegação, selecione Replication instances.
3. Selecione Criar instância de replicação e insira as seguintes informações:
   - Em Nome, insira um nome de sua escolha. Por exemplo, `.docdb36todocdb40`
   - Em Descrição, insira uma descrição de sua escolha. Para o item de lista, a instância de replicação do Amazon DocumentDB 3.6 para o Amazon DocumentDB 4.0.
   - Para a classe Instância, escolha o tamanho com base em suas necessidades.
   - Para a versão Mecanismo, escolha `3.4.1.`
   - Para a Amazon VPC, escolha a Amazon VPC que abriga seus clusters Amazon DocumentDB de origem e destino.
   - Para Armazenamento alocado (GiB), use o padrão de 50 GiB. Se você tiver uma workload de alto throughput de gravação, aumente esse valor para corresponder à sua workload.
   - Para Multi-AZ, escolha Sim se você precisar de alta disponibilidade e suporte de failover.
   - Para Acessível publicamente, habilite esta opção.



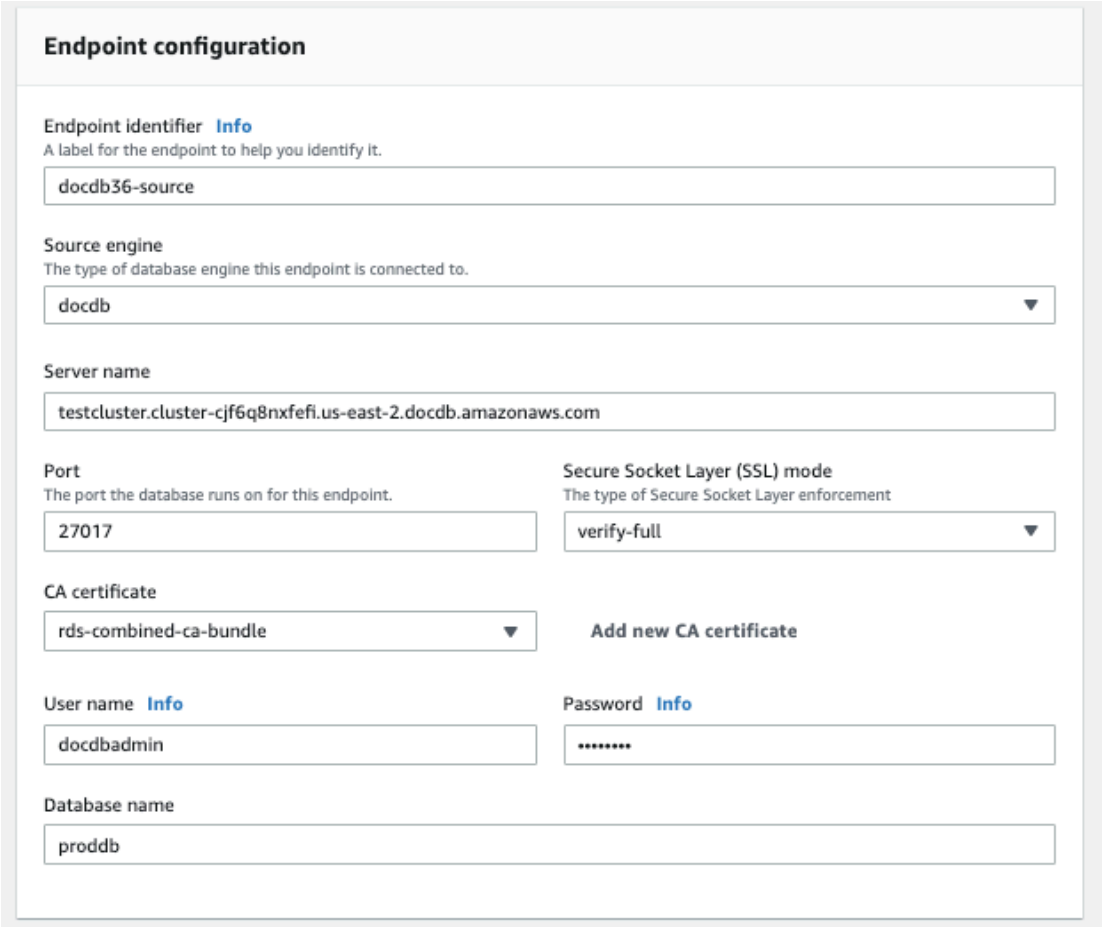
4. Selecione Create replication instance.

Etapa 5: criar um AWS DMS endpoint de origem

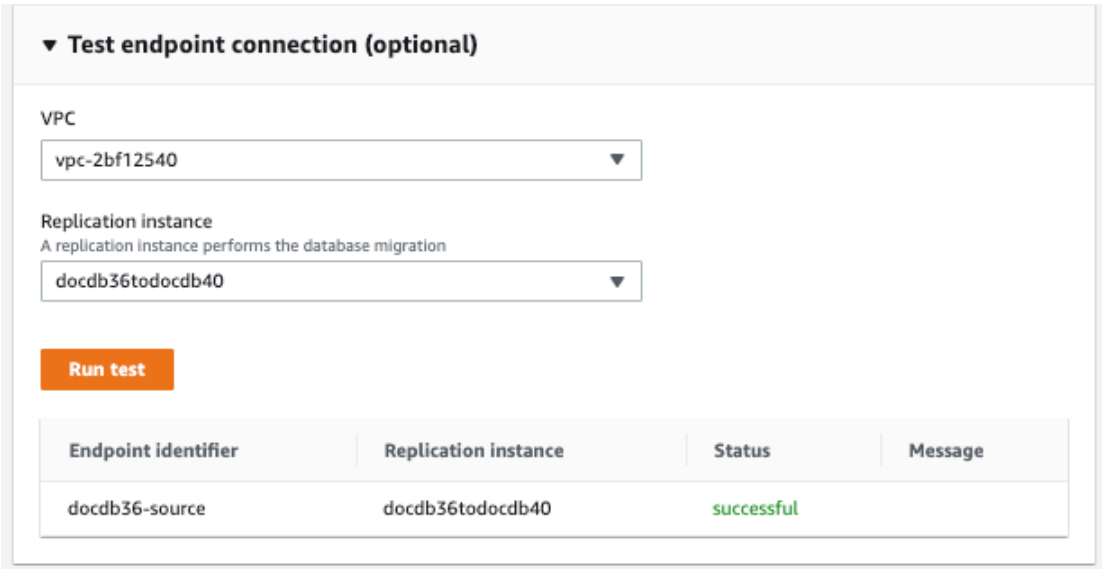
O endpoint de origem é usado para o cluster Amazon DocumentDB de origem.

Para criar um endpoint de origem

1. Abra o AWS DMS console.
2. No painel de navegação, escolha Endpoints.
3. Escolha `Create endpoint` e insira as seguintes informações:
   - Em Tipo de endpoint, selecione Origem.
   - >Em Identificador do endpoint, insira um nome que seja fácil de lembrar, por exemplo `docdb-source`.
   - Em Mecanismo de origem, escolha `docdb`.
   - Em Nome do servidor, insira o nome DNS do cluster do Amazon DocumentDB de sua origem.
   - Em Porta, insira o número da porta do cluster do Amazon DocumentDB de sua origem.
   - Para Modo SSL, escolha `verify-full`.
   - Para certificado CA, escolha Adicionar novo certificado CA. Baixe o novo certificado CA para criar um pacote de conexões TLS. Para Identificador de certificado, insira `rds-combined-ca-bundle` . Para Importar arquivo de certificado, escolha Escolher arquivo e navegue até o arquivo `.pem` que você baixou anteriormente. Selecione e abra o arquivo. Escolha Importar certificado, e `rds-combined-ca-bundle` na lista suspensa selecione Escolher um certificado
   - Em Nome de usuário, insira o nome de usuário principal do seu cluster de origem do Amazon DocumentDB.
   - Em Senha, digite a senha principal do cluster se origem do Amazon DocumentDB.
   - Em Nome do banco de dados, insira o nome do banco de dados que você deseja atualizar.



4. Teste sua conexão para verificar se ela foi configurada com sucesso.



5. Escolha Criar endpoint.

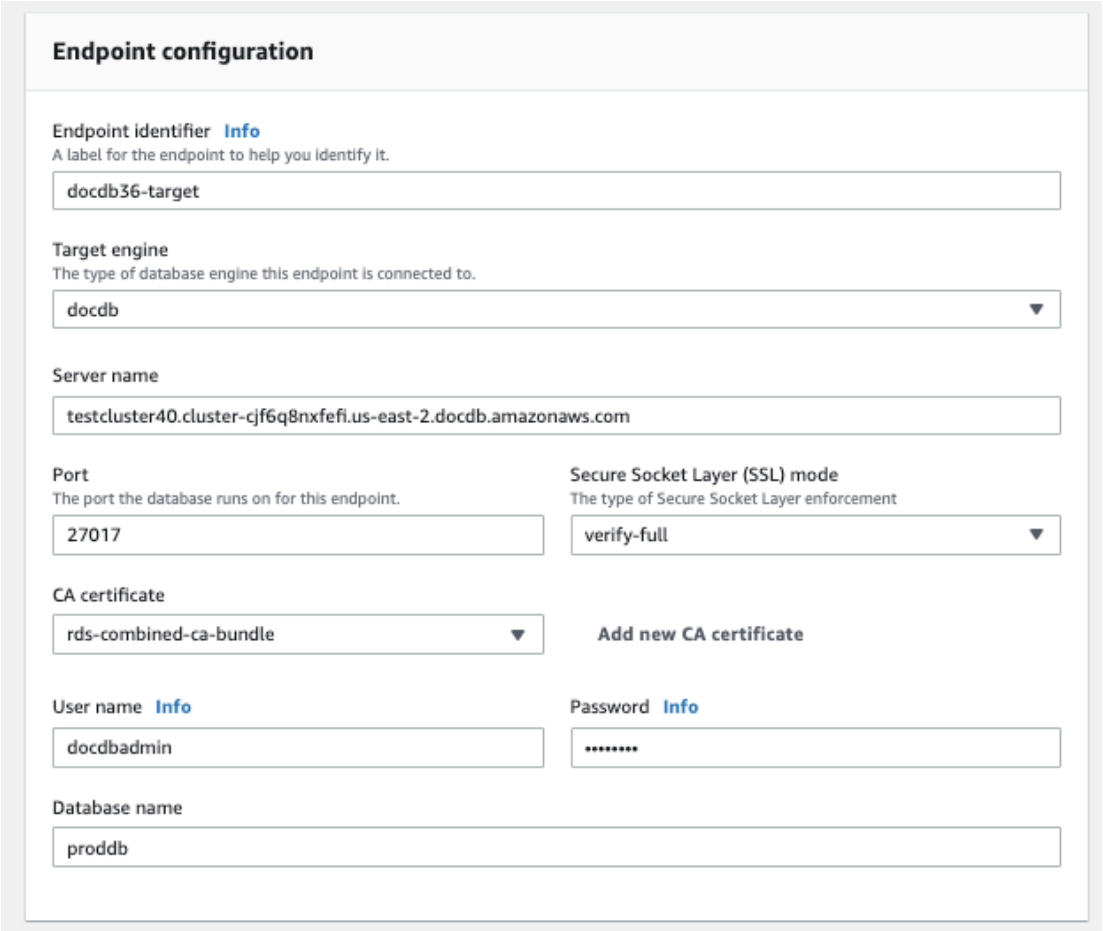
> **Note**
>
> AWS DMS só pode migrar um banco de dados por vez.

Etapa 6: criar um AWS DMS endpoint alvo

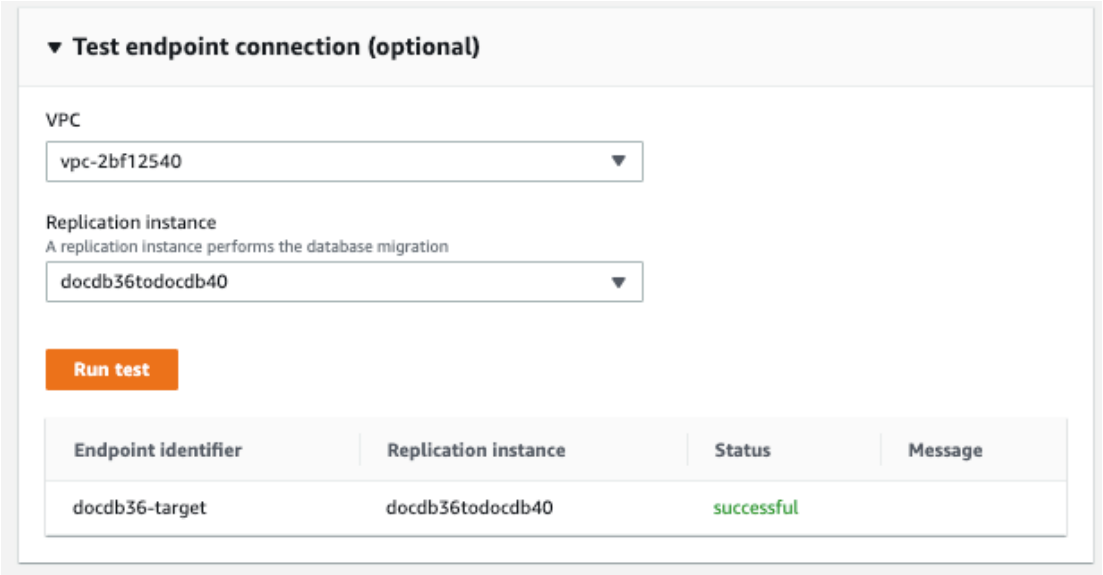
O endpoint de destino é para seu cluster Amazon DocumentDB de destino.

Para criar um endpoint de destino:

1. Abra o console do AWS DMS.
2. No painel de navegação, escolha Endpoints.
3. Selecione Criar endpoint e insira as seguintes informações:
   - Em Tipo de endpoint, selecione Destino.
   - Em Identificador de endpoint, insira um nome que seja fácil de lembrar, por exemplo `docdb-target`.
   - Em Mecanismo de origem, escolha `docdb`.
   - Em Nome do servidor, insira o nome DNS do cluster do Amazon DocumentDB de seu destino.
   - Em Porta, insira o número da porta do cluster do Amazon DocumentDB de seu destino.
   - Para Modo SSL, escolha `verify-full`.
   - Para certificado CA, escolha o `rds-combined-ca-bundle` certificado existente no menu suspenso Escolher um certificado.
   - Em Nome de usuário, insira o nome de usuário principal do seu cluster de destino do Amazon DocumentDB.
   - Em Senha, insira a senha principal do cluster de destino do Amazon DocumentDB.
   - Em Nome do banco de dados, insira o mesmo nome do banco de dados que você usou para configurar seu endpoint de origem.



4. Teste sua conexão para verificar se ela foi configurada com sucesso.



5. Escolha Criar endpoint.

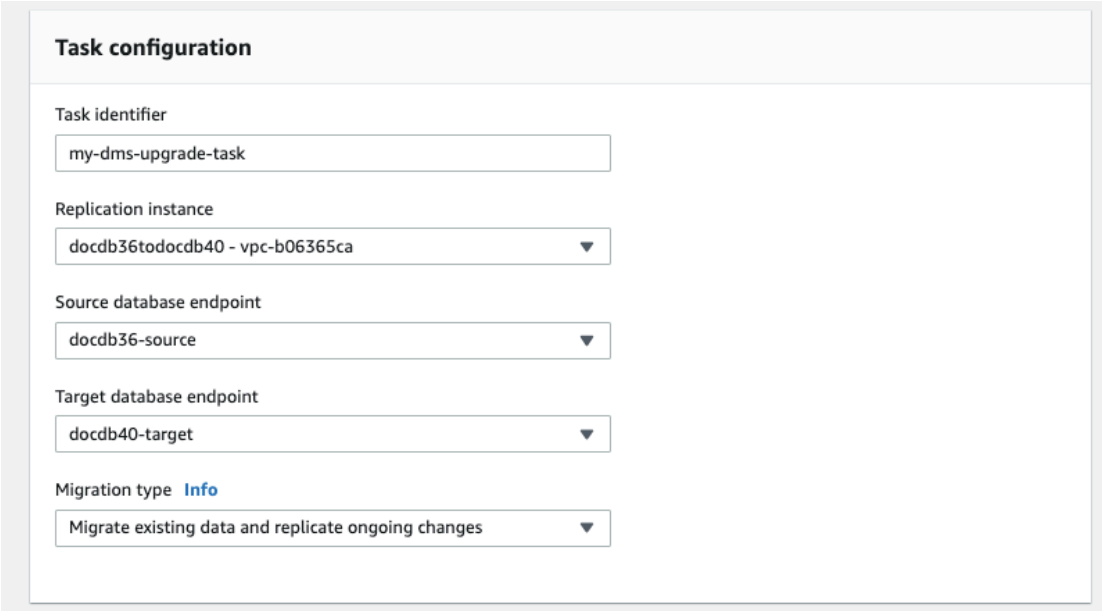
Etapa 7: criar e executar uma tarefa de migração

Uma AWS DMS tarefa vincula a instância de replicação à sua instância de origem e de destino. Ao criar uma tarefa de migração, você especifica o endpoint da origem, o endpoint de destino e a instância de replicação, com todas as configurações de migração. Uma AWS DMS tarefa pode ser criada com três tipos diferentes de migração: migrar dados existentes, migrar dados existentes e replicar alterações em andamento ou replicar somente alterações de dados. Como o objetivo dessa apresentação é atualizar um cluster do Amazon DocumentDB com o mínimo de tempo de inatividade, as etapas utilizam a opção de migrar dados existentes e replicar as alterações em andamento. Com essa opção, AWS DMS captura as alterações ao migrar seus dados existentes. AWS DMS continua capturando e aplicando alterações mesmo após o carregamento dos dados em massa. Por fim, os bancos de dados de origem e de destino ficarão sincronizados, permitindo uma migração com tempo de inatividade mínimo.

Abaixo estão as etapas para criar uma tarefa de migração para uma migração com o mínimo de tempo de inatividade:

1. Abra o AWS DMS console.
2. No painel de navegação, escolha Tarefas de migração de banco de dados.
3. Escolha Criar tarefa de migração de banco de dados e insira as informações a seguir na seção Configuração da tarefa:

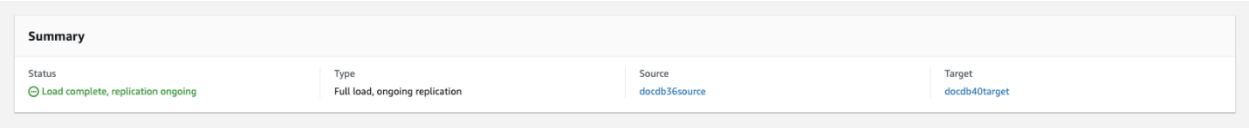
   - Em Identificador da tarefa, insira um nome que seja fácil de lembrar, por exemplo `my-dms-upgrade-task`.
   - Em Nome do recurso da Amazon (ARN) descritivo, insira um nome fácil de usar para substituir o ARN padrão do DMS.
   - Em Replication instance (Instância de replicação), escolha a instância de replicação que você criou em Etapa 4: criar um AWS DMS instância de replicação.
   - Em Endpoint de origem do banco de dados, escolha o endpoint de origem que você criou em Etapa 5: criar um AWS DMS endpoint de origem.
   - Em Endpoint de destino do banco de dados, escolha o endpoint de destino que você criou em Etapa 6: criar um AWS DMS endpoint alvo.
   - Em Tipo de migração, escolha Migrar e replicar.



4. Insira as informações a seguir na seção Configurações de tarefa:

   - Na seção Modo de preparação da tabela de destino, escolha Não fazer nada. Isso garantirá que os índices criados na etapa 3 não sejam descartados.
   - Na subseção Registros de tarefas, selecione Ativar CloudWatch registros.
   - Em Configuração de startup da tarefa de migração, escolha Automaticamente na criação. Isso iniciará a tarefa de migração automaticamente assim que você a criar.
   - Escolha Criar tarefa de migração de banco de dados.

AWS DMS agora começa a migrar dados do seu cluster Amazon DocumentDB de origem para seu cluster Amazon DocumentDB de destino. O status da tarefa deve alterar de Iniciando para Em execução. Você pode monitorar o progresso escolhendo Tarefas no AWS DMS console. Depois de várias minutes/hours (dependendo do tamanho da migração), o status deve mudar de Carga concluída, replicação em andamento. Isso significa que AWS DMS concluiu uma migração de carga completa do seu cluster Amazon DocumentDB de origem para um cluster Amazon DocumentDB de destino e agora está replicando eventos de alteração.



Eventualmente, sua origem e destino estarão sincronizados. É possível verificar se eles estão sincronizados executando uma operação `count()` em suas coleções para verificar se todos os eventos de alteração foram migrados.

Etapa 8: alterar o endpoint da aplicação para o cluster Amazon DocumentDB de destino

Depois que a carga completa estiver concluída e o processo de CDC estiver sendo replicado continuamente, você estará pronto para alterar o endpoint de conexão de banco de dados da sua aplicação do seu cluster Amazon DocumentDB de origem para seu cluster Amazon DocumentDB de destino.

Usar um release de de suporte de longo prazo (LTS)

O Amazon DocumentDB designa determinadas versões como versões do Long-Term Support (LTS). Os clusters de banco de dados que usam LTS podem permanecer na mesma versão por mais tempo e passar por menos ciclos de atualização do que os clusters que usam versões não LTS. As versões LTS serão corrigidas somente no caso de patches críticos de estabilidade e segurança, sem a introdução de novos recursos. Patches opcionais adicionais podem ser lançados, como métricas para depurar falhas críticas. Eles serão aplicados cumulativamente com os patches necessários.

Versão atual do LTS

A versão atual do LTS para o Amazon DocumentDB é:

- Versão secundária 5.0.0

Para determinar a versão em que seu cluster está, consultethe section called “Como verificar se meu cluster existente da versão 5.0.0 do motor está no LTS”.

Para obter mais informações sobre cronogramas de suporte e ciclos de lançamento para as versões LTS, consulte. Datas de suporte da versão do mecanismo do Amazon DocumentDB O Amazon DocumentDB anunciará a próxima versão do LTS por meio de canais de comunicação padrão.

> **Note**
>
> Uma versão LTS para o Amazon DocumentDB 8.0 ainda não foi designada. Se você precisar de LTS, use a versão 5.0.0 do mecanismo com a versão de patch 3.0.17983 ou posterior.

Quem deve usar o LTS

As versões LTS do Amazon DocumentDB foram projetadas para clusters com janelas de upgrade limitadas. O LTS é ideal para ambientes de produção em que a disponibilidade do banco de dados afeta diretamente as operações comerciais.

Considere as versões do LTS se:

- Seu aplicativo tem requisitos rígidos de tempo de atividade que limitam as oportunidades de upgrade
- Você tem todos os recursos e aprimoramentos necessários do mecanismo de banco de dados que seu aplicativo precisa

Em vez disso, considere as versões padrão se:

- Seu aplicativo pode lidar com o mínimo de interrupção
- Você quer os recursos e aprimoramentos mais recentes assim que eles forem lançados

> **Important**
>
> - As atualizações são operações unidirecionais - você não pode fazer o downgrade após a atualização
> - Seu cluster passará por um breve período de inatividade durante o processo de atualização

Como criar um novo cluster LTS

Para criar um cluster LTS, siga estas etapas:

1. No Management Console, crie um novo cluster da versão 5.0.0 do mecanismo
2. Verifique se você está usando a versão 5.0.0 do motor após a criação do AWS CLI cluster usando:

```
aws docdb describe-db-clusters \
```

```
    --db-cluster-identifier mydocdbcluster \
    --query 'DBClusters[0].EngineVersion'
```

A saída dessa operação é semelhante à seguinte:

```
"5.0.0"
```

Como fazer o upgrade da versão 3.6.0 ou 4.0.0 do motor para um cluster LTS 5.0.0

Para atualizar para um cluster LTS, siga estas etapas:

1. No Management Console, atualize a versão principal (MVU) do cluster da versão 3.6.0 ou 4.0.0 do mecanismo para 5.0.0. Para obter instruções sobre como executar uma MVU, consulte. Atualização da versão principal implementada do Amazon DocumentDB no local
2. Verifique se você está usando a versão 5.0.0 do motor após a AWS CLI atualização usando:

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

Como verificar se meu cluster existente da versão 5.0.0 do motor está no LTS

Para verificar se o cluster da versão 5.0.0 do seu motor está executando uma versão LTS:

1. Verifique se você está usando a versão 5.0.0 do motor usando: AWS CLI

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

2. Se necessário, atualize seu cluster atual da versão 5.0.0 do motor. Para obter instruções sobre como realizar uma atualização de patch, consulteExecutar uma atualização de patch para a versão do mecanismo de um cluster.

Suporte estendido do Amazon DocumentDB

O Suporte estendido do Amazon DocumentDB permite que você continue executando uma versão do mecanismo após o término do suporte padrão, mediante o pagamento de um custo adicional. Seus clusters são inscritos no sxtended eupport após o final da data de suporte padrão. Se você não concluir as atualizações até o final da data de suporte padrão da versão do mecanismo, seus clusters serão cobrados de acordo com a taxa de Suporte estendido.

Tópicos

- Visão geral do Suporte estendido
- Cobranças do Suporte estendido
- Responsabilidades

O Suporte estendido do Amazon DocumentDB fornece as atualizações e o suporte técnico a seguir:

- Atualizações de segurança para CVEs críticas e altas para sua instância ou cluster
- Correções de erros e patches para problemas críticos
- A capacidade de abrir casos de suporte e receber ajuda para solução de problemas dentro do acordo de serviço padrão do Amazon DocumentDB

Essa oferta paga oferece a você mais tempo para atualizar para uma versão com suporte do mecanismo. Por exemplo, a data de fim do suporte padrão do Amazon DocumentDB para a versão 3.6 é 30 de março de 2026, mas você não está pronto para fazer o upgrade para a versão 5.0 ou 8.0 antes dessa data. Nesse caso, o Amazon DocumentDB inscreve seu cluster da versão 3.6 no Suporte Estendido em 31 de março de 2026 para que você possa continuar usando a versão 3.6 do Amazon DocumentDB. A partir de 1º de julho de 2026, você AWS cobra um prêmio de suporte estendido para clusters Amazon DocumentDB versão 3.6.

O Suporte estendido do Amazon DocumentDB ficará disponível por até três anos após a data de término do suporte padrão da versão do mecanismo. Após esse período, se você não tiver atualizado a versão do seu mecanismo para uma versão com suporte, o Amazon DocumentDB atualizará a versão do seu mecanismo. Atualize para uma versão de mecanismo compatível o mais rápido possível.

Data de término do suporte padrão e datas de Suporte estendido do Amazon DocumentDB versão 3.6:

- Lançamento da versão 3.6: 9 de janeiro de 2019
- Término do suporte padrão da versão 3.6: 30 de março de 2026
- Início da versão 3.6: Suporte estendido, preço do primeiro ano — 1º de julho de 2026
- Início do Suporte estendido da versão 3.6, preço do ano 3: 31 de março de 2028
- Término do Suporte estendido da versão 3.6: 30 de março de 2029

Visão geral do Suporte estendido

O Amazon DocumentDB atualiza seu cluster para a última versão lançada do mecanismo (antes do final da data do Suporte estendido), caso seu cluster ainda não esteja nessa versão. O período de atualização não ocorrerá até depois da data de suporte padrão para essa versão do mecanismo.

É possível criar novos clusters com uma versão de mecanismo que tenha atingido a data de término do suporte padrão, se você já tiver um cluster nessa versão de mecanismo. O Amazon DocumentDB inscreverá esses novos clusters no Suporte estendido do Amazon DocumentDB e cobra por essa oferta. Se você fizer a atualização para uma versão do mecanismo que ainda esteja sob o suporte padrão do Amazon DocumentDB antes da data de término do suporte padrão da versão 3.6, você não pagará o prêmio de Suporte estendido.

É possível encerrar a inscrição no Suporte estendido do Amazon DocumentDB a qualquer momento. Para encerrar a inscrição, atualize cada cluster inscrito para uma versão mais nova do mecanismo que ainda esteja sob o suporte padrão do Amazon DocumentDB. O término da inscrição no Suporte estendido do Amazon DocumentDB entrará em vigor no dia em que você concluir a atualização para uma versão mais recente do mecanismo que ainda esteja sob o suporte padrão do Amazon DocumentDB.

Para ter mais informações sobre as datas de término do suporte padrão do Amazon DocumentDB e do Suporte estendido do Amazon DocumentDB, consulte Datas de suporte da versão do mecanismo do Amazon DocumentDB.

Cobranças do Suporte estendido

Você incorrerá em cobranças por todos os clusters em execução no Suporte Estendido a partir de 1º de julho de 2026. Para saber a data de término do suporte padrão do Amazon DocumentDB versão 3.6, consulte Datas de suporte da versão do mecanismo do Amazon DocumentDB.

A cobrança adicional pelo Suporte estendido do Amazon DocumentDB é interrompida com a realização de uma das ações a seguir:

- Atualização para uma versão do mecanismo coberta pelo suporte padrão.
- Exclusão do cluster que está executando a vesrão 3.6 após a data de término do suporte padrão.

Por exemplo, a versão 3.6 do Amazon DocumentDB entra em suporte estendido em 31 de março de 2026, mas as cobranças só começam em 1º de julho de 2026. Se você atualizar seu cluster Amazon DocumentDB versão 3.6 para o Amazon DocumentDB versão 5.0 em 31 de julho de 2026, você só será cobrado por 30 dias de suporte estendido no Amazon DocumentDB versão 3.6. Para obter mais informações, consulte Preços do Amazon DocumentDB.

Responsabilidades

A seguir estão as responsabilidades do Amazon DocumentDB e suas responsabilidades com o Suporte estendido do Amazon DocumentDB.

Responsabilidades do Amazon DocumentDB

Após a data de término do suporte padrão do Amazon DocumentDB versão 3.6, o Amazon DocumentDB aplicará patches, correções de erros e atualizações para clusters versão 3.6 inscritos no Suporte estendido.

Suas responsabilidades

Você é responsável por atualizar seu mecanismo para uma versão mais nova antes da data de término do Suporte estendido do Amazon DocumentDB versão 3.6. Se você não atualizar seu cluster após o término do Suporte estendido da versão 3.6 do Amazon DocumentDB, o Amazon DocumentDB tentará atualizar seu cluster para a versão mais recente do mecanismo compatível com o suporte padrão do Amazon DocumentDB.

Recursos e configurações do Amazon DocumentDB

As tabelas a seguir comparam recursos, capacidades e tipos de instâncias compatíveis entre as versões do mecanismo Amazon DocumentDB e os clusters elásticos. Use essas tabelas para determinar qual versão do mecanismo melhor atende às suas necessidades de carga de trabalho.

Status do motor

Status do motor por versão

Característica/Capacidade	v3.6	v4.0	v5.0	v8.0	
Status do suporte	EOL: março de 2026	Ativo	Ativo (LTS disponível)	Ativo (GA de novembro de 2025)	
Suporte estendido disponível	Sim (pago, até março de 2029)	N/A	N/A	N/A	
Versão LTS disponível	Não	Não	Sim (v5.0 LTS)	Não	
Clusters elásticos	Não	Não	Sim	Não	

Tipos de cluster

Tipos de cluster por versão

Cluster Type	v3.6	v4.0	v5.0	v8.0	Clusters elásticos
Instance-based agrupamento	Sim	Sim	Sim	Sim	Sim
Instâncias sem servidor	Não	Não	Sim	Sim	Não

Recursos

Características e configurações por versão do motor

Recurso	v3.6	v4.0	v5.0	v8.0	Clusters elásticos
Clusters globais	Não	Sim	Sim	Sim	Não
Multi-AZ (3 AZs)	Sim	Sim	Sim	Sim	Sim (por fragmento)
Suporte para réplicas de leitura	15	15	15	15	15 por fragmento
Transações ACID	Não	Sim	Sim	Sim	Não
Gravações repetíveis	Não	Não	Não	Sim (8.0.2+)	Não
Change streams	Sim	Sim	Sim	Sim	Não
Pesquisa vetorial	Não	Não	Sim	Sim	Não
Insights de Performance	Sim	Sim	Sim	Sim	Não
In-place atualização de versão principal (MVU)	Sim (até 5,0 ou 8,0)	Sim (até 5,0 ou 8,0)	Sim (até 8,0)	Não (somente alvo)	Não
Client-side criptografia em nível de campo (FLE)	Não	Não	Sim	Sim	Não
I/O-Optimized armazenamento	Não	Não	Sim	Sim	Não
Rotação de certificados TLS (sem reinicialização)	Sim (patch 1.0.208662+)	Sim (patch 2.0.10179+)	Sim (patch 3.0.4780+)	Sim	Sim

Recurso	v3.6	v4.0	v5.0	v8.0	Clusters elásticos
Cross-Region cópia instantânea	Sim	Sim	Sim	Sim	Sim
Cross-account compartil hamento de instantâneos	Sim	Sim	Sim	Sim	Não
Registros de auditoria e lentidão de consultas	Sim	Sim	Sim	Sim	Sim
Dual-stack IPv4/IPv6	Não	Sim (patch 2.0.11747 +)	Sim (patch 3.0.15902 +)	Sim	N/A
Collation (Agrupamento)	Não	Não	Não	Sim	Não
Visualizações	Não	Não	Não	Sim	Não
Suporte à compactação	Não	Não	LZ4	LZ4/Ztsd	Não
Índice de texto	Não	Não	V1	V2	Não
FedRAMP () GovCloud	Não	Não	Sim	Sim	Não

Tipos de instância

Suporte ao tipo de instância por versão

Tipo de instância	v3.6	v4.0	v5.0	v8.0	
db.r4 (geração anterior)	Sim	Não	Não	Não	
db.r5	Sim	Sim	Sim	Sim	
db.r6g (Graviton 2)	Não	Sim	Sim	Sim	

Tipo de instância	v3.6	v4.0	v5.0	v8.0	
db.r6gd (Graviton 2) NVMe-backed	Não	Não	Sim	Sim	
db.r8g (Graviton 4)	Não	Não	Sim	Sim	
db.t3.medium	Sim	Sim	Sim	Sim	
db.t4g.medium (Graviton 2)	Não	Sim	Sim	Sim	
tecnologia sem servidor	Não	Não	Sim	Sim	

AWS integrações

AWS suporte de integração por versão

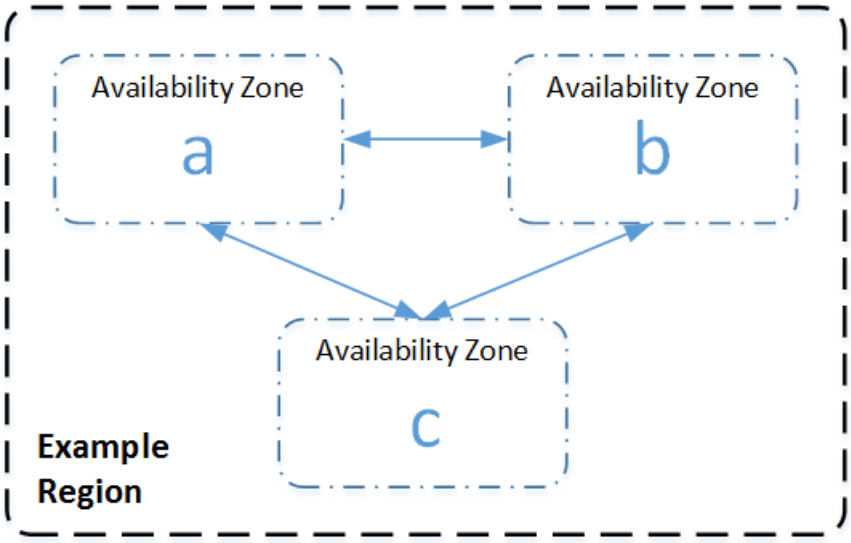
AWS Integração/Funcionalidade	v3.6	v4.0	v5.0	v8.0	Clusters elásticos
AWS Backup	Sim	Sim	Sim	Sim	Não
AWS Secrets Manager	Sim	Sim	Sim	Sim	Não
AWS Lambda (ESM nativo)	Não	Sim	Sim	Não	Não
Ingestão OpenSearch de serviços da Amazon () Zero-ETL	Não	Sim	Sim	Sim	Não
AWS DMS suporte à migração	Sim	Sim	Sim	Sim (a partir da versão 5.0 até AWS DMS)	Sim

AWS Integração/Funcionalidade	v3.6	v4.0	v5.0	v8.0	Clusters elásticos
AWS Identity and Access Management authentication	Não	Não	Sim (baseado em instância 5.0)	Sim	Não
AWS CloudShell	Não	Não	Sim	Sim	Não
AWS Key Management Service	Sim	Sim	Sim	Sim	Sim
AWS CloudTrail	Sim	Sim	Sim	Sim	Sim

Escolher regiões e zonas de disponibilidade

Os recursos de computação em nuvem da Amazon são hospedados em vários locais no mundo todo. Esses locais consistem em zonas de Regiões da AWS disponibilidade. Cada uma Região da AWS é uma área geográfica separada. Cada região contém vários locais isolados conhecidos como Zonas de Disponibilidade. O Amazon DocumentDB permite que você coloque recursos, como instâncias de banco de dados, e dados em vários locais. Os recursos não são replicados, Regiões da AWS a menos que você faça isso especificamente.

A Amazon opera data centers avançados e altamente disponíveis. Embora sejam raras, podem ocorrer falhas que afetam a disponibilidade das instâncias que estão no mesmo local. Se você hospeda todas as suas instâncias em um único local afetado por tal falha, nenhuma delas fica disponível. O diagrama a seguir mostra um Região da AWS com três zonas de disponibilidade.



É importante lembrar que cada região é totalmente independente. Qualquer atividade do Amazon DocumentDB iniciada por você (por exemplo, criar instâncias ou listar instâncias disponíveis) é executada somente em seu padrão atual Região da AWS. Você também pode alterar a região padrão no console, definindo a variável de ambiente `EC2_REGION`. Ou pode substituí-la usando o parâmetro `--region` na AWS CLI. Para obter mais informações, consulte Configurando AWS Command Line Interface, especificamente, as seções sobre variáveis de ambiente e opções de linha de comando.

Quando você cria um cluster usando o console do Amazon DocumentDB e opta por criar uma réplica em uma zona de disponibilidade diferente, o Amazon DocumentDB cria duas instâncias. Ele cria a instância primária em uma zona de disponibilidade e a instância de réplica em uma zona de disponibilidade diferente. O volume do cluster é sempre replicado em três zonas de disponibilidade.

Para criar ou trabalhar com uma instância do Amazon DocumentDB em uma instância específica Região da AWS, use o endpoint de serviço regional correspondente.

Disponibilidade de regiões

O Amazon DocumentDB está disponível nas seguintes AWS regiões.

Regiões compatíveis com o Amazon DocumentDB

Nome da Região	Região	Zonas de disponibilidade (computação)
Leste dos EUA (Ohio)	`us-east-2`	3

Nome da Região	Região	Zonas de disponibilidade (computação)
Leste dos EUA (Norte da Virgínia)	us-east-1	6
Oeste dos EUA (Oregon)	us-west-2	4
África (Cidade do Cabo)	af-south-1	3
América do Sul (São Paulo)	sa-east-1	3
Ásia-Pacífico (Hong Kong)	ap-east-1	3
Ásia-Pacífico (Hyderabad)	ap-south-2	3
Ásia-Pacífico (Malásia)	ap-southeast-5	3
Ásia-Pacífico (Mumbai)	ap-south-1	3
Ásia-Pacífico (Osaka)	ap-northeast-3	3
Ásia-Pacífico (Seul)	ap-northeast-2	4
Ásia-Pacífico (Singapura)	ap-southeast-1	3
Ásia-Pacífico (Sydney)	ap-southeast-2	3
Ásia-Pacífico (Jacarta)	ap-southeast-3	3

Nome da Região	Região	Zonas de disponibilidade (computação)
Ásia-Pacífico (Melbourne)	ap-southeast-4	3
Ásia-Pacífico (Tailândia)	ap-southeast-7	3
Ásia-Pacífico (Tóquio)	ap-northeast-1	3
Canadá (Central)	ca-central-1	3
Oeste do Canadá (Calgary)	ca-west-1	3
Região China (Pequim)	cn-north-1	3
China (Ningxia)	cn-northwest-1	3
Europa (Frankfurt)	eu-central-1	3
Europa (Zurique)	eu-central-2	3
Europa (Irlanda)	eu-west-1	3
Europa (Londres)	eu-west-2	3
Europa (Milão)	eu-south-1	3
Europa (Paris)	eu-west-3	3
Europa (Espanha)	eu-south-2	3
Europa (Estocolmo)	eu-north-1	3
México (Centro)	mx-central-1	3
Oriente Médio (Emirados Árabes Unidos)	me-central-1	3

Nome da Região	Região	Zonas de disponibilidade (computação)
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

Por padrão, o fuso horário de um cluster de bancos de dados Amazon DocumentDB é o Tempo Universal Coordenado (UTC).

Para obter informações sobre como encontrar os endpoints de conexão para clusters e instâncias em uma região específica, consulteNoções básicas sobre endpoints do Amazon DocumentDB.

Zero-ETL integração com o Amazon OpenSearch Service

Tópicos

- Amazon OpenSearch Service como destino
- Limitações

Amazon OpenSearch Service como destino

OpenSearch A integração de serviços com o Amazon DocumentDB permite que você transmita eventos de carga total e altere os eventos de dados para OpenSearch domínios. A infraestrutura de ingestão é hospedada como pipelines de OpenSearch ingestão e fornece um mecanismo de alta escala e baixa latência para transmitir continuamente dados das coleções do Amazon DocumentDB.

Durante a carga total, a integração Zero-ETL primeiro extrai dados históricos de carga total para OpenSearch usar um pipeline de ingestão. Depois que os dados de carga total forem ingeridos, os pipelines de OpenSearch ingestão começarão a ler os dados dos fluxos de alterações do Amazon DocumentDB e, eventualmente, se atualizarão para manter a consistência de dados quase em tempo real entre o Amazon DocumentDB e. OpenSearch OpenSearch armazena documentos em índices. Os dados recebidos de uma coleção do Amazon DocumentDB podem ser enviados para um índice ou podem ser particionados em índices diferentes. Os pipelines de ingestão sincronizarão

todos os eventos de criação, atualização e exclusão em uma coleção do Amazon DocumentDB conforme a criação, atualização e exclusão correspondentes de OpenSearch documentos para manter os dois sistemas de dados sincronizados. Os pipelines de ingestão podem ser configurados para ler dados de uma coleção e gravar em um índice ou ler dados de uma coleção e rotear condicionalmente para vários índices.

Os pipelines de ingestão podem ser configurados para transmitir dados do Amazon DocumentDB para o Amazon Service usando: OpenSearch

- Somente carga total
- Transmitir eventos de fluxo de alterações do Amazon DocumentDB sem carga total
- Carga total seguida por fluxos de alterações do Amazon DocumentDB

Para configurar seu pipeline de ingestão, execute as seguintes etapas:

Etapa 1: Crie um domínio do Amazon OpenSearch Service ou uma coleção OpenSearch sem servidor

É necessária uma coleção do Amazon OpenSearch Service com as permissões apropriadas para ler dados. Consulte Introdução ao Amazon OpenSearch Service ou Introdução ao Amazon OpenSearch Serverless no Guia do desenvolvedor do Amazon OpenSearch Service para criar uma coleção. Consulte o Amazon OpenSearch Ingestion no Guia do desenvolvedor do Amazon OpenSearch Service para criar uma função do AIM com as permissões corretas para acessar dados de gravação na coleção ou no domínio.

Etapa 2: ativar fluxos de alterações no cluster do Amazon DocumentDB

Certifique-se de que os fluxos de alterações estejam ativados nas coleções necessárias no cluster do Amazon DocumentDB. Consulte Usar fluxos de alterações com o Amazon DocumentDB para obter mais informações.

Etapa 3: configurar o perfil do pipeline com permissões para gravar no bucket do Amazon S3 e no domínio ou coleção de destino

Depois de criar a coleção do Amazon DocumentDB e ativar o fluxo de alterações, defina o perfil de pipeline que você deseja usar na configuração do pipeline e adicione as seguintes permissões a esse perfil:

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 um OpenSearch pipeline grave dados em um OpenSearch domínio, o domínio deve ter uma política de acesso em nível de domínio que permita que a função de pipeline sts_role_arn o acesse. O exemplo de política de acesso ao domínio a seguir permite que a função de pipeline chamada`pipeline-role`, que você criou na etapa anterior, grave dados no domínio chamado `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}/*"
    }
  ]
}
```

Etapa 4: adicione as permissões necessárias na função do pipeline para criar 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" }
```

```
            }
        }
    ]
}
```

Etapa 5: criar o pipeline

Configure um pipeline OpenSearch de ingestão especificando o Amazon DocumentDB como origem. Esse exemplo de configuração de pipeline pressupõe o uso de um mecanismo de busca do fluxo de alterações. Consulte Como usar um pipeline OpenSearch de ingestão com o Amazon DocumentDB no Guia do desenvolvedor do Amazon OpenSearch Service para obter mais informações.

Limitações

As seguintes limitações se aplicam à OpenSearch integração com o Amazon DocumentDB:

- Somente uma coleção do Amazon DocumentDB como fonte por pipeline é aceita.
- Cross-region a ingestão de dados não é suportada. Seu cluster e OpenSearch domínio do Amazon DocumentDB devem estar no mesmo Região da AWS.
- Cross-account a ingestão de dados não é suportada. Seu cluster e pipeline de OpenSearch ingestão do Amazon DocumentDB devem estar na mesma AWS conta.
- Clusters elásticos do Amazon DocumentDB não têm suporte. Somente clusters baseados em instâncias do Amazon DocumentDB têm suporte.
- Certifique-se de que o cluster Amazon DocumentDB tenha a autenticação habilitada usando AWS segredos. AWS segredos são o único mecanismo de autenticação suportado.
- A configuração do pipeline existente não pode ser atualizada para ingerir dados de um banco de dados diferente, de and/or uma coleção diferente. Para atualizar o nome da and/or coleção de banco de dados de um pipeline, você deve criar um novo pipeline.

Desenvolver com o Amazon DocumentDB

Esta seção aborda a conexão com seu cluster Amazon DocumentDB, a consulta e o gerenciamento de dados, o trabalho com índices e o uso de recursos como fluxos de alterações, transações e pesquisa vetorial para aplicativos generativos de IA.

Tópicos

- Conecte-se ao Amazon DocumentDB
- Programação com o Amazon DocumentDB
- Consultar no Amazon DocumentDB
- Índices
- Usar fluxos de alterações com o Amazon DocumentDB
- Transações no Amazon DocumentDB
- Gravações que podem ser repetidas no Amazon DocumentDB
- Usando agrupamento no Amazon DocumentDB 8.0
- Usando visualizações no Amazon DocumentDB 8.0
- Inteligência artificial generativa do Amazon DocumentDB
- Utilizar AWS Lambda com fluxos de mudança

Conecte-se ao Amazon DocumentDB

O Amazon DocumentDB fornece várias opções de conexão, incluindo conexões programáticas usando drivers MongoDB, conexão de fora de uma VPC por meio de tunelamento SSH ou VPN e ferramentas populares como Studio 3T e. AWS DataGrip O serviço oferece suporte a conexões de conjuntos de réplicas para alta disponibilidade. Para análises e relatórios, é possível se conectar usando drivers de JDBC ou ODBC. Instâncias do EC2 na mesma VPC podem se conectar diretamente aos clusters do DocumentDB. Todas as conexões exigem TLS e adequadas AWS credentials/authentication.

Tópicos

- Conectar-se de forma programática ao Amazon DocumentDB
- Conectar-se a um cluster do Amazon DocumentDB de fora de uma Amazon VPC
- Conectar-se ao Amazon DocumentDB como um conjunto de réplicas

- Conectar a um cluster Amazon DocumentDB a partir do Studio 3T
- Conecte-se ao Amazon DocumentDB usando DataGrip
- Conecte usando o Amazon EC2
- Conecte-se usando o driver JDBC do Amazon DocumentDB
- Conecte-se usando o driver ODBC do Amazon DocumentDB

Conectar-se de forma programática ao Amazon DocumentDB

Esta seção contém exemplos de código que demonstram como se conectar ao Amazon DocumentDB (compatível com MongoDB) usando diversas linguagens diferentes. Os exemplos são separados em duas seções com base na conexão com um cluster que tenha o Transport Layer Security (TLS) ativado ou desativado. Por padrão, o TLS fica ativado em clusters do Amazon DocumentDB. No entanto, se quiser, é possível desativar o TLS. Para obter mais informações, consulte Criptografia de dados em trânsito.

Se você estiver tentando se conectar ao seu Amazon DocumentDB de fora da VPC na qual seu cluster reside, consulte. Conectar-se a um cluster do Amazon DocumentDB de fora de uma Amazon VPC

Antes de se conectar ao cluster, é necessário saber se o TLS está ativado no cluster. A próxima seção mostra como determinar o valor do parâmetro `tls` do seu cluster usando o Console de gerenciamento da AWS ou a AWS CLI. Depois disso, você pode continuar localizando e aplicando o exemplo de código apropriado.

Tópicos

- Determinando o valor do seu parâmetro tls
- Conectar-se com o TLS habilitado
- Conectar-se com o TLS desabilitado
- Compressão de rede no Amazon DocumentDB

Determinando o valor do seu **`parâmetro tls`**

Determinar se o cluster tem o TLS habilitado é um processo de duas etapas que você pode executar usando o Console de gerenciamento da AWS ou. AWS CLI

1. Descubra qual grupo de parâmetros está regendo de seu cluster.

Using the Console de gerenciamento da AWS

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação à esquerda, escolha Clusters.
3. Na lista de clusters, selecione o nome do cluster.
4. A página resultante exibe os detalhes do cluster selecionado. Selecione a guia Configuration (Configuração). Na seção Configurações e status, localize o nome do grupo de parâmetros abaixo do Grupo de parâmetros do cluster.

Using the AWS CLI

O AWS CLI código a seguir determina qual parâmetro está governando seu cluster. Não se esqueça de substituir *sample-cluster* pelo nome do cluster.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].[DBClusterIdentifier,DBClusterParameterGroup]'
```

A saída dessa operação é semelhante à seguinte:

```
[
       [
           "sample-cluster",
           "sample-parameter-group"
       ]
]
```

2. Descubra o valor do parâmetro **`tls`** no grupo de parâmetros de seu cluster.

Using the Console de gerenciamento da AWS

1. No painel de navegação, selecione Grupos de parâmetros.
2. Na janela Grupos de parâmetros do cluster, selecione o nome do seu grupo de parâmetros do cluster da Etapa 1d.
3. A página resultante mostra os parâmetros do grupo de parâmetros do cluster. É possível ver o valor do parâmetro `tls` aqui. Para obter informações sobre como modificar

esse parâmetro, consulte Modificando grupos de parâmetros de cluster do Amazon DocumentDB.

Using the AWS CLI

Você pode usar o `describe-db-cluster-parameters` AWS CLI comando para visualizar os detalhes dos parâmetros em seu grupo de parâmetros do cluster.

- **`--describe-db-cluster-parameters`** — para listar todos os parâmetros em um grupo de parâmetros e seus valores.
  - **`--db-cluster-parameter-group name`** — obrigatório. O nome do grupo de parâmetros de cluster.

Nos exemplos a seguir, substitua cada um *`user input placeholder`* pelas informações do seu cluster.

```
aws docdb describe-db-cluster-parameters \
    --db-cluster-parameter-group-name sample-parameter-group
```

A saída dessa operação é semelhante à seguinte:

```
{
        "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**
>
> O Amazon DocumentDB oferece suporte a endpoints FIPS 140-3 começando com clusters do Amazon DocumentDB 5.0 (engine versão 3.0.3727) nas seguintes regiões: ca-central-1, us-west-2, us-east-1, us-east-2, us-gov-east-1, us-gov-east-1, us-gov-west-1.

Depois de descobrir o valor do parâmetro `tls`, continue a conexão ao cluster usando um dos exemplos de código nas seções a seguir.

- Conectar-se com o TLS habilitado
- Conectar-se com o TLS desabilitado

Conectar-se com o TLS habilitado

Para visualizar um exemplo de código para se conectar programaticamente a um cluster do TLS-enabled Amazon DocumentDB, escolha a guia apropriada para a linguagem que você deseja usar.

Para criptografar dados em trânsito, faça o download da chave pública para Amazon DocumentDB nomeada `global-bundle.pem` usando a operação a seguir.

```
wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
```

Se sua aplicação estiver no Microsoft Windows e exigir um arquivo PKCS7, é possível fazer o download de certificados PKCS7. Esse pacote contém os certificados intermediários e raiz em https://truststore.pki.rds.amazonaws.com/global/global-bundle.p7b.

Python

O código a seguir demonstra como se conectar ao Amazon DocumentDB usando Python quando o TLS está ativado.

No exemplo a seguir, substitua cada um *`user input placeholder`* pelas informações do seu cluster.

```
import pymongo
import sys

##Create a MongoDB client, open a connection to Amazon DocumentDB as a replica set
 and specify the read preference as secondary preferred
client = pymongo.MongoClient('mongodb://sample-user:password@sample-
cluster.node.us-east-1.docdb.amazonaws.com:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

##Specify the database to be used
db = client.sample_database

##Specify the collection to be used
col = db.sample_collection

##Insert a single document
col.insert_one({'hello':'Amazon DocumentDB'})

##Find the document that was previously written
x = col.find_one({'hello':'Amazon DocumentDB'})

##Print the result to the screen
print(x)

##Close the connection
client.close()
```

Node.js

O código a seguir demonstra como se conectar ao Amazon DocumentDB usando Node.js quando o TLS está habilitado.

> **Important**
>
> Há uma limitação conhecida com Node.js drivers anteriores à versão 6.13.1, que atualmente não são suportados pela autenticação de identidade IAM para o Amazon DocumentDB. Node.js drivers e ferramentas que usam Node.js driver (por exemplo, mongosh) devem ser atualizados para usar a versão 6.13.1 ou Node.js superior do driver.

No exemplo a seguir, substitua cada um *user input placeholder* pelas informações do seu cluster.

```
var MongoClient = require('mongodb').MongoClient

//Create a MongoDB client, open a connection to DocDB; as a replica set,
//  and specify the read preference as secondary preferred

var client = MongoClient.connect(
'mongodb://sample-user:password@sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017/sample-database?
tls=true&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false',
{
  tlsCAFile: `global-bundle.pem` //Specify the DocDB; cert
},
function(err, client) {
    if(err)
        throw err;

    //Specify the database to be used
    db = client.db('sample-database');

    //Specify the collection to be used
    col = db.collection('sample-collection');

    //Insert a single document
    col.insertOne({'hello':'Amazon DocumentDB'}, function(err, result){
      //Find the document that was previously written
      col.findOne({'hello':'Amazon DocumentDB'}, function(err, result){
        //Print the result to the screen
        console.log(result);

        //Close the connection
        client.close()
```

```
      });
   });
});
```

PHP

O código a seguir demonstra como se conectar ao Amazon DocumentDB usando PHP quando o TLS está ativado.

No exemplo a seguir, substitua cada um *user input placeholder* pelas informações do seu cluster.

```
<?php
//Include Composer's autoloader
require 'vendor/autoload.php';

$TLS_DIR = "/home/ubuntu/global-bundle.pem";

//Create a MongoDB client and open connection to Amazon DocumentDB
$client = new MongoDB\Client("mongodb://sample-user:password@sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017/?retryWrites=false", ["tls" => "true", "tlsCAFile"
 => $TLS_DIR ]);

//Specify the database and collection to be used
$col = $client->sampledatabase->samplecollection;

//Insert a single document
$result = $col->insertOne( [ 'hello' => 'Amazon DocumentDB'] );

//Find the document that was previously written
$result = $col->findOne(array('hello' => 'Amazon DocumentDB'));

//Print the result to the screen
print_r($result);
?>
```

Go

O código a seguir demonstra como se conectar ao Amazon DocumentDB usando Go quando o TLS está ativado.

> **Note**
>
> Desde a versão 1.2.1, o MongoDB Go Driver usará apenas o primeiro certificado de servidor CA encontrado em `sslcertificateauthorityfile`. O código de exemplo abaixo aborda essa limitação anexando manualmente todos os certificados de servidor encontrados em `sslcertificateauthorityfile` a uma configuração TLS personalizada usada durante a criação do cliente.

Nos exemplos a seguir, substitua cada um *`user input placeholder`* pelas informações do seu cluster.

```
package main

import (
 "context"
 "fmt"
 "log"
 "time"

 "go.mongodb.org/mongo-driver/bson"
 "go.mongodb.org/mongo-driver/mongo"
 "go.mongodb.org/mongo-driver/mongo/options"

 "io/ioutil"
 "crypto/tls"
 "crypto/x509"
 "errors"
)

const (
 // Path to the AWS CA file
 caFilePath = "global-bundle.pem"

 // Timeout operations after N seconds
 connectTimeout  = 5
 queryTimeout    = 30
 username        = "sample-user"
 password        = "password"
 clusterEndpoint = "sample-cluster.node.us-east-1.docdb.amazonaws.com:27017"

 // Which instances to read from
```

```
 readPreference = "secondaryPreferred"

 connectionStringTemplate = "mongodb://%s:%s@%s/sample-database?
tls=true&replicaSet=rs0&readpreference=%s"
)

func main() {

 connectionURI := fmt.Sprintf(connectionStringTemplate, username, password,
 clusterEndpoint, readPreference)

 tlsConfig, err := getCustomTLSConfig(caFilePath)
 if err != nil {
  log.Fatalf("Failed getting TLS configuration: %v", err)
 }

 client, err :=
 mongo.NewClient(options.Client().ApplyURI(connectionURI).SetTLSConfig(tlsConfig))
 if err != nil {
  log.Fatalf("Failed to create client: %v", err)
 }

 ctx, cancel := context.WithTimeout(context.Background(),
 connectTimeout*time.Second)
 defer cancel()

 err = client.Connect(ctx)
 if err != nil {
  log.Fatalf("Failed to connect to cluster: %v", err)
 }

 // Force a connection to verify our connection string
 err = client.Ping(ctx, nil)
 if err != nil {
  log.Fatalf("Failed to ping cluster: %v", err)
 }

 fmt.Println("Connected to DocumentDB!")

 collection := client.Database("sample-database").Collection("sample-collection")

 ctx, cancel = context.WithTimeout(context.Background(), queryTimeout*time.Second)
 defer cancel()
```

```
 res, err := collection.InsertOne(ctx, bson.M{"name": "pi", "value": 3.14159})
 if err != nil {
  log.Fatalf("Failed to insert document: %v", err)
 }

 id := res.InsertedID
 log.Printf("Inserted document ID: %s", id)

 ctx, cancel = context.WithTimeout(context.Background(), queryTimeout*time.Second)
 defer cancel()

 cur, err := collection.Find(ctx, bson.D{})

 if err != nil {
  log.Fatalf("Failed to run find query: %v", err)
 }
 defer cur.Close(ctx)

 for cur.Next(ctx) {
  var result bson.M
  err := cur.Decode(&result)
  log.Printf("Returned: %v", result)

  if err != nil {
   log.Fatal(err)
  }
 }

 if err := cur.Err(); err != nil {
  log.Fatal(err)
 }

}

func getCustomTLSConfig(caFile string) (*tls.Config, error) {
 tlsConfig := new(tls.Config)
 certs, err := ioutil.ReadFile(caFile)

 if err != nil {
  return tlsConfig, err
 }

 tlsConfig.RootCAs = x509.NewCertPool()
 ok := tlsConfig.RootCAs.AppendCertsFromPEM(certs)
```

```
if !ok {
 return tlsConfig, errors.New("Failed parsing pem file")
}

return tlsConfig, nil
```

Java

Ao se conectar a um cluster do TLS-enabled Amazon DocumentDB a partir de um aplicativo Java, seu programa deve usar o arquivo AWS de autoridade de certificação (CA) fornecido para validar a conexão. Para usar o certificado da Amazon RDS CA, faça o seguinte:

1. Faça o download do arquivo da Amazon RDS CA do https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem .
2. Crie um armazenamento confiável com o certificado da CA contido no arquivo executando os seguintes comandos. Certifique-se de alterá-lo *truststore-password* para outra coisa. Se estiver acessando um armazenamento confiável que contenha o certificado da CA antigo (`rds-ca-2015-root.pem`) e o novo (`rds-ca-2019-root.pem`), é possível importar o pacote de certificados para o armazenamento confiável.

   Veja a seguir um exemplo de script shell que importa o pacote de certificados para um armazenamento confiável em um sistema operacional Linux. Nos exemplos a seguir, substitua cada um *user input placeholder* por suas próprias informações. Mais notavelmente, onde quer que o diretório de exemplo *mydir* "" esteja localizado no script, substitua-o por um diretório que você criou para essa tarefa.

   ```
   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
```

Veja a seguir um exemplo de script do shell que importa o pacote de certificados em um armazenamento de confiança no 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. Use o `keystore` no seu programa definindo as seguintes propriedades do sistema na aplicação antes de estabelecer uma conexão com o cluster do Amazon DocumentDB.

```
javax.net.ssl.trustStore: truststore
javax.net.ssl.trustStorePassword: truststore-password;
```

4. O código a seguir demonstra como se conectar ao Amazon DocumentDB usando Java quando o TLS está ativado.

   No exemplo a seguir, substitua cada um *`user input placeholder`* pelas informações do seu cluster.

```
package com.example.documentdb;

import com.mongodb.client.*;
import org.bson.Document;

public final class Test {
    private Test() {
    }
    public static void main(String[] args) {

        String template = "mongodb://%s:%s@%s/sample-database?
ssl=true&replicaSet=rs0&readpreference=%s";
        String username = "sample-user";
        String password = "password";
        String clusterEndpoint = "sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017";
        String readPreference = "secondaryPreferred";
        String connectionString = String.format(template, username, password,
 clusterEndpoint, readPreference);

        String truststore = "truststore";
        String truststorePassword = "truststore-password";
```

```
        System.setProperty("javax.net.ssl.trustStore", truststore);
        System.setProperty("javax.net.ssl.trustStorePassword",
 truststorePassword);

        MongoClient mongoClient = MongoClients.create(connectionString);

        MongoDatabase testDB = mongoClient.getDatabase("sample-database");
        MongoCollection<Document> numbersCollection =
 testDB.getCollection("sample-collection");

        Document doc = new Document("name", "pi").append("value", 3.14159);
        numbersCollection.insertOne(doc);

        MongoCursor<Document> cursor = numbersCollection.find().iterator();
        try {
            while (cursor.hasNext()) {
                System.out.println(cursor.next().toJson());
            }
        } finally {
            cursor.close();
        }

    }
}
```

C# / .NET

O código a seguir demonstra como conectar-se ao Amazon DocumentDB usando C# / .NET quando o TLS está habilitado.

No exemplo a seguir, substitua cada um *`user input placeholder`* pelas informações do seu cluster.

```
using System;
using System.Text;
using System.Linq;
using System.Collections.Generic;
using System.Security.Cryptography;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using MongoDB.Driver;
```

```
using MongoDB.Bson;

namespace DocDB
{
    class Program
    {
        static void Main(string[] args)
        {
            string template = "mongodb://{0}:{1}@{2}/sampledatabase?
tls=true&replicaSet=rs0&readpreference={3}";
            string username = "sample-user";
            string password = "password";
            string readPreference = "secondaryPreferred";
            string clusterEndpoint="sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017";
            string connectionString = String.Format(template, username, password,
 clusterEndpoint, readPreference);

            string pathToCAFile = "<PATH/global-bundle.p7b_file>";

            // ADD CA certificate to local trust store
            // DO this once - Maybe when your service starts
            X509Store localTrustStore = new X509Store(StoreName.Root);
            X509Certificate2Collection certificateCollection = new
 X509Certificate2Collection();
            certificateCollection.Import(pathToCAFile);
            try
            {
                localTrustStore.Open(OpenFlags.ReadWrite);
                localTrustStore.AddRange(certificateCollection);
            }
            catch (Exception ex)
            {
                Console.WriteLine("Root certificate import failed: " + ex.Message);
                throw;
            }
            finally
            {
                localTrustStore.Close();
            }

            var settings = MongoClientSettings.FromUrl(new
 MongoUrl(connectionString));
            var client = new MongoClient(settings);
```

```
            var database = client.GetDatabase("sampledatabase");
            var collection =
 database.GetCollection<BsonDocument>("samplecollection");
            var docToInsert = new BsonDocument { { "pi", 3.14159 } };
            collection.InsertOne(docToInsert);
        }
    }
}
```

MongoDB Shell

O código a seguir demonstra como se conectar e consultar o Amazon DocumentDB usando a versão mais recente, mongosh, ou a versão anterior do shell do mongo, quando o TLS está habilitado.

Conecte-se ao Amazon DocumentDB com o mongosh.

> **Important**
>
> Há uma limitação conhecida com Node.js drivers anteriores à versão 6.13.1, que atualmente não são suportados pela autenticação de identidade IAM para o Amazon DocumentDB. Node.js drivers e ferramentas que usam Node.js driver (por exemplo, mongosh) devem ser atualizados para usar a versão 6.13.1 ou Node.js superior do driver.

Nos exemplos a seguir, substitua cada um *`user input placeholder`* pelas informações do seu cluster.

```
mongosh --tls --host cluster-end-point:27017 --tlsCAFile  global-bundle.pem --
username sample-user --password password --retryWrites false
```

Conecte-se ao Amazon DocumentDB com a versão anterior do shell do mongo

Se você usa o IAM, deve usar uma versão anterior do shell mongo. Insira uma das opções de comando a seguir:

```
mongo --ssl --host cluster-end-point:27017 --sslCAFile global-bundle.pem --
username sample-user --password password
```

Se você estiver usando uma versão igual ou superior a 4.2, use o código a seguir para conectar. A partir da versão 8.0.2 do mecanismo, o Amazon DocumentDB oferece suporte a gravações que podem ser repetidas. Nas versões anteriores do mecanismo, não há suporte para gravações repetidas. Se você estiver usando o shell mongo (não mongosh), não inclua o comando `retryWrites=false` em nenhuma string de código. Para obter mais informações, consulte Gravações que podem ser repetidas no Amazon DocumentDB.

```
mongo --tls --host cluster-end-point:27017 --tlsCAFile global-bundle.pem --
username sample-user --password password
```

Teste a conexão

1. Insira um único documento.

   ```
   db.myTestCollection.insertOne({'hello':'Amazon DocumentDB'})
   ```

2. Encontre o documento inserido anteriormente.

   ```
   db.myTestCollection.find({'hello':'Amazon DocumentDB'})
   ```

R

O código a seguir demonstra como se conectar ao Amazon DocumentDB com R usando o mongolite (https://jeroen.github.io/mongolite/) quando o TLS está habilitado.

No exemplo a seguir, substitua cada um *user input placeholder* pelas informações do seu cluster.

```
#Include the mongolite library.
library(mongolite)

mongourl <- paste("mongodb://sample-user:password@sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017/test2?ssl=true&",
          "readPreference=secondaryPreferred&replicaSet=rs0", sep="")

#Create a MongoDB client, open a connection to Amazon DocumentDB as a replica
#   set and specify the read preference as secondary preferred
client <-  mongo(url = mongourl, options = ssl_options(weak_cert_validation = F, ca
 ="<PATH/global-bundle.pem>"))
```

```
#Insert a single document
str <- c('{"hello" : "Amazon DocumentDB"}')
client$insert(str)

#Find the document that was previously written
client$find()
```

Ruby

O código a seguir demonstra como se conectar ao Amazon DocumentDB com o Ruby quando o TLS está habilitado.

No exemplo a seguir, substitua cada um *`user input placeholder`* pelas informações do seu cluster.

```
require 'mongo'
require 'neatjson'
require 'json'
client_host = 'mongodb://sample-cluster.node.us-east-1.docdb.amazonaws.com:27017'
client_options = {
   database: 'test',
   replica_set: 'rs0',
   read: {:secondary_preferred => 1},
   user: 'sample-user',
   password: 'password',
   ssl: true,
   ssl_verify: true,
   ssl_ca_cert: 'PATH/global-bundle.pem',
   retry_writes: false
}


begin
   ##Create a MongoDB client, open a connection to Amazon DocumentDB as a
   ##   replica set and specify the read preference as secondary preferred
   client = Mongo::Client.new(client_host, client_options)

   ##Insert a single document
   x = client[:test].insert_one({"hello":"Amazon DocumentDB"})

   ##Find the document that was previously written
   result = client[:test].find()
```

```
   #Print the document
   result.each do |document|
      puts JSON.neat_generate(document)
   end
end

#Close the connection
client.close
```

Conectar-se com o TLS desabilitado

Para ver um exemplo de código para se conectar programaticamente a um cluster do TLS-disabled Amazon DocumentDB, escolha a guia do idioma que você deseja usar.

Python

O código a seguir demonstra como conectar-se ao Amazon DocumentDB usando Python quando o TLS está desabilitado.

No exemplo a seguir, substitua cada um *`user input placeholder`* pelas informações do seu cluster.

```
## Create a MongoDB client, open a connection to Amazon DocumentDB as a replica set
 and specify the read preference as secondary preferred

import pymongo
import sys

client = pymongo.MongoClient('mongodb://sample-user:password@sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017/?
replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

##Specify the database to be used
db = client.sample_database

##Specify the collection to be used
col = db.sample_collection

##Insert a single document
col.insert_one({'hello':'Amazon DocumentDB'})

##Find the document that was previously written
```

```
x = col.find_one({'hello':'Amazon DocumentDB'})

##Print the result to the screen
print(x)

##Close the connection
client.close()
```

Node.js

O código a seguir demonstra como se conectar ao Amazon DocumentDB usando Node.js quando o TLS está desativado.

> **Important**
>
> Há uma limitação conhecida com Node.js drivers anteriores à versão 6.13.1, que atualmente não são suportados pela autenticação de identidade IAM para o Amazon DocumentDB. Node.js drivers e ferramentas que usam Node.js driver (por exemplo, mongosh) devem ser atualizados para usar a versão 6.13.1 ou Node.js superior do driver.

No exemplo a seguir, substitua cada um *user input placeholder* pelas informações do seu cluster.

```
var MongoClient = require('mongodb').MongoClient;

//Create a MongoDB client, open a connection to Amazon DocumentDB as a replica set,
//  and specify the read preference as secondary preferred
var client = MongoClient.connect(
'mongodb://sample-user:password@sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017/sample-database?
replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false',
{
  useNewUrlParser: true
},

function(err, client) {
    if(err)
        throw err;
    //Specify the database to be used
    db = client.db('sample-database');
```

```
    //Specify the collection to be used
    col = db.collection('sample-collection');

    //Insert a single document
    col.insertOne({'hello':'Amazon DocumentDB'}, function(err, result){
      //Find the document that was previously written
      col.findOne({'hello':'Amazon DocumentDB'}, function(err, result){
        //Print the result to the screen
        console.log(result);

        //Close the connection
        client.close()
      });
   });
});
```

PHP

O código a seguir demonstra como se conectar ao Amazon DocumentDB usando PHP quando o TLS está desabilitado.

No exemplo a seguir, substitua cada um *user input placeholder* pelas informações do seu cluster.

```
<?php
//Include Composer's autoloader
require 'vendor/autoload.php';

//Create a MongoDB client and open connection to Amazon DocumentDB
$client = new MongoDB\Client("mongodb://sample-user:password@sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017/?retryWrites=false");

//Specify the database and collection to be used
$col = $client->sampledatabase->samplecollection;

//Insert a single document
$result = $col->insertOne( [ 'hello' => 'Amazon DocumentDB'] );

//Find the document that was previously written
$result = $col->findOne(array('hello' => 'Amazon DocumentDB'));

//Print the result to the screen
print_r($result);
```

```
?>
```

Go

O código a seguir demonstra como se conectar ao Amazon DocumentDB usando Go quando o TLS está desabilitado.

No exemplo a seguir, substitua cada um *user input placeholder* pelas informações do seu cluster.

```
package main

import (
 "context"
 "fmt"
 "log"
 "time"

 "go.mongodb.org/mongo-driver/bson"
 "go.mongodb.org/mongo-driver/mongo"
 "go.mongodb.org/mongo-driver/mongo/options"
)

const (
 // Timeout operations after N seconds
 connectTimeout  = 5
 queryTimeout    = 30
 username        = "sample-user"
 password        = "password"
 clusterEndpoint = "sample-cluster.node.us-east-1.docdb.amazonaws.com:27017"

 // Which instances to read from
 readPreference           = "secondaryPreferred"
 connectionStringTemplate = "mongodb://%s:%s@%s/sample-database?
replicaSet=rs0&readpreference=%s"
)

func main() {

 connectionURI := fmt.Sprintf(connectionStringTemplate, username, password,
 clusterEndpoint, readPreference)

 client, err := mongo.NewClient(options.Client().ApplyURI(connectionURI))
```

```
if err != nil {
 log.Fatalf("Failed to create client: %v", err)
}

ctx, cancel := context.WithTimeout(context.Background(),
connectTimeout*time.Second)
defer cancel()

err = client.Connect(ctx)
if err != nil {
 log.Fatalf("Failed to connect to cluster: %v", err)
}

// Force a connection to verify our connection string
err = client.Ping(ctx, nil)
if err != nil {
 log.Fatalf("Failed to ping cluster: %v", err)
}

fmt.Println("Connected to DocumentDB!")

collection := client.Database("sample-database").Collection("sample-collection")

ctx, cancel = context.WithTimeout(context.Background(), queryTimeout*time.Second)
defer cancel()

res, err := collection.InsertOne(ctx, bson.M{"name": "pi", "value": 3.14159})
if err != nil {
 log.Fatalf("Failed to insert document: %v", err)
}

id := res.InsertedID
log.Printf("Inserted document ID: %s", id)

ctx, cancel = context.WithTimeout(context.Background(), queryTimeout*time.Second)
defer cancel()

cur, err := collection.Find(ctx, bson.D{})

if err != nil {
 log.Fatalf("Failed to run find query: %v", err)
}
defer cur.Close(ctx)
```

```
 for cur.Next(ctx) {
  var result bson.M
  err := cur.Decode(&result)
  log.Printf("Returned: %v", result)

  if err != nil {
   log.Fatal(err)
  }
 }

 if err := cur.Err(); err != nil {
  log.Fatal(err)
 }

}
```

Java

O código a seguir demonstra como se conectar ao Amazon DocumentDB usando Java quando o TLS está desabilitado.

No exemplo a seguir, substitua cada um *`user input placeholder`* pelas informações do seu cluster.

```
package com.example.documentdb;

import com.mongodb.MongoClient;
import com.mongodb.MongoClientURI;
import com.mongodb.ServerAddress;
import com.mongodb.MongoException;
import com.mongodb.client.MongoCursor;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.MongoCollection;
import org.bson.Document;


public final class Main {
    private Main() {
    }
    public static void main(String[] args) {

        String template = "mongodb://%s:%s@%s/sample-database?
replicaSet=rs0&readpreference=%s";
```

```
        String username = "sample-user";
        String password = "password";
        String clusterEndpoint = "sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017";
        String readPreference = "secondaryPreferred";
        String connectionString = String.format(template, username, password,
 clusterEndpoint, readPreference);

        MongoClientURI clientURI = new MongoClientURI(connectionString);
        MongoClient mongoClient = new MongoClient(clientURI);

        MongoDatabase testDB = mongoClient.getDatabase("sample-database");
        MongoCollection<Document> numbersCollection = testDB.getCollection("sample-
collection");

        Document doc = new Document("name", "pi").append("value", 3.14159);
        numbersCollection.insertOne(doc);

        MongoCursor<Document> cursor = numbersCollection.find().iterator();
        try {
            while (cursor.hasNext()) {
                System.out.println(cursor.next().toJson());
            }
        } finally {
            cursor.close();
        }

    }
}
```

C# / .NET

O código a seguir demonstra como se conectar ao Amazon DocumentDB usando C#/.NET quando o TLS está desabilitado.

No exemplo a seguir, substitua cada um *user input placeholder* pelas informações do seu cluster.

```
using System;
using System.Text;
using System.Linq;
using System.Collections.Generic;
using System.Security.Cryptography;
```

```
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using MongoDB.Driver;
using MongoDB.Bson;

namespace CSharpSample
{
   class Program
    {
       static void Main(string[] args)
        {
           string template = "mongodb://{0}:{1}@{2}/sampledatabase?
replicaSet=rs0&readpreference={3}";
           string username = "sample-user";
           string password = "password";
           string clusterEndpoint = "sample-cluster.node.us-
east-1.docdb.amazonaws.com:27017";
           string readPreference = "secondaryPreferred";
           string connectionString = String.Format(template, username, password,
 clusterEndpoint, readPreference);

           var settings = MongoClientSettings.FromUrl(new
 MongoUrl(connectionString));
           var client = new MongoClient(settings);

           var database = client.GetDatabase("sampledatabase");
           var collection =
 database.GetCollection<BsonDocument>("samplecollection");
           var docToInsert = new BsonDocument { { "pi", 3.14159 } };
            collection.InsertOne(docToInsert);
       }
    }
}
```

MongoDB Shell

O código a seguir demonstra como se conectar e consultar o Amazon DocumentDB usando a versão mais recente, mongosh, ou a versão anterior do shell do mongo, quando o TLS está desabilitado.

Conecte-se ao Amazon DocumentDB com o mongosh.

> **Important**
>
> Há uma limitação conhecida com Node.js drivers anteriores à versão 6.13.1, que atualmente não são suportados pela autenticação de identidade IAM para o Amazon DocumentDB. Node.js drivers e ferramentas que usam Node.js driver (por exemplo, mongosh) devem ser atualizados para usar a versão 6.13.1 ou Node.js superior do driver.

Nos exemplos a seguir, substitua cada um *user input placeholder* pelas informações do seu cluster.

```
mongosh --host cluster-end-point:27017 --username sample-user --password password --
retryWrites false
```

Conecte-se ao Amazon DocumentDB com a versão anterior do shell do mongo

Se você usa o IAM, deve usar uma versão anterior do shell mongo. Insira uma das opções de comando a seguir:

```
mongo --host cluster-end-point:27017 --username sample-user --password password
```

Se você estiver usando uma versão igual ou superior a 4.2, use o código a seguir para conectar. Não há suporte para gravações com novas tentativas no Amazon DocumentDB. Se você estiver usando o shell mongo (não mongosh), não inclua o comando `retryWrites=false` em nenhuma string de código. Por padrão, as gravações que podem ser repetidas estão desabilitadas. A inclusão de `retryWrites=false` pode causar falha nos comandos de leitura normais.

```
mongo --host cluster-end-point:27017 --username sample-user --password password
```

Teste a conexão

1. Insira um único documento.

   ```
   db.myTestCollection.insertOne({'hello':'Amazon DocumentDB'})
   ```

2. Encontre o documento inserido anteriormente.

   ```
   db.myTestCollection.find({'hello':'Amazon DocumentDB'})
   ```

R

O código a seguir demonstra como se conectar ao Amazon DocumentDB com R usando o mongolite (https://jeroen.github.io/mongolite/) quando o TLS está desabilitado.

No exemplo a seguir, substitua cada um *user input placeholder* pelas informações do seu cluster.

```
#Include the mongolite library.
library(mongolite)

#Create a MongoDB client, open a connection to Amazon DocumentDB as a replica
#   set and specify the read preference as secondary preferred
client <- mongo(url = "mongodb://sample-user:password@sample-
cluster.node.us-east-1.docdb.amazonaws.com:27017/sample-database?
readPreference=secondaryPreferred&replicaSet=rs0")

##Insert a single document
str <- c('{"hello" : "Amazon DocumentDB"}')
client$insert(str)

##Find the document that was previously written
client$find()
```

Ruby

O código a seguir demonstra como se conectar ao Amazon DocumentDB com o Ruby quando o TLS está desabilitado.

No exemplo a seguir, substitua cada um *user input placeholder* pelas informações do seu cluster.

```
require 'mongo'
require 'neatjson'
require 'json'
client_host = 'mongodb://sample-cluster.node.us-east-1.docdb.amazonaws.com:27017'
client_options = {
   database: 'test',
   replica_set: 'rs0',
   read: {:secondary_preferred => 1},
   user: 'sample-user',
   password: 'password',
```

```
   retry_writes: false
}

begin
   ##Create a MongoDB client, open a connection to Amazon DocumentDB as a
   ##   replica set and specify the read preference as secondary preferred
   client = Mongo::Client.new(client_host, client_options)

   ##Insert a single document
   x = client[:test].insert_one({"hello":"Amazon DocumentDB"})

   ##Find the document that was previously written
   result = client[:test].find()

   #Print the document
   result.each do |document|
      puts JSON.neat_generate(document)
   end
end

#Close the connection
client.close
```

Compressão de rede no Amazon DocumentDB

Com a compactação de protocolo de fio, você pode reduzir os dados que seu aplicativo envia pela rede. O Amazon DocumentDB 8.0.1 e versões posteriores oferecem suporte à compactação. Você obtém um rendimento aprimorado para cargas de trabalho que transferem documentos grandes ou grandes conjuntos de resultados. Depois de ativar a compactação, o driver e o servidor trocam mensagens compactadas.

Ao ativar a compactação de rede, você troca recursos de CPU por largura de banda de rede reduzida. Ative a compactação quando sua carga de trabalho estiver vinculada à rede e transferir grandes cargas úteis. Para solicitações e respostas pequenas, talvez você não veja um benefício líquido, pois a sobrecarga de compressão pode exceder a economia.

Compressores compatíveis

O Amazon DocumentDB oferece suporte ao `snappy` compressor. O Amazon DocumentDB não oferece suporte a outros compressores que os drivers do MongoDB oferecem, como e. `zlib` `zstd`

Se um driver solicitar uma lista de compressores, o Amazon DocumentDB negocia somente os compatíveis. Por exemplo, se um driver oferece `snappy ezstd`, o Amazon DocumentDB seleciona `snappy` e ignora. `zstd` Se um driver oferecer somente compressores não suportados, o Amazon DocumentDB não comprime a conexão. Nesse caso, o driver e o servidor trocam mensagens sem compressão.

Ativar compressão

Para ativar a compactação, adicione a `compressors` opção à sua cadeia de conexão e defina-a como`snappy`. Na cadeia de conexão a seguir *`usernamepassword`*, substitua e *`docdb-cluster-endpoint`* com seus próprios valores:

```
mongodb://username:password@docdb-cluster-endpoint:27017/?
tls=true&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false&compressors=snappy
```

Ao usar um driver MongoDB, defina a mesma opção por meio das opções de conexão do driver. A maioria dos drivers exige uma dependência adicional do lado do cliente para realizar uma compactação rápida:

- O Node.js motorista exige o `snappy` pacote.
- O driver Python (PyMongo) requer o `python-snappy` pacote. No Linux, `python-snappy` também requer a biblioteca snappy do sistema (por exemplo, por `snappy-devel` meio de `dnf` ou`yum`).

Para obter o nome exato da opção e os pacotes necessários, consulte a documentação do motorista.

> **Note**
>
> Se a dependência de compressão do lado do cliente estiver ausente, o driver se conectará com êxito. No entanto, ele troca mensagens sem compressão e não retorna um erro. Para confirmar se a compressão está ativa, verifique o compressor negociado conforme descrito emVerifique a compressão.

O exemplo de código Python a seguir se conecta ao seu cluster com a compactação rápida ativada e usa um `except` bloco `try` and para lidar com erros de conexão:

```
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}')
```

Verifique a compressão

Para confirmar se um driver e o Amazon DocumentDB negociaram a compressão, inspecione o `compression` campo que o comando wire protocol do `hello` MongoDB retorna. Quando o driver e o Amazon DocumentDB negociam a compressão, esse campo lista os compressores acordados.

Conectar-se a um cluster do Amazon DocumentDB de fora de uma Amazon VPC

Os clusters do Amazon DocumentDB (compatível com MongoDB) são implantados dentro de uma Amazon Virtual Private Cloud (Amazon VPC). Eles podem ser acessados diretamente pelas instâncias do Amazon EC2 ou por outros AWS serviços implantados na mesma Amazon VPC. Além disso, o Amazon DocumentDB pode ser acessado por instâncias do EC2 ou outros AWS serviços em diferentes VPCs na mesma Região da AWS ou em outras regiões por meio do emparelhamento de VPCs.

No entanto, suponha que seu caso de uso exija que você (ou sua aplicação) acesse seus recursos do Amazon DocumentDB de fora da VPC do cluster. Nesse caso, é possível usar o tunelamento SSH (também conhecido como encaminhamento de porta) para acessar seus recursos do Amazon DocumentDB.

Não faz parte da finalidade deste tópico abordar detalhadamente o tunelamento SSH. Para obter mais informações sobre o tunelamento SSH, consulte o seguinte:

- Túnel de SSH
- Exemplo de encaminhamento de porta SSH, especificamente a seção Encaminhamento local

Para criar um túnel SSH, você precisa de uma instância do Amazon EC2 em execução na mesma Amazon VPC que seu cluster do Amazon DocumentDB. É possível usar uma instância do EC2 existente na mesma VPC que seu cluster ou criar uma instância. Para obter mais informações, consulte o tópico apropriado para seu sistema operacional:

- Como iniciar com instâncias do Linux do Amazon EC2
- Como iniciar com instâncias do Windows do Amazon EC2

Normalmente, é possível se conectar a uma instância do EC2 com o seguinte comando.

```
ssh -i "ec2Access.pem" ubuntu@ec2-34-229-221-164.compute-1.amazonaws.com
```

Se esse for o caso, será possível configurar um túnel SSH para o cluster do Amazon DocumentDB `sample-cluster.node.us-east-1.docdb.amazonaws.com` executando o comando a seguir no seu computador local. O sinalizador `-L` é usado para encaminhar uma porta local. Ao usar um túnel SSH, conecte-se ao cluster usando o endpoint do cluster e não tente se conectar no modo de conjunto de réplicas (ou seja, especificando `replicaSet=rs0` na string de conexão), pois isso resultará em um erro.

```
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
```

Depois que o túnel SSH for criado, todos os comandos que você emitir para `localhost:27017` serão encaminhados para o cluster do Amazon DocumentDB `sample-cluster` em execução na Amazon VPC. Se o Transport Layer Security (TLS) estiver ativado no cluster do Amazon DocumentDB, será necessário fazer o download da chave pública para o Amazon DocumentDB em https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem . A operação a seguir faz download deste arquivo:

```
wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
```

> **Note**
>
> O TLS está habilitado por padrão para todos os novos clusters do Amazon DocumentDB. No entanto, você poderá desabilitá-lo. Para obter mais informações, consulte Gerenciar configurações de TLS de cluster do Amazon DocumentDB.

Para conectar-se ao seu cluster do Amazon DocumentDB de fora do Amazon VPC, use o comando a seguir.

```
mongo --sslAllowInvalidHostnames --ssl --sslCAFile global-bundle.pem --username
 <yourUsername> --password <yourPassword>
```

Conectar-se ao Amazon DocumentDB como um conjunto de réplicas

Ao desenvolver com base no Amazon DocumentDB, conecte-se ao seu cluster como um conjunto de réplicas e distribua leituras para instâncias de réplica usando os recursos integrados de preferência de leitura do seu driver. Esta seção aprofunda o que isso significa e descreve como você pode se conectar ao seu cluster Amazon DocumentDB como um conjunto de réplicas usando Python como exemplo.

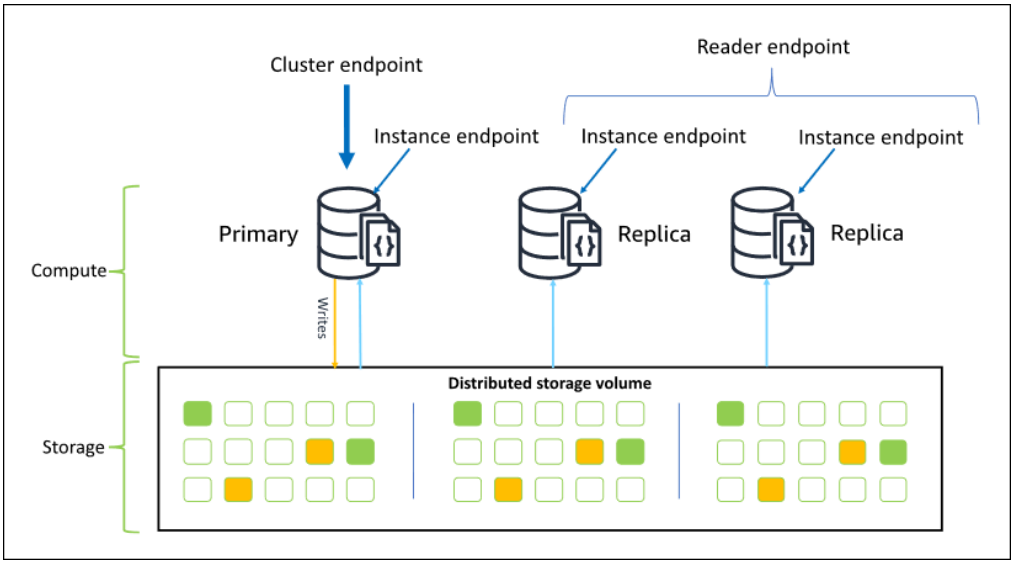
O Amazon DocumentDB tem três endpoints que podem ser usados para se conectar ao cluster:

- Endpoint do cluster
- Endpoint de leitor
- Endpoints da instância

Ao se conectar ao Amazon DocumentDB, use o endpoint do cluster. Isso é um CNAME que aponta para a instância principal no cluster, conforme mostrado no diagrama a seguir.

Ao usar um túnel SSH, conecte-se ao cluster usando o endpoint do cluster. Não conecte no modo de conjunto de réplicas (especificando `replicaSet=rs0` em sua string de conexão), pois isso resultará em um erro.

> **Note**
>
> Para obter mais informações sobre endpoints do Amazon DocumentDB, consulte Endpoints do Amazon DocumentDB.



Ao usar o endpoint do cluster, é possível se conectar ao cluster no modo de conjunto de réplicas. Depois, você poderá usar os recursos integrados do driver de preferência de leitura. No exemplo a seguir, especificar `/?replicaSet=rs0` significa para o SDK que você deseja se conectar como um conjunto de réplicas. Se você omitir `/?replicaSet=rs0'`, o cliente roteará todas as solicitações para o endpoint do cluster, ou seja, sua instância principal.

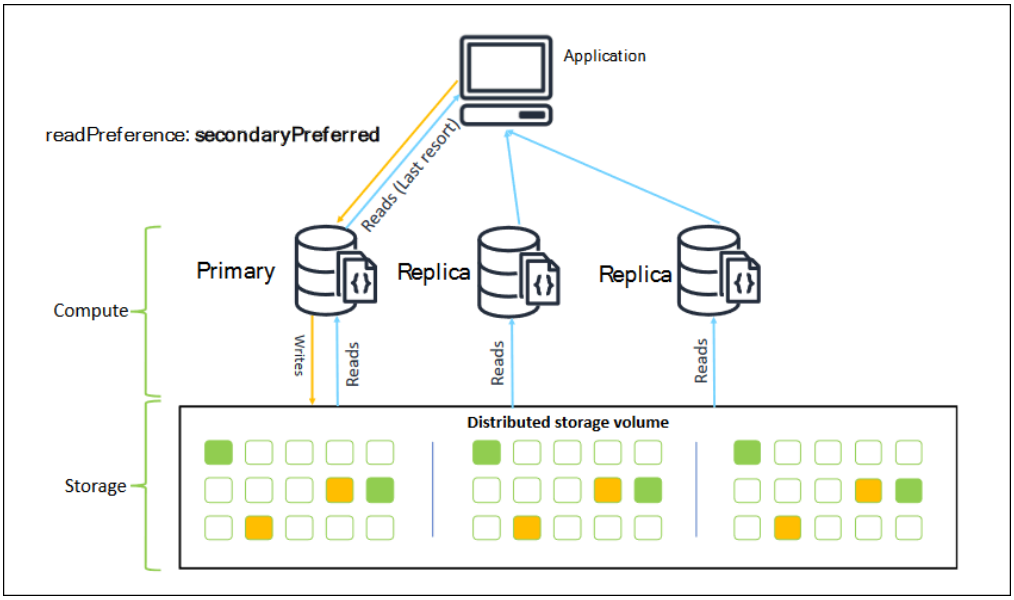
```
## Create a MongoDB client, open a connection to Amazon DocumentDB as a
   ## replica set and specify the read preference as secondary preferred
   client = pymongo.MongoClient('mongodb://<user-name>:<password>@mycluster.node.us-
east-1.docdb.amazonaws.com:27017/?replicaSet=rs0')
```

A vantagem de se conectar como um conjunto de réplicas é que isso permite que o SDK descubra a topografia do cluster automaticamente, incluindo quando as instâncias são adicionadas ou removidas do cluster. Você poderá usar seu cluster de forma mais eficiente roteando solicitações de leitura para suas instâncias de réplica.

Ao se conectar como um conjunto de réplicas, é possível especificar a `readPreference` para a conexão. Se você especificar uma preferência de leitura de `secondaryPreferred`, o cliente roteará as consultas de leitura para suas réplicas e as consultas de gravação para sua instância principal (como no diagrama a seguir). Esse é um uso melhor dos recursos do cluster. Para obter mais informações, consulte Opções de preferência de leitura.

```
## 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')
```



As leituras de réplicas do Amazon DocumentDB são eventualmente consistentes. Elas retornam os dados na mesma ordem em que foram gravados na instância principal, e geralmente há menos de 50 ms de tempo de atraso de replicação. Você pode monitorar o atraso da réplica do seu cluster usando as CloudWatch métricas da Amazon e. `DBInstanceReplicaLag` `DBClusterReplicaLagMaximum` Para obter mais informações, consulte Monitorando o Amazon DocumentDB com CloudWatch.

Diferentemente da arquitetura de banco de dados monolítica tradicional, o Amazon DocumentDB separa o armazenamento e a computação. Considerando essa arquitetura moderna, recomendamos que você faça a escalabilidade de leitura nas instâncias de réplica. As leituras nas instâncias de réplica não bloqueiam as gravações que são replicadas a partir da instância principal. É possível adicionar até 15 instâncias de réplica de leitura em um cluster e expandir para milhões de leituras por segundo.

O principal benefício de se conectar como um conjunto de réplicas e distribuir leituras para réplicas é que ele aumenta os recursos gerais em seu cluster que estão disponíveis para trabalhar em sua aplicação. Como prática recomendada, conecte-se como um conjunto de réplicas. Isso é mais benéfico nos seguintes cenários:

- Você está usando quase 100% de CPU na principal.
- A proporção de acertos do cache em buffer é próxima de zero.

- Você atinge os limites de conexão ou de cursor para uma instância individual.

Expandir o tamanho de uma instância de cluster é uma opção e, em alguns casos, essa pode ser a melhor maneira de escalar o cluster. Mas você também deve considerar como usar melhor as réplicas que já tem em seu cluster. Isso permite aumentar a escala sem o aumento do custo de usar um tipo de instância maior. Também recomendamos que você monitore e alerte sobre esses limites (ou seja `CPUUtilization`, `DatabaseConnections`, `eBufferCacheHitRatio`) usando CloudWatch alarmes para saber quando um recurso está sendo muito usado.

Para saber mais, consulte os seguintes tópicos:

- Práticas recomendadas do Amazon DocumentDB
- Cotas do Amazon DocumentDB

Usar conexões de cluster

Considere o cenário de uso de todas as conexões em seu cluster. Por exemplo, uma instância `r5.2xlarge` tem um limite de 4.500 conexões (e 450 cursores abertos). Se você criar um cluster do Amazon DocumentDB de três instâncias e se conectar somente à instância principal usando o endpoint do cluster, os limites do cluster para conexões abertas e cursores serão 4.500 e 450, respectivamente. Talvez você atinja esses limites se estiver criando aplicações que usem muitos operadores que sejam configurados em contêineres. Os contêineres abrem várias conexões de uma só vez e saturam o cluster.

Em vez disso, é possível se conectar ao cluster do Amazon DocumentDB como um conjunto de réplicas e distribuir suas leituras para as instâncias de réplica. Depois, é possível efetivamente triplicar o número de conexões e cursores disponíveis no cluster para 13.500 e 1.350, respectivamente. Adicionar mais instâncias ao cluster só aumentará o número de conexões e cursores para workloads de leitura. Se for necessário aumentar o número de conexões para gravações em seu cluster, recomendamos aumentar o tamanho da instância.

> **Note**
>
> O número de conexões para instâncias `large`, `xlarge` e `2xlarge` aumenta com o tamanho da instância, chegando até 4.500. O número máximo de conexões por instância para instâncias `4xlarge` ou maiores é 4.500. Para obter mais informações sobre limites por tipos de instância, consulte Cotas de instâncias.

Normalmente, não recomendamos que você se conecte ao cluster usando a preferência de leitura de `secondary`. Isso ocorre porque, se não houver instâncias de réplica no cluster, haverá falha nas leituras. Por exemplo, suponha que você tenha um cluster do Amazon DocumentDB de duas instâncias com uma principal e uma de réplica. Se a instância de réplica tiver um problema, haverá falha nas solicitações de leitura de um grupo de conexão que esteja definido como `secondary`. A vantagem de `secondaryPreferred` é que, se o cliente não conseguir encontrar uma instância de réplica adequada à qual se conectar, ele voltará para a principal para leituras.

Vários grupos de conexões

Em alguns cenários, as leituras em uma aplicação devem ter consistência de leitura após gravação, que pode ser atendida somente a partir da instância principal no Amazon DocumentDB. Nesses cenários, você pode criar dois grupos de conexão de cliente: um para gravações e outro para leituras que precisam de consistência de leitura após gravação. Para fazer isso, seu código deve ser semelhante ao seguinte:

```
## 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')
```

Outra opção é criar um único grupo de conexões e substituir a preferência de leitura para uma determinada coleção.

```
##Specify the collection and set the read preference level for that collection
col = db.review.with_options(read_preference=ReadPreference.SECONDARY_PREFERRED)
```

Resumo

Para usar melhor os recursos em seu cluster, conecte-se usando o modo de conjunto de réplicas. Se for adequado para sua aplicação, você poderá fazer a escalabilidade de leitura de sua aplicação distribuindo suas leituras para as instâncias de réplica.

Conectar a um cluster Amazon DocumentDB a partir do Studio 3T

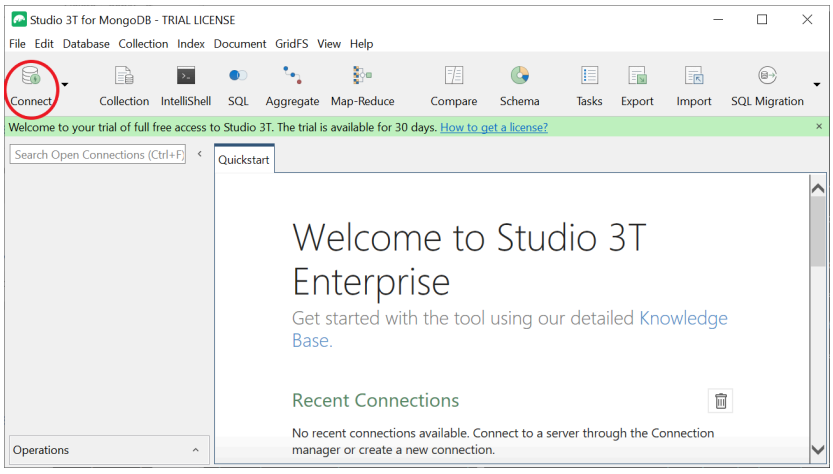
O Studio 3T é uma GUI e IDE popular para desenvolvedores e engenheiros de dados que trabalham com o MongoDB. Ele oferece vários recursos poderosos: visualizações em árvore, tabela e JSON de seus dados, fáceis import/export em CSV, JSON, SQL e, opção de consulta flexível BSON/ mongodump, uma interface visual de arrastar e soltar, um shell mongo integrado com preenchimento automático, um editor de pipeline de agregação e suporte a consultas SQL.

Pré-requisitos

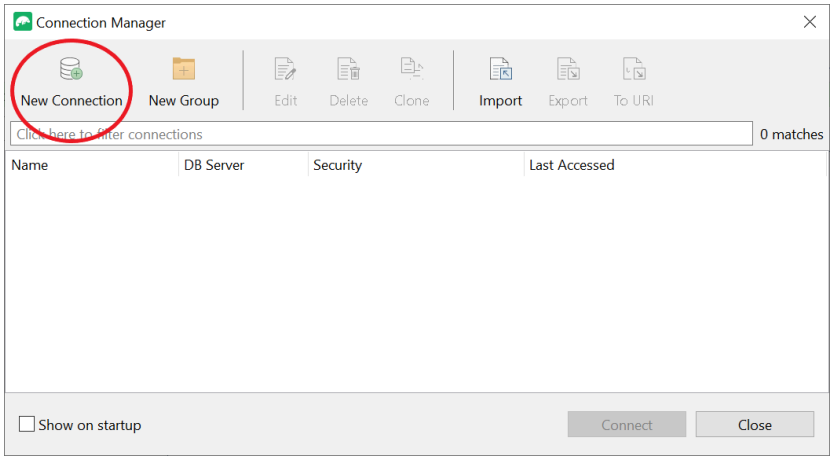
- Se você ainda não tem um cluster Amazon DocumentDB usando o Amazon EC2 como bastion/ jump host, consulte. Conecte usando o Amazon EC2
- Se você não tem o Studio 3T, faça o download e instale.

Conectar com o Studio 3T

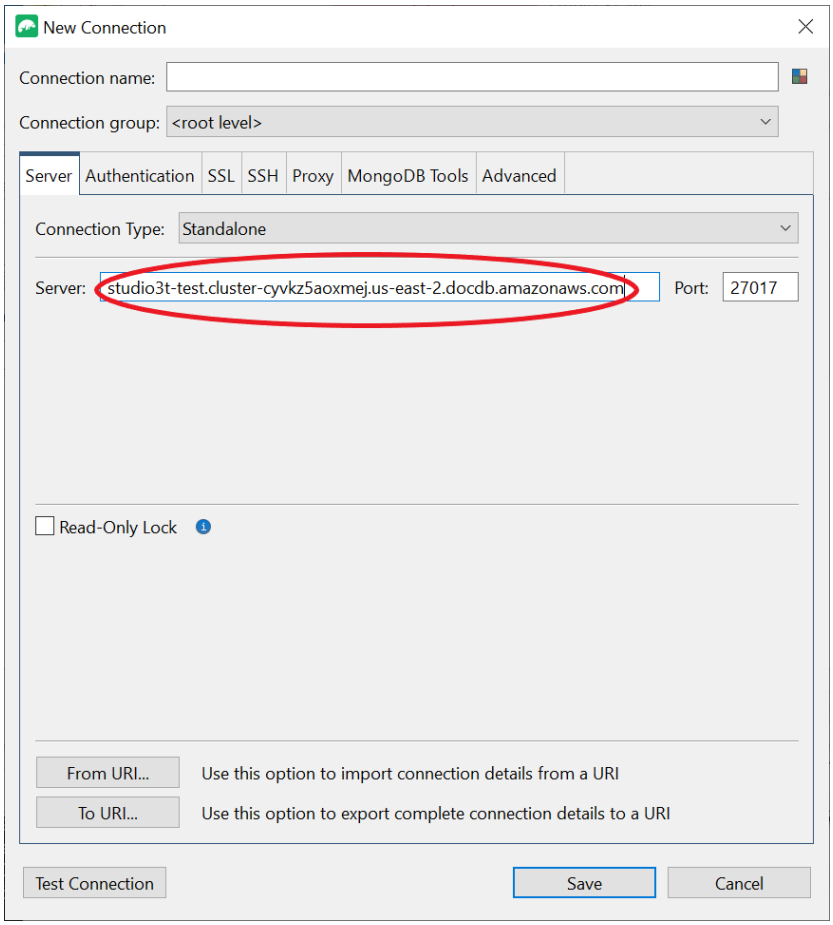
1. Escolha Conectar no canto superior esquerdo da barra de ferramentas.



2. Escolha Nova conexão no canto superior esquerdo da barra de ferramentas.

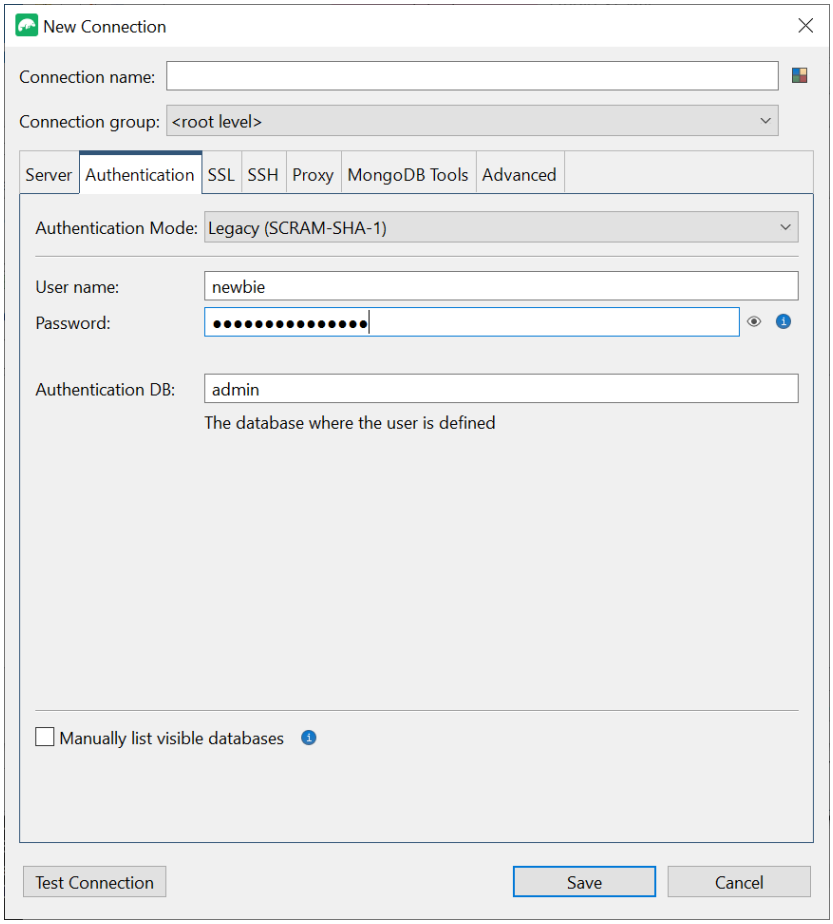


3. Na guia Servidor, no campo Servidor, digite as informações do endpoint do cluster.

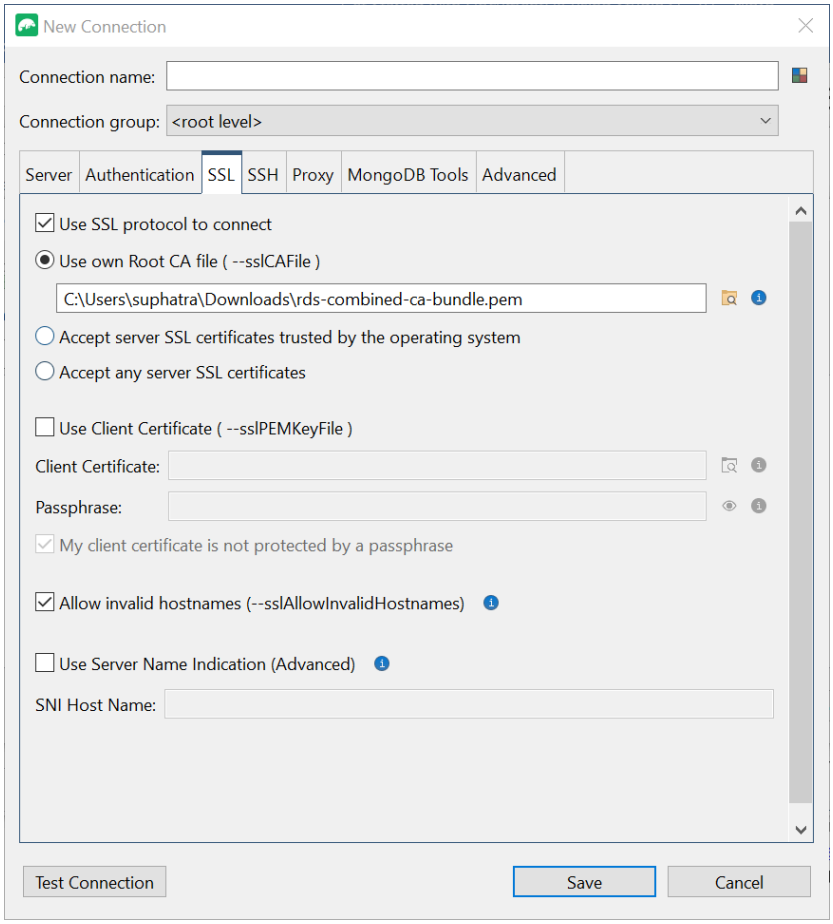


> **Note**
>
> Para encontrar seu endpoint de cluster, consulte Localizar o endpoint de uma instância.

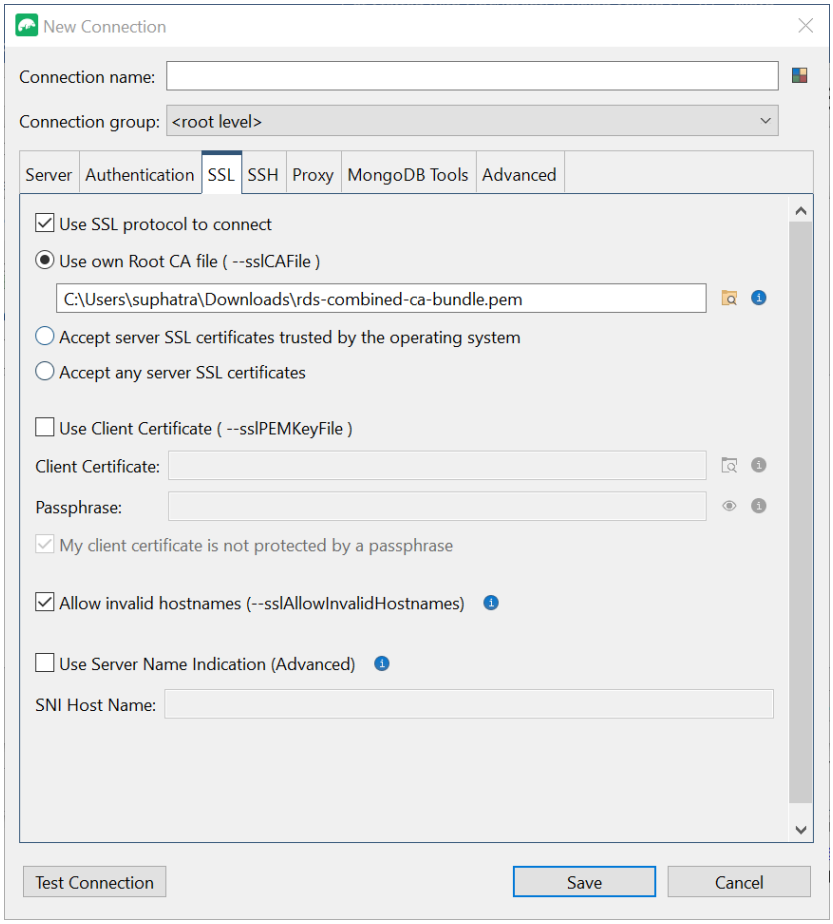
4. Escolha a guia Autenticação e selecione Legado no menu suspenso do Modo de Autenticação.



5. Insira seu nome de usuário e credenciais nos campos Nome de usuário e Senha.
6. Escolha a guia SSL e marque a caixa Usar protocolo SSL para conectar.



7. Escolha Usar arquivo CA raiz próprio. Em seguida, adicione o certificado Amazon DocumentDB (você pode pular essa etapa se o SSL estiver desabilitado no seu cluster do DocumentDB). Marque a caixa para permitir nomes de host inválidos.



> **Note**
>
> Não tem o certificado? É possível baixar com o seguinte comando:
> `wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem`

8. Se estiver se conectando a partir de uma máquina externa à Amazon VPC, você precisará criar um túnel SSH. Você fará isso na guia SSH.

   a. Marque a caixa Usar túnel SSH e insira o endereço SSH no campo Endereço SSH. Essa é a sua instância DNS pública (IPV4). É possível obter essa URL no Console Amazon EC2 Management.

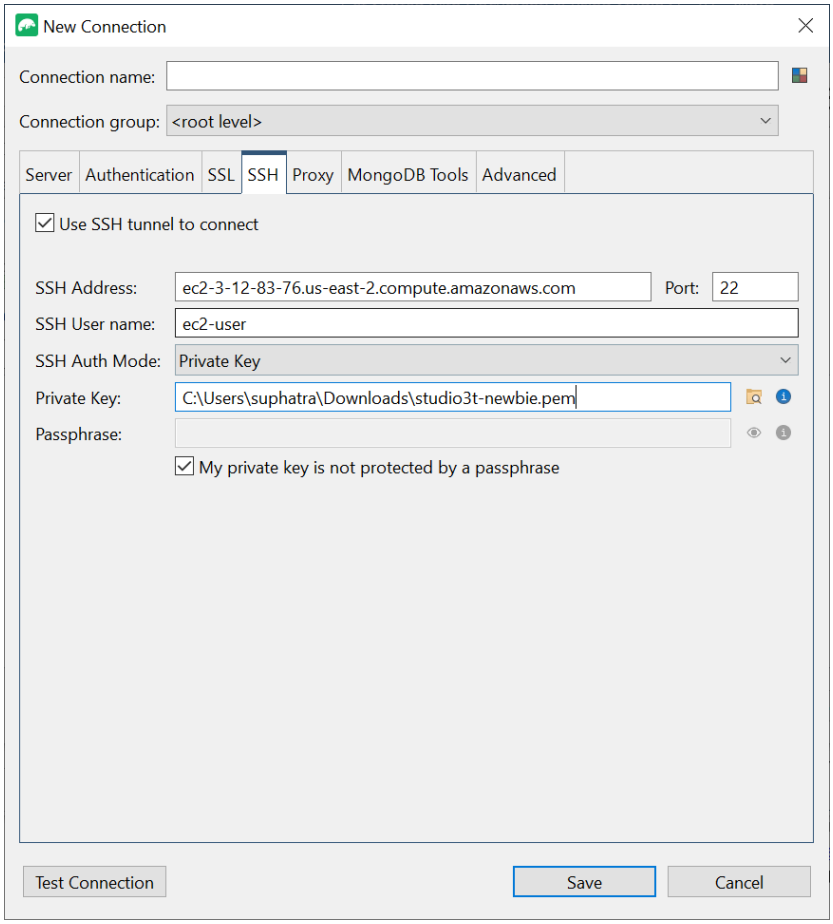
   b. Insira seu nome de usuário. Este é o nome de usuário da instância do Amazon EC2

   c. Para o Modo de Autenticação SSH, selecione Chave privada. No campo Chave privada, escolha o ícone do localizador de arquivos para localizar e selecionar a chave privada da

sua instância do Amazon EC2. Esse é o arquivo .pem (par de chave) que você salvou ao criar sua instância no console do Amazon EC2.

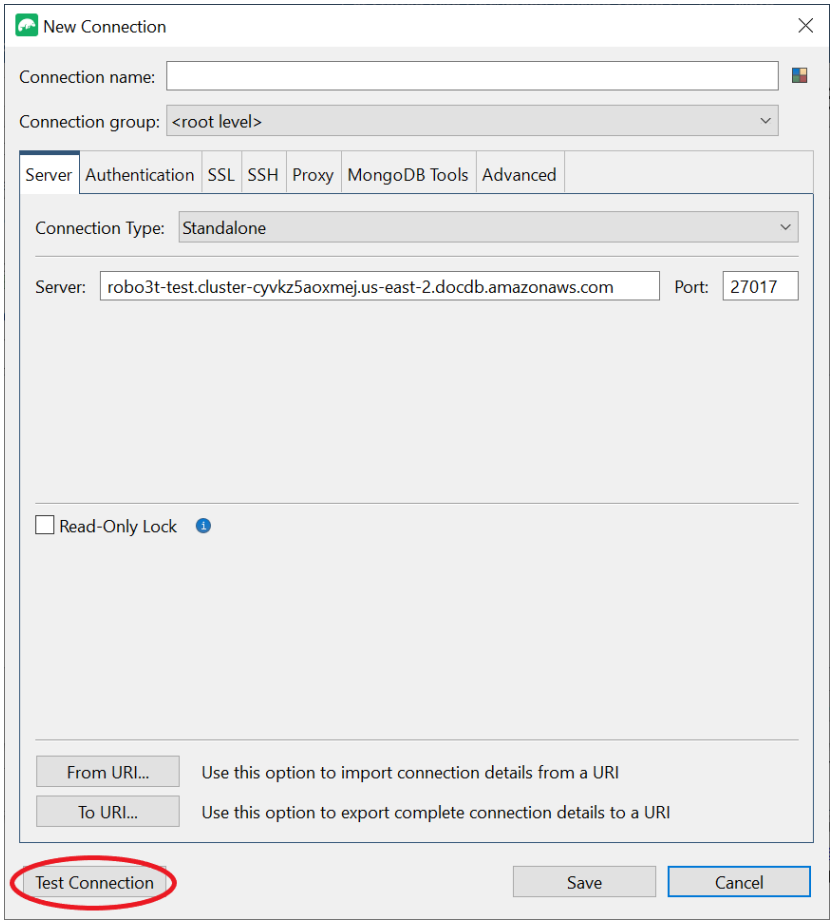
d. Se você estiver na máquina Linux/macOS cliente, talvez seja necessário alterar as permissões da sua chave privada usando o seguinte comando:

```
chmod 400 /fullPathToYourPemFile/<yourKey>.pem
```

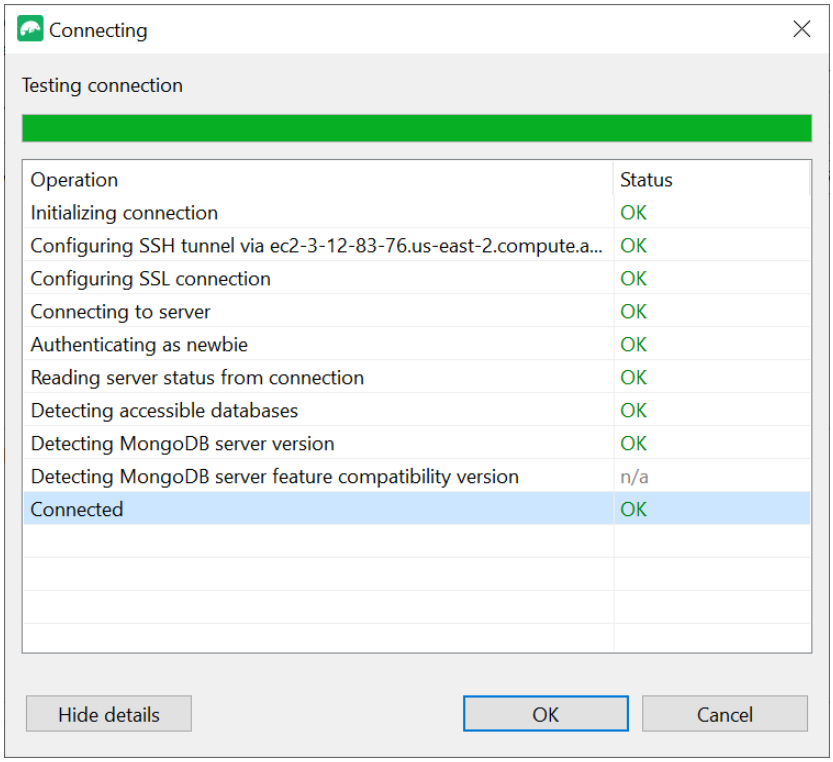


> **Note**
>
> Essa instância do Amazon EC2 deve estar na mesma Amazon VPC e grupo de segurança que o cluster do DocumentDB. É possível obter o endereço SSH, o nome de usuário e a chave privada no Console de Gerenciamento do Amazon EC2.

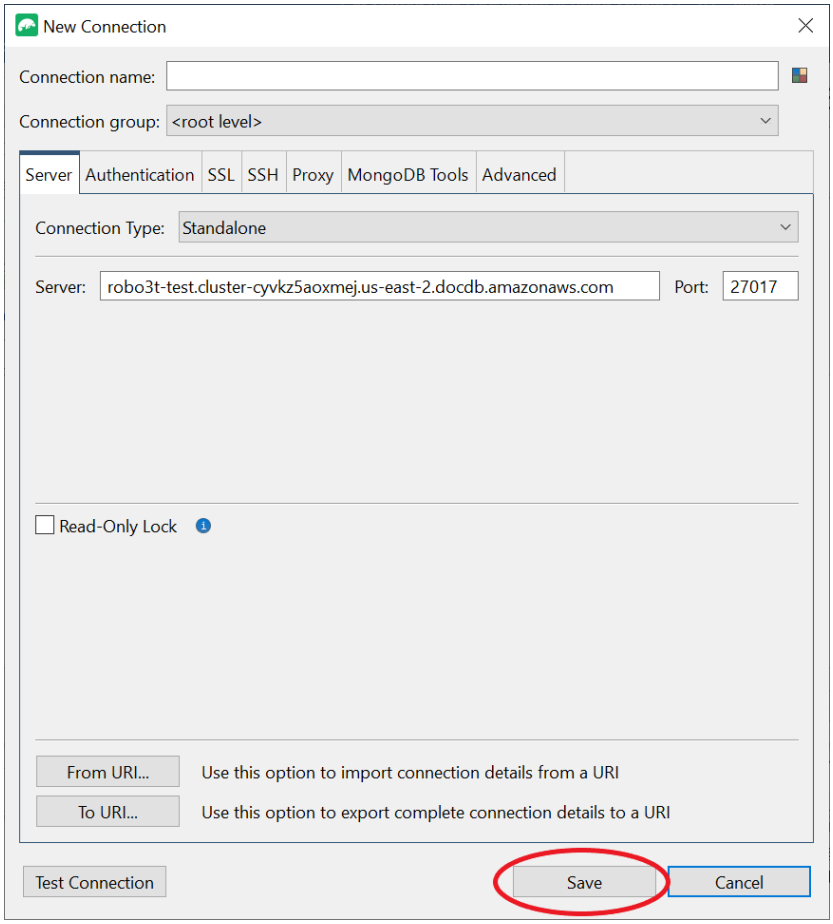
9. Agora teste sua configuração escolhendo o botão Testar conexão.



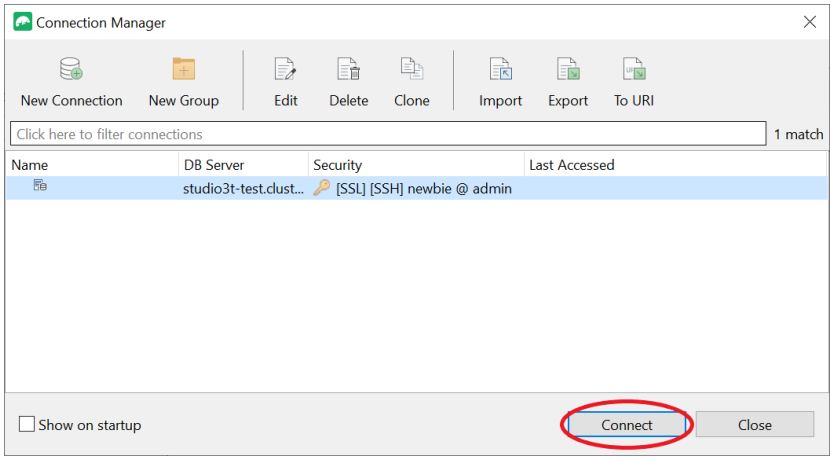
10. Uma janela de diagnóstico deve carregar uma barra verde para indicar que o teste foi bem-sucedido. Agora escolha OK para fechar a janela de diagnóstico.



11. Escolha Salavar para salvar a conexão para uso futuro.



12. Agora selecione seu cluster e escolha Conectar.



Parabéns! Agora você está conectado ao seu cluster Amazon DocumentDB por meio do Studio 3T.

Conecte-se ao Amazon DocumentDB usando DataGrip

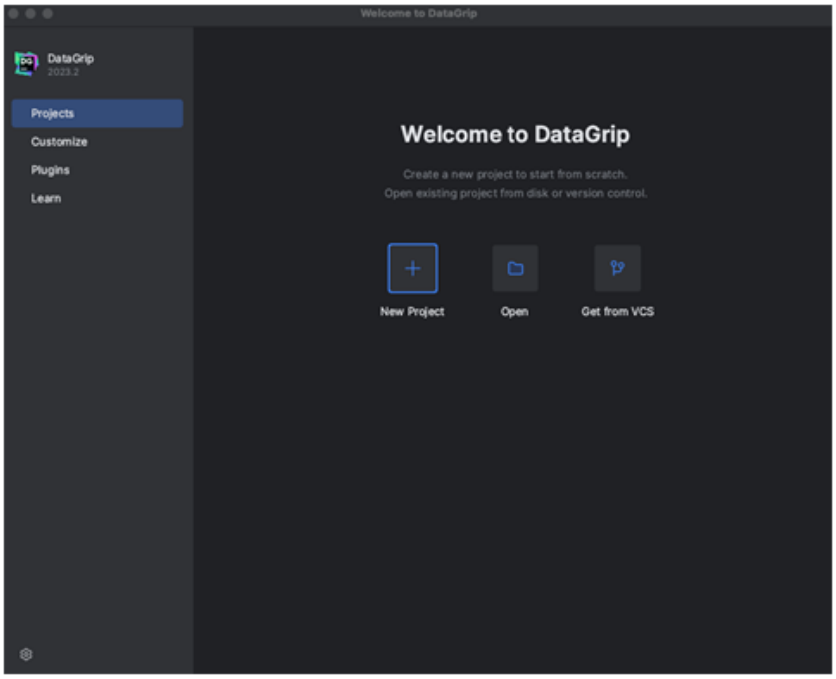
DataGripé um poderoso ambiente de desenvolvimento integrado (IDE) que oferece suporte a vários sistemas de banco de dados, incluindo o Amazon DocumentDB. Esta seção mostra as etapas para se conectar ao seu cluster Amazon DocumentDB usando DataGrip, permitindo que você gerencie e consulte facilmente seus dados usando uma interface gráfica.

Pré-requisitos

- DataGrip IDE instalado em sua máquina. Você pode baixá-lo em JetBrains.
- Uma instância do Amazon EC2 em execução na mesma VPC que o seu cluster do Amazon DocumentDB. Você usará essa instância para estabelecer um túnel seguro da sua máquina local para o Amazon DocumentDBCluster. Siga estas instruções para saber como Conecte usando o Amazon EC2.
- Alternativa para uma instância do Amazon EC2, uma conexão VPN ou se você já estiver acessando sua AWS infraestrutura usando uma VPN segura. Se você preferir essa opção, siga as instruções para acessar com segurança o Amazon DocumentDB usando. AWS Client VPN

Conecte-se usando DataGrip

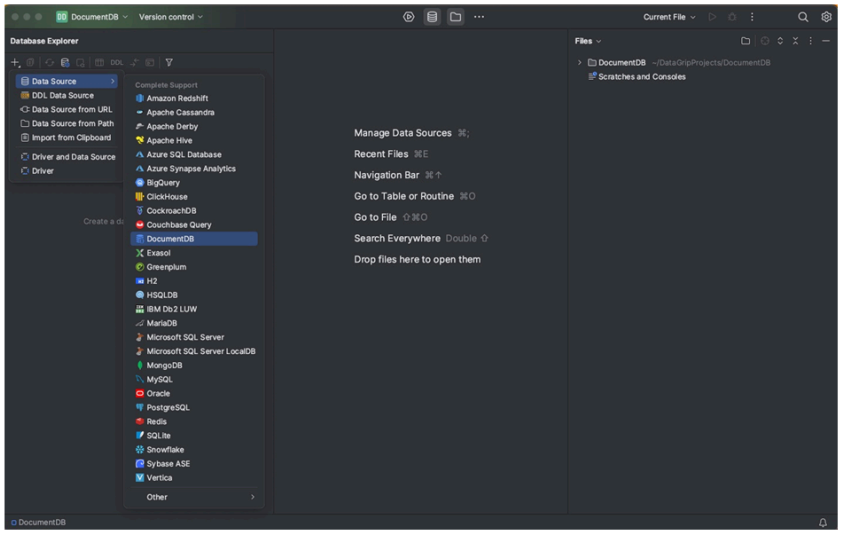
1. Inicie DataGrip no seu computador e crie um novo projeto.



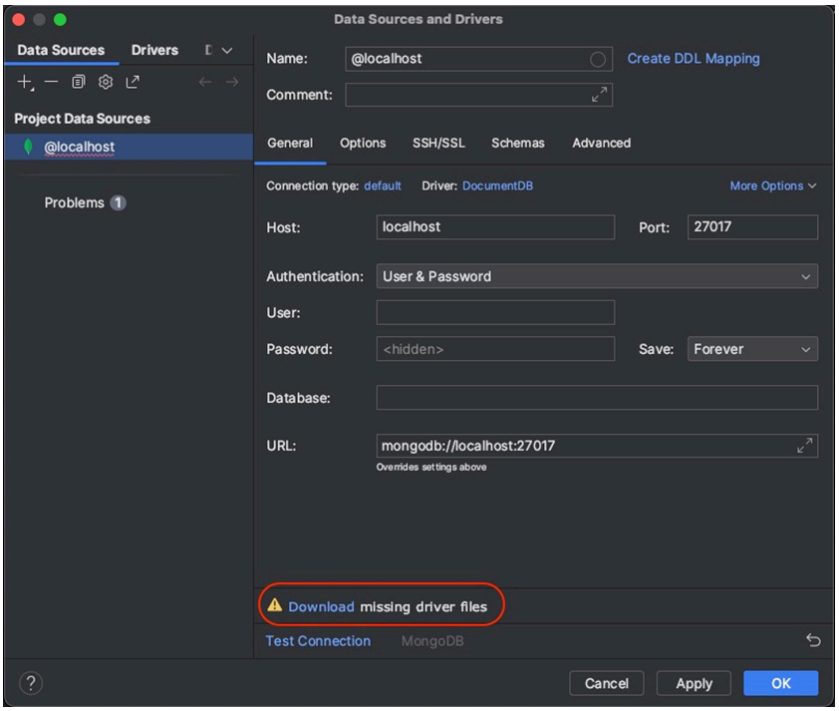
2. Adicione uma nova fonte de dados usando uma das seguintes maneiras:

a. No menu principal, navegue até Arquivo — Novo — Fonte de dados e selecione DocumentDB

b. No Database Explorer, escolha o novo ícone (+) na barra de ferramentas. Navegue até Fonte de dados e selecione DocumentDB.



3. Na página Fontes de dados na guia Geral, verifique se há um link para Fazer o download dos arquivos de driver ausentes na parte inferior da área de configurações da conexão. Escolha esse link para baixar os drivers necessários para interagir com um banco de dados. Para obter um link de download direto, consulte os drivers JetBrains JDBC.



4. Na guia Geral, especifique os detalhes da conexão:

a. No campo Host, especifique o endpoint do cluster Amazon DocumentDB.

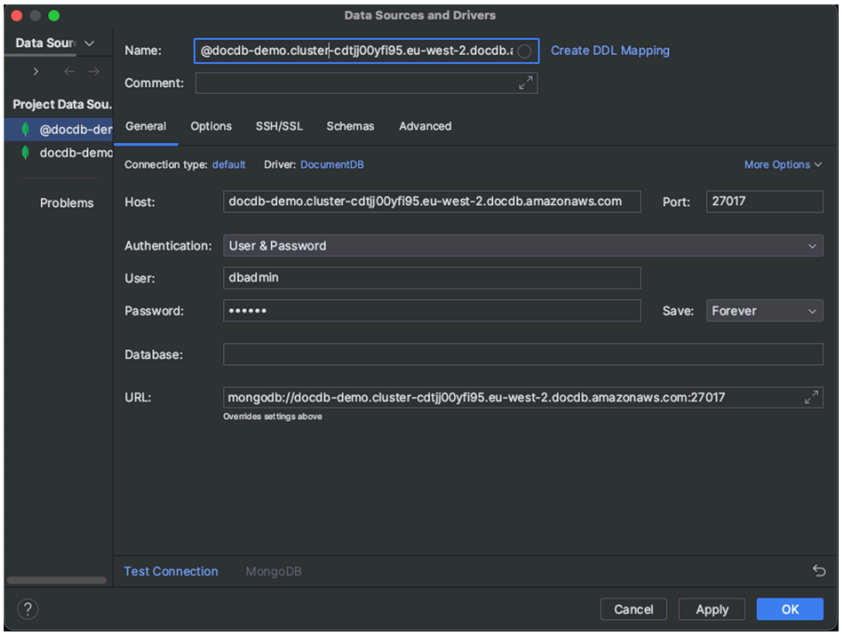
b. A porta já está definida como 27017. Altere-a se seu cluster foi implantado em uma porta diferente.

c. Em Método de autenticação, escolha Nome e senha do usuário.

d. Insira as informações de nome de usuário e senha.

e. O campo Banco de dados é opcional. É possível especificar o banco de dados ao qual deseja conectar-se.

f. O campo URL é preenchido automaticamente à medida que você adiciona os detalhes anteriores.



5. Na SSH/SSLguia, habilite Usar túnel SSH e escolha o ícone para abrir a caixa de diálogo Configuração SSH. Insira as seguintes informações:

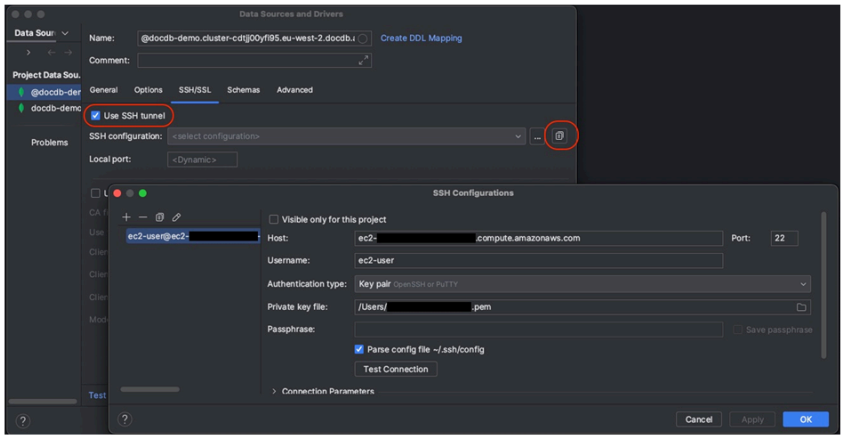
a. No campo Host, insira o nome do host da sua instância do Amazon EC2.

b. Insira o nome de usuário e senha da sua instância do Amazon EC2.

c. Em Tipo de autenticação, escolha Par de chave.

d. Insira o arquivo de chave privada.

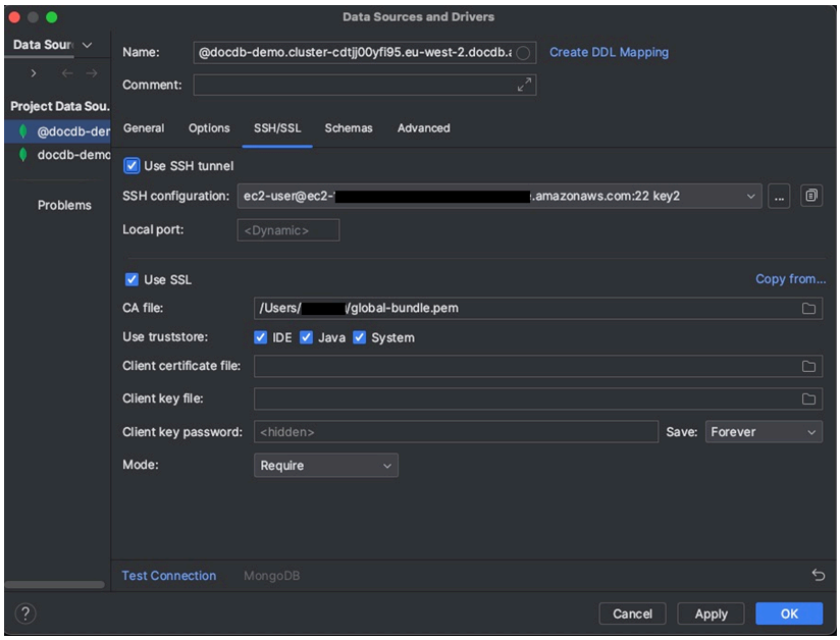
> **Note**
>
> Se você estiver usando a opção VPN, não há necessidade de configurar o túnel SSH.



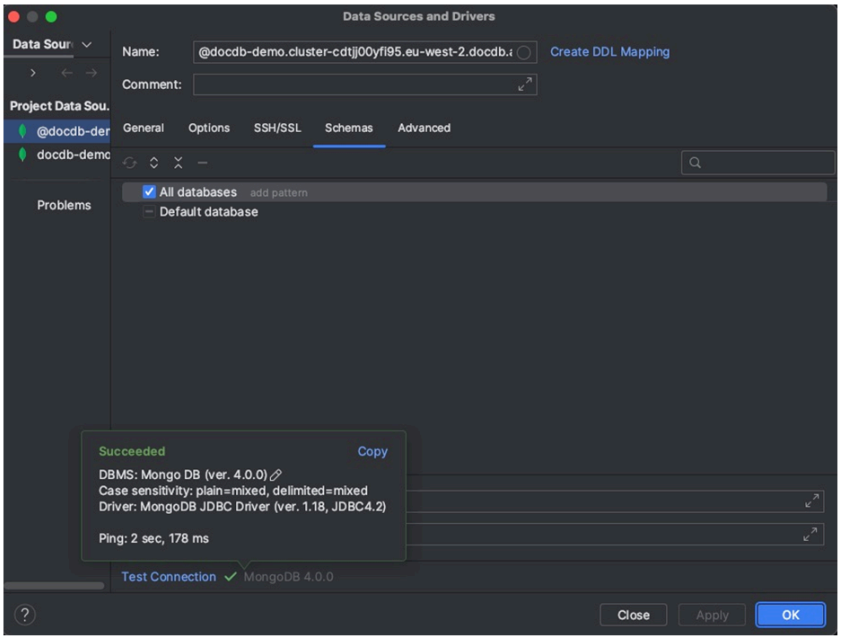
6. Na SSH/SSLguia, habilite Usar SSL. No campo Arquivo CA, insira o local do arquivo `global-bundle.pem` no seu computador. Em Modo, deixe a opção Exigir como está.

> **Note**
>
> É possível fazer o download do certificado a partir deste local ou com este comando: wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem

> **Note**
>
> Se você estiver tentando conectar-se ao cluster elástico Amazon DocumentDB, não precisará especificar o arquivo CA. Deixe a opção Usar SSL, bem como todas as outras opções, marcada em seus valores padrão.



7. Na guia Esquemas, escolha Todos os bancos de dados ou insira o filtro “*: *” no campo Padrão do esquema. Escolha Testar conexão para testar a conexão.



8. Depois que a conexão for testada com sucesso, escolha OK para salvar a configuração da fonte de dados.

DataGrip features

DataGrip fornece vários recursos para ajudar você a trabalhar com o Amazon DocumentDB de forma eficiente:

- Editor SQL — Escreva e execute SQL-like consultas em suas coleções do DocumentDB usando o editor SQL em. DataGrip
- Visual Query Builder — Use o Visual Query Builder para criar consultas graficamente sem gravar código SQL.
- Gerenciamento de esquemas — Gerencie facilmente seu esquema de banco de dados, inclusive a criação, alteração e eliminação de coleções.
- Visualização de dados — Visualize e analise seus dados usando várias ferramentas de visualização disponíveis em. DataGrip
- Exportar e importar dados — Transfira dados entre o Amazon DocumentDB e outros bancos de dados usando os recursos DataGrip de exportação e importação do Amazon.

Consulte a DataGrip documentação oficial para obter recursos e dicas mais avançados sobre como trabalhar com o Amazon DocumentDB e outros sistemas de banco de dados.

Conecte usando o Amazon EC2

Esta seção descreve como configurar a conectividade entre um cluster do Amazon DocumentDB e o Amazon EC2 e acessar o cluster do Amazon DocumentDB a partir da instância do Amazon EC2.

Há duas opções para configurar a conexão do EC2:

- Conectar automaticamente a instância do EC2 a um banco de dados do Amazon DocumentDB — use o recurso de conexão automática no console do EC2 para configurar automaticamente a conexão entre a instância do EC2 e um banco de dados do Amazon DocumentDB novo ou existente. Essa conexão permite que haja tráfego entre a instância do EC2 e o banco de dados do Amazon DocumentDB. Essa opção geralmente é usada para testar e criar novos grupos de segurança.
- Conectar manualmente a instância do EC2 ao banco de dados do Amazon DocumentDB — configure a conexão entre a instância do EC2 e o banco de dados do Amazon DocumentDB configurando e atribuindo manualmente os grupos de segurança para reproduzir a configuração criada pelo recurso de conexão automática. Essa opção geralmente é usada para alterar configurações mais avançadas e usar grupos de segurança existentes.

Pré-requisitos

Independentemente da opção e antes de criar o primeiro cluster do Amazon DocumentDB, você deve fazer o seguinte:

Crie uma conta (AWS) da Amazon Web Services

Antes de começar a usar o Amazon DocumentDB, você deve ter uma conta da Amazon Web Services (AWS). A AWS conta é gratuita. Você paga apenas pelos serviços e recursos usados.

Se você não tiver um Conta da AWS, conclua as etapas a seguir para criar um.

Para se inscrever em um Conta da AWS

1. Aberto https://signin.aws.amazon.com/signup? request_type=registrar.
2. Siga as instruções online.

   Parte do procedimento de inscrição envolve receber uma chamada telefônica ou uma mensagem de texto e inserir um código de verificação pelo teclado do telefone.

   Quando você se inscreve em um Conta da AWS, um Usuário raiz da conta da AWS é criado. O usuário-raiz tem acesso a todos os Serviços da AWS e recursos na conta. Como prática recomendada de segurança, atribua o acesso administrativo a um usuário e use somente o usuário-raiz para executar tarefas que exigem acesso de usuário-raiz.

Opcionalmente, configure as permissões necessárias AWS Identity and Access Management (IAM).

O acesso para gerenciar recursos do Amazon DocumentDB, como clusters, instâncias e grupos de parâmetros de cluster, exige credenciais que AWS possam ser usadas para autenticar suas solicitações. Para obter mais informações, consulte Gerenciamento de identidade e Gerenciamento de acesso para o Amazon DocumentDB.

1. Na barra de pesquisa do Console de gerenciamento da AWS, insira **`IAM`** e selecione IAM no menu suspenso exibido.
2. Depois de chegar ao console do IAM, selecione Usuários no painel de navegação.
3. Selecione o nome de usuário.
4. Escolha Adicionar permissões.
5. Selecione Attach existing policies directly (Anexar políticas existentes diretamente).

6. Digite `AmazonDocDBFullAccess` na barra de pesquisa e selecione-a quando ela aparecer nos resultados da pesquisa.
7. Escolha Próximo: revisar.
8. Escolha Adicionar permissões.

Como criar uma Amazon Virtual Private Cloud (Amazon VPC)

Dependendo de onde Região da AWS você estiver, você pode ou não ter uma VPC padrão já criada. Se você não tiver uma VPC padrão, conclua a etapa 1 de Conceitos básicos da Amazon VPC no Guia do usuário da Amazon VPC. Isso levará menos de cinco minutos.

Conectar o Amazon EC2 automaticamente

Tópicos

- Conectar automaticamente uma instância do EC2 a um novo banco de dados do Amazon DocumentDB
- Conectar automaticamente uma instância do EC2 a um banco de dados do Amazon DocumentDB existente
- Visão geral da conectividade automática com uma instância do EC2
- Visualizar recursos computacionais conectados

Antes de configurar uma conexão entre uma instância do EC2 e um novo banco de dados do Amazon DocumentDB, verifique se você atende aos requisitos descritos em Visão geral da conectividade automática com uma instância do EC2. Se você alterar esses grupos de segurança depois de configurar a conectividade, as alterações poderão afetar a conexão entre a instância do EC2 e o banco de dados do Amazon DocumentDB.

> **Note**
>
> Você só pode configurar automaticamente uma conexão entre uma instância do EC2 e um banco de dados do Amazon DocumentDB utilizando o Console de gerenciamento da AWS. Você não pode configurar uma conexão automaticamente com a API AWS CLI ou com a API Amazon DocumentDB.

Conectar automaticamente uma instância do EC2 a um novo banco de dados do Amazon DocumentDB

O processo a seguir pressupõe que você concluiu as etapas do tópico Pré-requisitos.

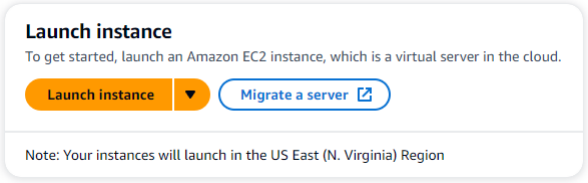
Etapas

- Etapa 1: criar uma instância do Amazon EC2
- Etapa 2: criar um cluster do Amazon DocumentDB
- Etapa 3: conectar-se à instância do Amazon EC2
- Etapa 4: instalação do shell do MongoDB
- Etapa 5: gerenciar o Amazon DocumentDB TLS
- Etapa 6: conectar-se ao cluster do Amazon DocumentDB
- Etapa 7: inserir e consultar dados
- Etapa 8: explorar

Etapa 1: criar uma instância do Amazon EC2

Nesta etapa, você criará uma instância do Amazon EC2 na mesma região e na Amazon VPC que você usará posteriormente para provisionar seu cluster Amazon DocumentDB.

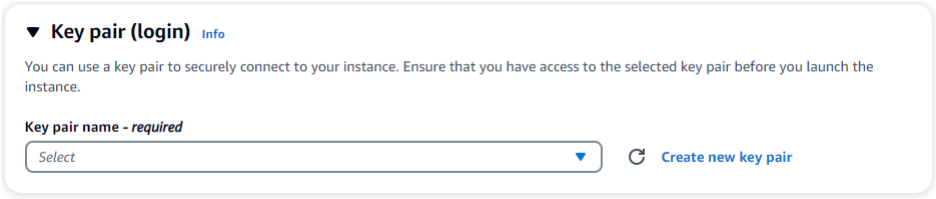
1. No console do Amazon EC2, selecione Iniciar instância.



2. Insira um nome ou identificador no campo Nome, localizado na seção Nome e tags.
3. Na lista suspensa Imagem de máquina da Amazon (AMI), localize a AMI do Amazon Linux 2 e escolha-a.

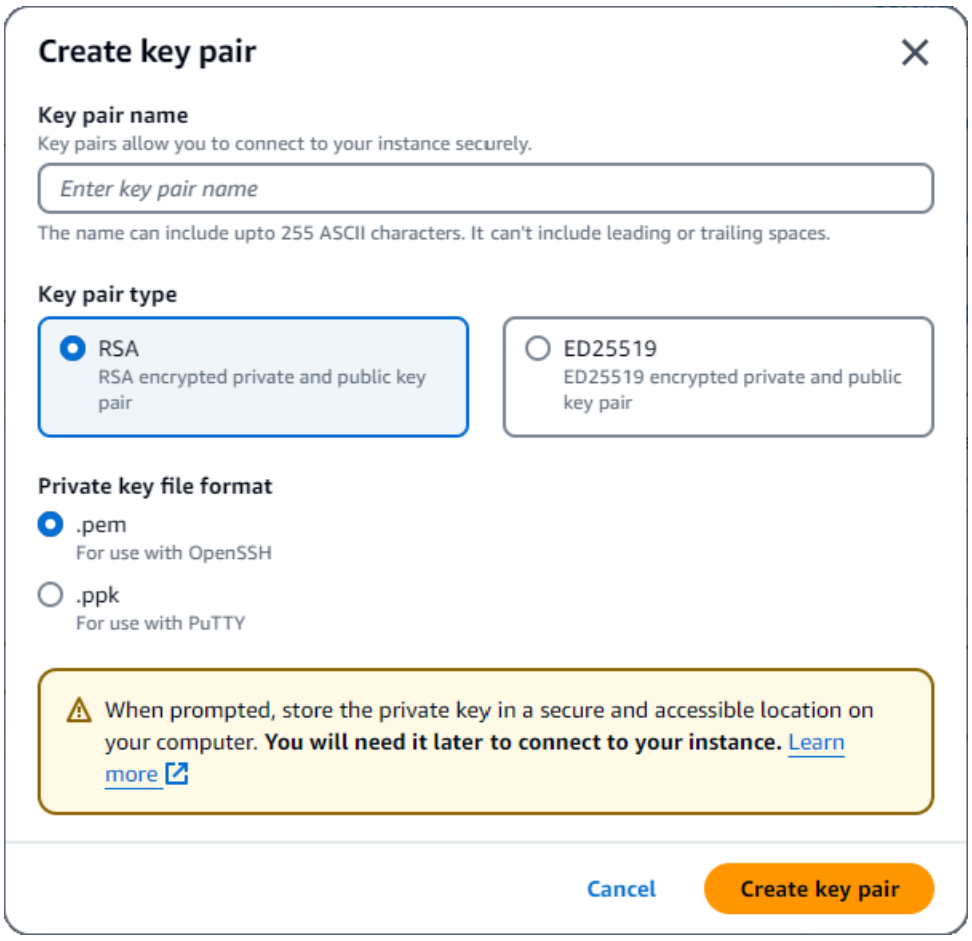


4. Localize e escolha t3.micro na lista suspensa Tipo de instância.
5. Na seção Par de chaves (login) insira o identificador de um par de chaves existente ou selecione Criar um novo par de chaves.



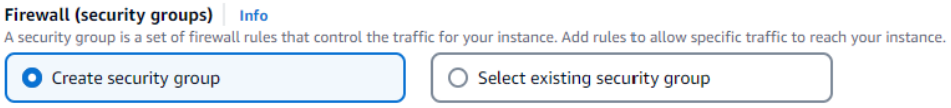
É necessário fornecer um par de chaves do Amazon EC2.

- Se você tiver um par de chaves do Amazon EC2:
  1. Selecione um par de chaves, escolha seu par de chaves na lista.
  2. Você já deve ter o arquivo de chave privada (arquivo .pem ou .ppk) disponível para fazer login na instância do Amazon EC2.
- Se você não tiver um par de chaves do Amazon EC2:
  1. Ao escolher Criar novo par de chaves, a caixa de diálogo Criar par de chaves é exibida.
  2. Insira um nome no campo Nome do par de chaves.
  3. Escolha o Tipo de par de chaves e o Formato de arquivo de chave privada.
  4. Escolha Create key pair (Criar par de chaves).



> **Note**
>
> Para fins de segurança, é altamente recomendável usar um par de chaves para conectividade de SSH e de internet com sua instância do EC2.

6. Opcional: na seção Configurações de rede, em Firewall (grupos de segurança), escolha Criar grupo de segurança.



Escolha Criar grupo de segurança (verifique todas as regras de permissão de tráfego que se apliquem à sua conectividade do EC2).

> **Note**
>
> Se você quiser usar um grupo de segurança existente, siga as instruções em Conectar o Amazon EC2 manualmente.

7. Na seção Resumo, revise sua configuração do EC2 e escolha Iniciar instância, se tudo estiver correto.

Etapa 2: criar um cluster do Amazon DocumentDB

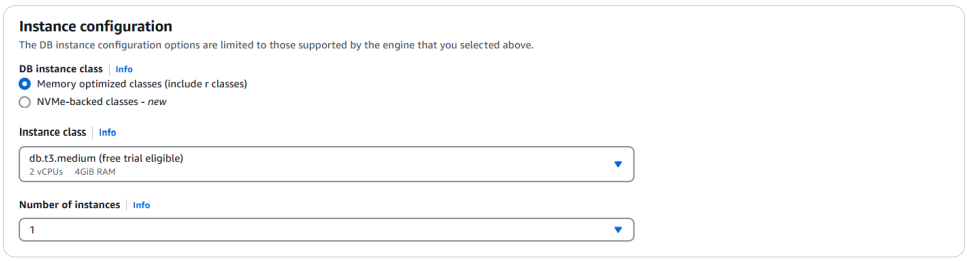
Enquanto a instância do Amazon EC2 estiver sendo provisionada, crie seu cluster do Amazon DocumentDB.

1. Navegue até o console do Amazon DocumentDB e escolha Clusters no painel de navegação.
2. Escolha Criar.
3. Deixe a configuração Tipo de cluster com o valor padrão, Cluster baseado em instância.
4. Em Configuração do cluster, em Identificador do cluster, insira um nome exclusivo. Observe que o console mudará todos os nomes de cluster para minúsculas, independentemente de como eles sejam inseridos.

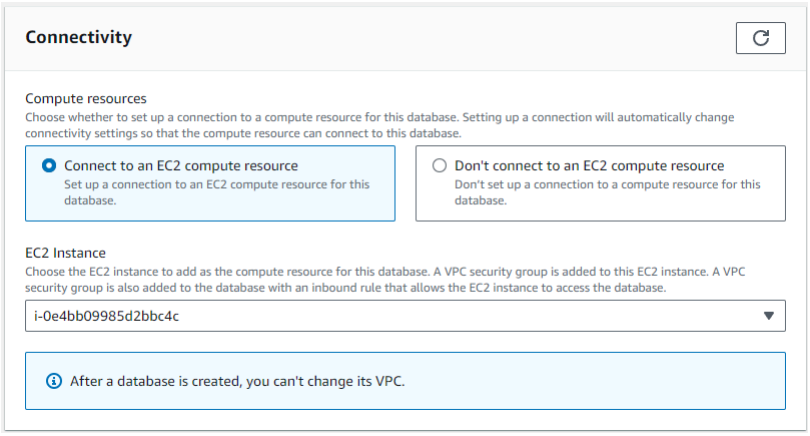
   Deixe a Versão do mecanismo com o valor padrão de 5.0.0.
5. Em Configuração de armazenamento em cluster, deixe a configuração padrão Amazon DocumentDB Padrão.
6. Em Configuração da instância:

   - Em Classe de instância de banco de dados, escolha Classes otimizadas para memória (incluir classes r) (esse é o padrão).

     A outra opção de instância são NVMe-backed classes. Para saber mais, consulte NVMe-backed instâncias.
   - Em Classe de instância, escolha o tipo de instância que atende às suas necessidades. Para obter uma explicação mais detalhada das classes de instância, consulte Especificações de classe de instância.
   - Em número de casos, escolha um número que melhor reflita suas necessidades. Lembre-se de que quanto menor o número, menor o custo e menor o read/write volume que pode ser gerenciado pelo cluster.



7. Em Conectividade, escolha Conectar-se a um recurso de computação do EC2. Esta é a instância do EC2 criada na etapa 1.



> **Note**
>
> Conectar-se a um recurso computacional do EC2 cria automaticamente um grupo de segurança para a conexão de recursos computacionais do EC2 com seu cluster do Amazon DocumentDB. Quando você tiver concluído a criação do seu cluster e quiser ver o grupo de segurança recém-criado, navegue até a lista de clusters e escolha o identificador respectivo. Na guia Conectividade e segurança, acesse Grupos de segurança e encontre seu grupo em Nome do grupo de segurança (ID). Ele conterá o nome do seu cluster e terá uma aparência semelhante a esta: `docdb-ec2-docdb-2023-12-11-21-33-41:i-0e4bb09985d2bbc4c (sg-0238e0b0bf0f73877)`.

8. Na seção Autenticação, insira um nome de usuário para o usuário principal e escolha Autogerenciado. Insira uma senha e confirme-a.

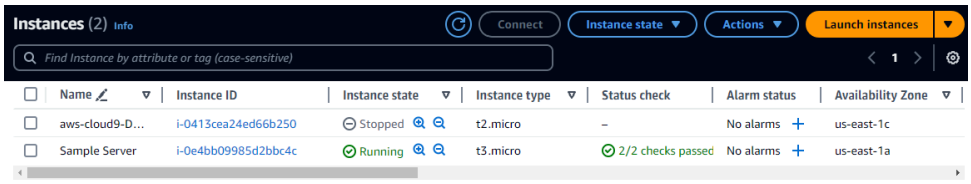
Se, em vez disso, você escolheu Gerenciado em AWS Secrets Manager, consulte Gerenciamento de senhas com o Amazon DocumentDB e AWS Secrets Manager para obter mais informações.

9. Selecione Criar cluster.

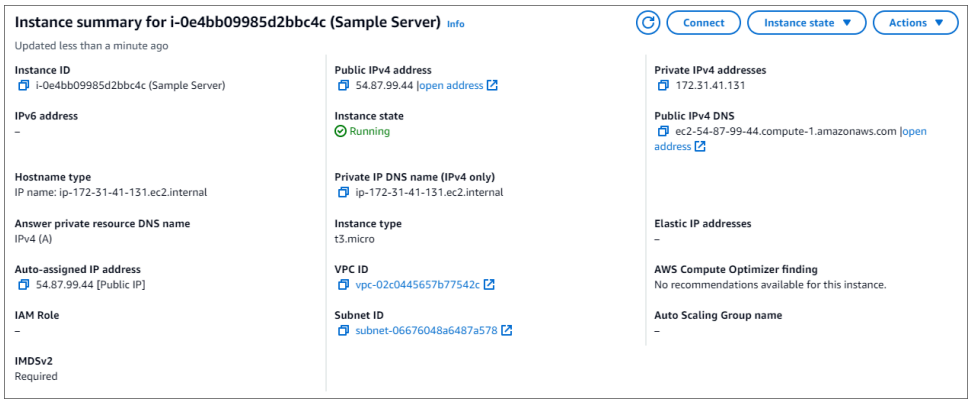
Etapa 3: conectar-se à instância do Amazon EC2

Para instalar o shell do Mongo, você deve primeiro se conectar à sua instância do Amazon EC2. A instalação do shell do Mongo permite que você se conecte e consulte seu cluster do Amazon DocumentDB. Execute as etapas a seguir:

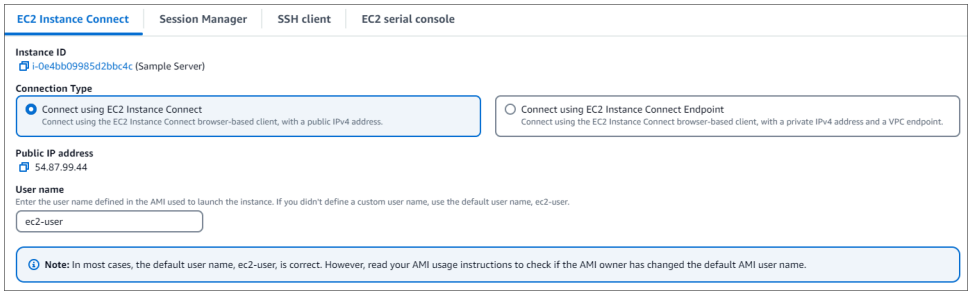
1. No console do Amazon EC2, navegue até suas instâncias e veja se a instância que você acabou de criar está em execução. Se for o caso, selecione a instância clicando no respectivo ID.



2. Selecione Conectar.



3. Você verá quatro guias de opções para seu método de conexão: Amazon EC2 Instance Connect, Session Manager, cliente SSH ou Console de Série do EC2. É necessário escolher um e seguir suas instruções. Ao concluir, escolha Conectar.



> **Note**
>
> Se seu endereço IP mudou depois que você iniciou essa demonstração ou se você estiver voltando ao seu ambiente posteriormente, atualize a regra de entrada do grupo de segurança `demoEC2` para ativar o tráfego de entrada do seu novo endereço de API.

Etapa 4: instalação do shell do MongoDB

Agora o shell do MongoDB pode ser instalado. Ele é um utilitário de linha de comando que você usa para se conectar e consultar seu cluster do Amazon DocumentDB. Atualmente, existem duas versões do shell do MongoDB: a versão mais recente, mongosh, e a versão anterior, mongo shell.

> **Important**
>
> Há uma limitação conhecida com Node.js drivers anteriores à versão 6.13.1, que atualmente não são suportados pela autenticação de identidade IAM para o Amazon DocumentDB. Node.js drivers e ferramentas que usam Node.js driver (por exemplo, mongosh) devem ser atualizados para usar a versão 6.13.1 ou Node.js superior do driver.

Siga as instruções abaixo para instalar o shell do MongoDB em seu sistema operacional.

On Amazon Linux

Para instalar o shell do MongoDB no Amazon Linux

Se você não estiver usando a autenticação do IAM e quiser usar o shell mais recente do MongoDB (mongosh) para se conectar ao seu cluster do Amazon DocumentDB, siga estas etapas:

1. Crie o arquivo do repositório. Na linha de comando da instância do EC2 que você criou, digite o comando a seguir:

```
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. Quando estiver concluído, instale o mongosh com uma das duas opções de comando a seguir no prompt de comando:

   Opção 1: se você escolheu o Amazon Linux 2023 padrão durante a configuração do Amazon EC2, insira este comando:

```
sudo yum install -y mongodb-mongosh-shared-openssl3
```

   Opção 2: se você escolheu o Amazon Linux 2 durante a configuração do Amazon EC2, insira este comando:

```
sudo yum install -y mongodb-mongosh
```

Se você estiver usando a autenticação do IAM, você deve usar a versão anterior do shell do MongoDB (5.0) para se conectar ao seu cluster do Amazon DocumentDB. Siga estas etapas:

1. Crie o arquivo do repositório. Na linha de comando da instância do EC2 que você criou, digite o comando a seguir:

```
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. Quando estiver concluído, instale o shell do mongodb 5.0 com a opção de comando a seguir no prompt de comando:

```
sudo yum install -y mongodb-org-shell
```

On Ubuntu

Para instalar o mongosh no Ubuntu

1. Importe a chave pública que será usada pelo sistema de gerenciamento de pacotes.

```
curl -fsSL https://pgp.mongodb.com/server-5.0.asc | sudo gpg --dearmor -o /usr/
share/keyrings/mongodb-server-5.0.gpg
```

2. Crie o arquivo de lista `mongodb-org-5.0.list` para o MongoDB usando o comando apropriado para a versão do 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 e atualize o banco de dados de pacotes local usando o comando a seguir:

```
sudo apt-get update
```

4. Instale o mongosh.

```
sudo apt-get install -y mongodb-mongosh
```

Para obter informações sobre como instalar versões anteriores do MongoDB no sistema do Ubuntu, consulte Instalar o MongoDB Community Edition no Ubuntu.

On other operating systems

Para instalar o shell do Mongo em outros sistemas operacionais, consulte Instalar o MongoDB Community Edition na documentação do MongoDB.

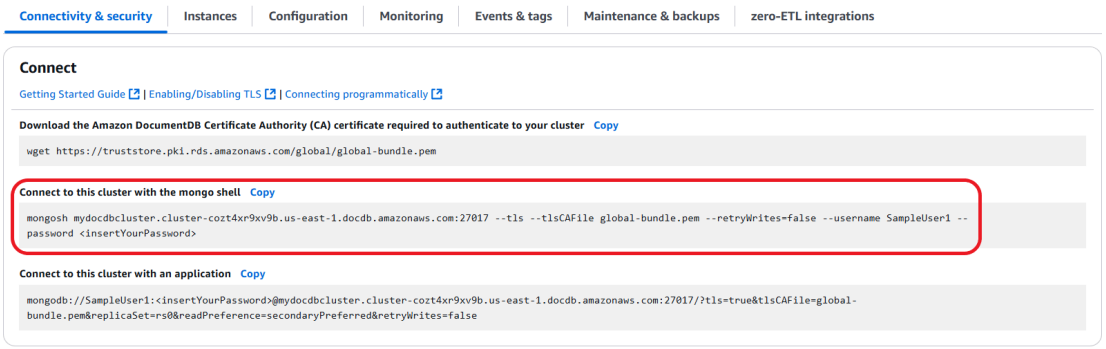
Etapa 5: gerenciar o Amazon DocumentDB TLS

Faça o download do certificado CA para o Amazon DocumentDB com o seguinte código: `wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem`

> **Note**
>
> O Transport Layer Security (TLS) está habilitado por padrão para qualquer novo cluster do Amazon DocumentDB. Para obter mais informações, consulte Criptografia de dados em trânsito.

Etapa 6: conectar-se ao cluster do Amazon DocumentDB

1. No console do Amazon DocumentDB, em Clusters, localize seu cluster. Escolha o cluster que você criou clicando no identificador do cluster desse cluster.
2. Na guia Conectividade e segurança, localize Conectar a este cluster com o shell mongo na caixa Conectar:



   Copie a string de conexão fornecida e cole-a no terminal.

   Faça as seguintes alterações:

   a. Certifique-se de que o nome de usuário correto está na string.
   b. Omita `<insertYourPassword>` para que você seja solicitado a fornecer a senha pelo shell mongo ao se conectar.
   c. Opcional: se você estiver usando a autenticação do IAM ou a versão anterior do shell do MongoDB, modifique sua string de conexão da forma a seguir:

      ```
      mongo --ssl --host mydocdbcluster.cluster-cozt4xr9xv9b.us-east-1.docdb.amazonaws.com:27017 --sslCAFile global-bundle.pem --username SampleUser1 --password
      ```

      Substitua `mydocdbcluster.cluster-cozt4xr9xv9b.us-east-1` pelas mesmas informações do seu cluster.

3. Pressione inserir no seu terminal. Agora você deverá fornecer sua senha. Insira a senha.
4. Ao inserir sua senha e ver o prompt `rs0 [direct: primary] <env-name>>`, você se conecta com êxito ao seu cluster do Amazon DocumentDB.

Para resolver problemas de conexão, consulteSolução de problemas do Amazon DocumentDB.

Etapa 7: inserir e consultar dados

Agora que você está conectado ao seu cluster, pode executar algumas consultas para se familiarizar com o uso de um banco de dados de documentos.

1. Para inserir um único documento, digite o seguinte:

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

   Você obterá o seguinte resultado:

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

2. É possível ler o documento que escreveu com o comando `findOne()` (porque ele retorna apenas um único documento). Insira o seguinte:

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

   Você obterá o seguinte resultado:

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

3. Para realizar mais algumas consultas, considere um caso de uso de perfis de jogos. Primeiro, insira algumas entradas em uma coleção intitulada `profiles`. Insira o seguinte:

```
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 }
```

```
])
```

Você obterá o seguinte resultado:

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

4. Use o `find()` comando para retornar todos os documentos na coleção de perfis. Insira o seguinte:

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

Você receberá uma saída que corresponderá aos dados digitados na Etapa 3.

5. Use uma consulta para um único documento por meio de um filtro. Insira o seguinte:

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

Você obterá o seguinte resultado:

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

6. Agora vamos tentar encontrar um perfil e modificá-lo usando o comando `findAndModify`. Atribuiremos ao usuário Matt mais 10z pontos com o código a seguir:

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

Você obtém o seguinte resultado (observe que a pontuação dele ainda não aumentou):

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

7. É possível verificar se a pontuação dele mudou com a seguinte consulta:

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

Você obterá a seguinte saída:

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

Etapa 8: explorar

Parabéns! Você concluiu com êxito o Guia de início rápido do Amazon DocumentDB.

E depois? Saiba como aproveitar totalmente esse poderoso banco de dados com alguns de seus atributos populares:

- Gerenciamento de clusters e instâncias do Amazon DocumentDB
- Escalabilidade
- Fazer backup e restaurar

> **Note**
>
> Para economizar, é possível interromper seu cluster Amazon DocumentDB para reduzir custos ou excluir o cluster. Por padrão, após 30 minutos de inatividade, seu AWS Cloud9 ambiente interromperá a instância subjacente do Amazon EC2.

Conectar automaticamente uma instância do EC2 a um banco de dados do Amazon DocumentDB existente

Para o procedimento a seguir, pressupõe-se que você tenha um cluster do Amazon DocumentDB existente e uma instância do Amazon EC2 existente.

Acessar seu cluster do Amazon DocumentDB e configure a conexão do Amazon EC2

1. Acesse o cluster do Amazon DocumentDB.

    a. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.

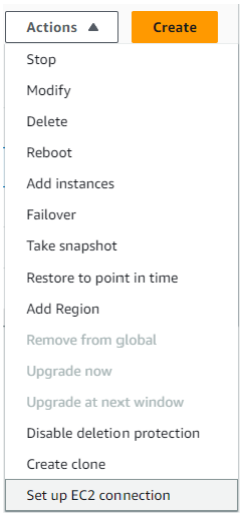
    b. No painel de navegação, escolha Clusters.

> **Tip**
>
> Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

c. Especifique o cluster que você quer escolhendo o botão à esquerda do nome do cluster.

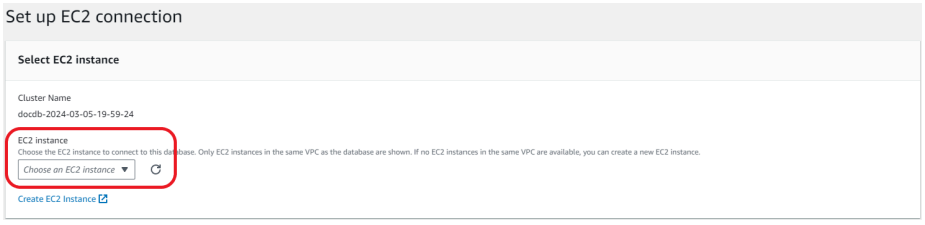
2. Configure a conexão do Amazon EC2.

a. Escolha Ações e Configurar conexão do EC2.



A caixa de diálogo Configurar conexão do EC2 é exibida.

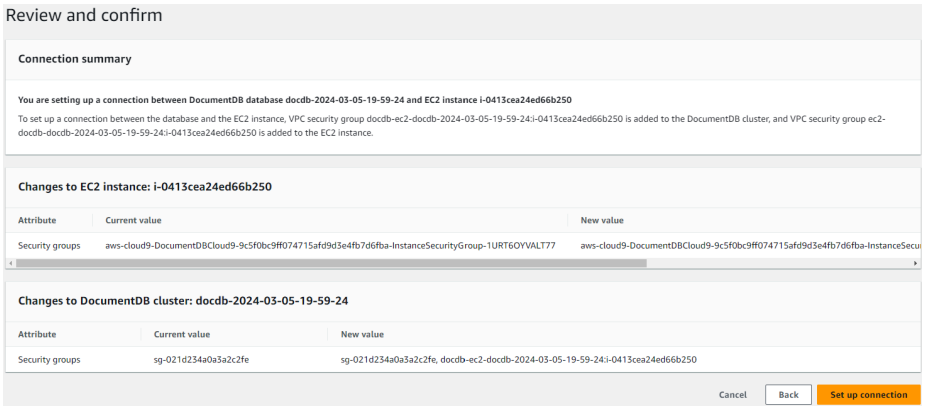
b. No campo Instância do EC2, escolha a instância do EC2 que você quer conectar ao cluster.



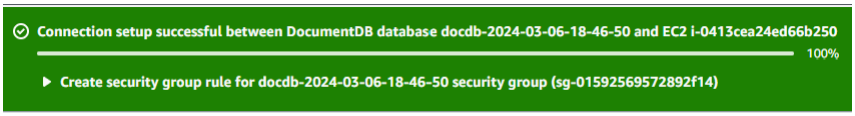
c. Escolha Continuar.

A caixa de diálogo Revisar e confirmar é exibida.

d. Verifique se as alterações estão corretas. Em seguida, escolha Configurar conexão.



Se for bem-sucedida, a seguinte verificação será exibida:



Visão geral da conectividade automática com uma instância do EC2

Quando você configura uma conexão entre uma instância do EC2 e um banco de dados do Amazon DocumentDB, o Amazon DocumentDB automaticamente configura o grupo de segurança VPC para sua instância do EC2 e para seu banco de dados do Amazon DocumentDB.

Confira a seguir os requisitos para a conexão de uma instância do EC2 com um banco de dados do Amazon DocumentDB:

- A instância do EC2 deve existir na mesma VPC do banco de dados do Amazon DocumentDB.

  Se não houver nenhuma instância do EC2 na mesma VPC, o console fornecerá um link para que você crie uma.
- O usuário que configura a conectividade deve ter permissões para realizar as seguintes operações do Amazon EC2:
  - `ec2:AuthorizeSecurityGroupEgress`
  - `ec2:AuthorizeSecurityGroupIngress`
  - `ec2:CreateSecurityGroup`
  - `ec2:DescribeInstances`
  - `ec2:DescribeNetworkInterfaces`
  - `ec2:DescribeSecurityGroups`
  - `ec2:ModifyNetworkInterfaceAttribute`

- `ec2:RevokeSecurityGroupEgress`

Se a instância de banco de dados e a instância do EC2 estiverem em zonas de disponibilidade diferentes, sua conta poderá incorrer em custos entre as zonas.

Quando você configura uma conexão com uma instância do EC2, o Amazon DocumentDB atua de acordo com a configuração atual dos grupos de segurança associados ao banco de dados do Amazon DocumentDB e à instância do EC2, conforme descrito na tabela a seguir:

Configuração atual do grupo de segurança do Amazon DocumentDB	Configuração atual do grupo de segurança do EC2	Ação do Amazon DocumentDB
Há um ou mais grupos de segurança associados ao banco de dados do Amazon DocumentDB com um nome correspondente ao padrão `DocumentDB-ec2-n`. Um grupo de segurança que corresponde ao padrão não foi modificado. Esse grupo de segurança tem apenas uma regra de saída com o grupo de segurança da VPC da instância do EC2 como origem.	Há um ou mais grupos de segurança associados à instância do EC2 com um nome correspondente ao padrão `DocumentDB-ec2-n` (em que n é um número). Um grupo de segurança que corresponde ao padrão não foi modificado. Esse grupo de segurança tem apenas uma regra de saída com o grupo de segurança da VPC do banco de dados do Amazon DocumentDB como origem.	O Amazon DocumentDB não realiza nenhuma ação. Uma conexão já foi configurada automaticamente entre a instância do EC2 e o banco de dados do Amazon DocumentDB. Como já existe uma conexão entre a instância do EC2 e o banco de dados do Amazon DocumentDB, os grupos de segurança não são modificados.
Qualquer uma das seguintes condições se aplica: • Não há um grupo de segurança associado ao banco de dados do Amazon DocumentDB com um nome correspondente ao	Qualquer uma das seguintes condições se aplica: • Não há um grupo de segurança associado à instância do EC2 com um nome que corresponde ao padrão `ec2-DocumentDB-n` .	Ação do Amazon DocumentDB: criar novos grupos de segurança

Configuração atual do grupo de segurança do Amazon DocumentDB	Configuração atual do grupo de segurança do EC2	Ação do Amazon DocumentDB
padrão `DocumentDB-ec2-n`. • Há um ou mais grupos de segurança associados ao Amazon DocumentDB com um nome correspondente ao padrão `DocumentDB-ec2-n` . No entanto, o Amazon DocumentDB não pode usar nenhum desses grupos de segurança para a conexão com a instância do EC2. O Amazon DocumentDB não pode usar um grupo de segurança que não tenha uma regra de entrada no grupo de segurança da VPC da instância do EC2 como origem. O Amazon DocumentDB também não pode usar um grupo de segurança que tenha sido modificado. São exemplos de modificação a adição de uma regra ou a alteração da porta de uma regra existente.	• Há um ou mais grupos de segurança associados à instância do EC2 com um nome que corresponde ao padrão `ec2-DocumentDB-n` . No entanto, o Amazon DocumentDB não pode usar nenhum desses grupos de segurança para a conexão com o banco de dados do Amazon DocumentDB. O Amazon DocumentDB não pode usar um grupo de segurança que não tenha uma regra de saída com o grupo de segurança da VPC do banco de dados do Amazon DocumentDB como origem. O Amazon DocumentDB também não pode usar um grupo de segurança que tenha sido modificado.	

Configuração atual do grupo de segurança do Amazon DocumentDB	Configuração atual do grupo de segurança do EC2	Ação do Amazon DocumentDB
Há um ou mais grupos de segurança associados ao banco de dados do Amazon DocumentDB com um nome correspondente ao padrão `DocumentDB-ec2-n`. Um grupo de segurança que corresponde ao padrão não foi modificado. Esse grupo de segurança tem apenas uma regra de saída com o grupo de segurança da VPC da instância do EC2 como origem.	Há um ou mais grupos de segurança associados à instância do EC2 com um nome que corresponde ao padrão `ec2-DocumentDB-n`. No entanto, o Amazon DocumentDB não pode usar nenhum desses grupos de segurança para a conexão com o banco de dados do Amazon DocumentDB. O Amazon DocumentDB não pode usar um grupo de segurança que não tenha uma regra de saída com o grupo de segurança da VPC do banco de dados do Amazon DocumentDB como origem. O Amazon DocumentDB também não pode usar um grupo de segurança que tenha sido modificado.	Ação do Amazon DocumentDB: criar novos grupos de segurança

Configuração atual do grupo de segurança do Amazon DocumentDB	Configuração atual do grupo de segurança do EC2	Ação do Amazon DocumentDB
Há um ou mais grupos de segurança associados ao banco de dados do Amazon DocumentDB com um nome correspondente ao padrão `DocumentDB-ec2-n`. Um grupo de segurança que corresponde ao padrão não foi modificado. Esse grupo de segurança tem apenas uma regra de saída com o grupo de segurança da VPC da instância do EC2 como origem.	Existe um grupo de segurança do EC2 válido para a conexão, mas ele não está associado à instância do EC2. Esse grupo de segurança tem um nome que corresponde ao padrão `DocumentDB-ec2-n`. Não foi modificado. Ele tem apenas uma regra de saída com o grupo de segurança da VPC do banco de dados do Amazon DocumentDB como origem.	Ação do Amazon DocumentDB: associar o grupo de segurança do EC2

Configuração atual do grupo de segurança do Amazon DocumentDB	Configuração atual do grupo de segurança do EC2	Ação do Amazon DocumentDB
Qualquer uma das seguintes condições se aplica: • Não há um grupo de segurança associado ao banco de dados do Amazon DocumentDB com um nome correspondente ao padrão `DocumentDB-ec2-n`. • Há um ou mais grupos de segurança associados ao Amazon DocumentDB com um nome correspondente ao padrão `DocumentDB-ec2-n` . No entanto, o Amazon DocumentDB não pode usar nenhum desses grupos de segurança para a conexão com a instância do EC2. O Amazon DocumentDB não pode usar um grupo de segurança que não tenha uma regra de entrada no grupo de segurança da VPC da instância do EC2 como origem. O Amazon DocumentDB também não pode usar um grupo de segurança que tenha sido modificado.	Há um ou mais grupos de segurança associados à instância do EC2 com um nome que corresponde ao padrão `DocumentDB-ec2-n`. Um grupo de segurança que corresponde ao padrão não foi modificado. Esse grupo de segurança tem apenas uma regra de saída com o grupo de segurança da VPC do banco de dados do Amazon DocumentDB como origem.	Ação do Amazon DocumentDB: criar novos grupos de segurança

Ação do Amazon DocumentDB: criar novos grupos de segurança

O Amazon DocumentDB realiza as seguintes ações:

- Cria um grupo de segurança que corresponde ao padrão `DocumentDB-ec2-n`. Esse grupo de segurança tem uma regra de entrada com o grupo de segurança da VPC da instância do EC2 como origem. Esse grupo de segurança está associado ao banco de dados do Amazon DocumentDB e permite que a instância do EC2 acesse o banco de dados do Amazon DocumentDB.
- Cria um grupo de segurança que corresponde ao padrão `ec2-DocumentDB-n`. Esse grupo de segurança tem uma regra de saída com o grupo de segurança da VPC do banco de dados do Amazon DocumentDB como origem. Esse grupo de segurança está associado à instância do EC2 e permite que ela envie tráfego ao banco de dados do Amazon DocumentDB.

Ação do Amazon DocumentDB: associar o grupo de segurança do EC2

O Amazon DocumentDB associa o grupo de segurança do EC2 existente e válido à instância do EC2. Esse grupo de segurança permite que a instância do EC2 envie tráfego ao banco de dados do Amazon DocumentDB.

Visualizar recursos computacionais conectados

Você pode usar o Console de gerenciamento da AWS para visualizar os recursos computacionais que estão conectados a um banco de dados Amazon DocumentDB. Os recursos mostrados incluem conexões de recursos computacionais que foram configuradas automaticamente. É possível configurar a conectividade com recursos computacionais automaticamente das seguintes maneiras:

- É possível selecionar o recurso computacional ao criar o banco de dados. Para obter mais informações, consulte Criação Criar um cluster do Amazon DocumentDB de um cluster de Multi-AZ banco de dados.
- É possível configurar a conectividade entre um banco de dados existente e um recurso computacional. Para obter mais informações, consulte Conectar o Amazon EC2 automaticamente.

Os recursos computacionais listados não incluem aqueles que foram conectados manualmente ao banco de dados. Por exemplo, você pode permitir que um recurso computacional acesse um banco de dados manualmente adicionando uma regra ao grupo de segurança da VPC associado ao banco de dados.

Para que um recurso computacional seja listado, as seguintes condições devem ser atendidas:

- O nome do grupo de segurança associado ao recurso computacional correspondente ao padrão `ec2-DocumentDB-n` (em que n é um número).
- O grupo de segurança associado ao recurso computacional tem uma regra de saída com o intervalo de portas definido como a porta usada pelo banco de dados do Amazon DocumentDB.
- O grupo de segurança associado ao recurso computacional tem uma regra de saída com a fonte definida como um grupo de segurança associado ao banco de dados do Amazon DocumentDB.
- O nome do grupo de segurança associado ao banco de dados do Amazon DocumentDB corresponde ao padrão `DocumentDB-ec2-n` (em que n é um número).
- O grupo de segurança associado ao banco de dados do Amazon DocumentDB dtem uma regra de entrada com o intervalo de portas definido como a porta usada pelo banco de dados do Amazon DocumentDB.
- O grupo de segurança associado ao banco de dados do Amazon DocumentDB tem uma regra de entrada com a fonte definida como um grupo de segurança associado ao recurso computacional.

Para visualizar recursos computacionais conectados a um banco de dados do Amazon DocumentDB

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Bancos de dados e depois escolha o nome do banco de dados do Amazon DocumentDB.
3. Na guia Conectividade e segurança, veja os recursos computacionais na seção Recursos computacionais conectados.

Conectar o Amazon EC2 manualmente

Tópicos

- Etapa 1: criar uma instância do Amazon EC2
- Etapa 2: criar um grupo de segurança
- Etapa 3: criar um cluster do Amazon DocumentDB
- Etapa 4: conectar a sua instância do Amazon EC2
- Etapa 5: instalação do shell do MongoDB
- Etapa 6: gerenciar o Amazon DocumentDB TLS

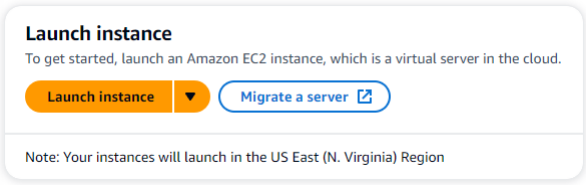
- Etapa 7: conectar ao cluster do Amazon DocumentDB
- Etapa 8: inserir e consultar dados
- Etapa 9: explorar

As etapas a seguir pressupõem que você concluiu as etapas do tópico Pré-requisitos.

Etapa 1: criar uma instância do Amazon EC2

Nesta etapa, você criará uma instância do Amazon EC2 na mesma região e na Amazon VPC que você usará posteriormente para provisionar seu cluster Amazon DocumentDB.

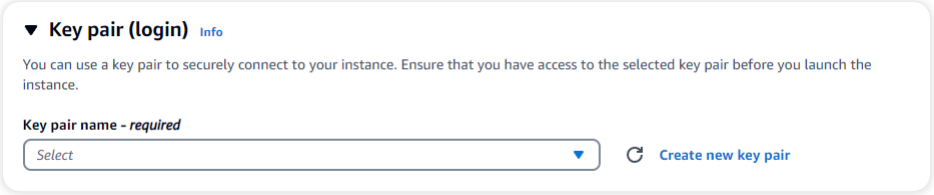
1. No console do Amazon EC2, selecione Iniciar instância.



2. Insira um nome ou identificador no campo Nome, localizado na seção Nome e tags.
3. Na lista suspensa Imagem de máquina da Amazon (AMI), localize a AMI do Amazon Linux 2 e escolha-a.



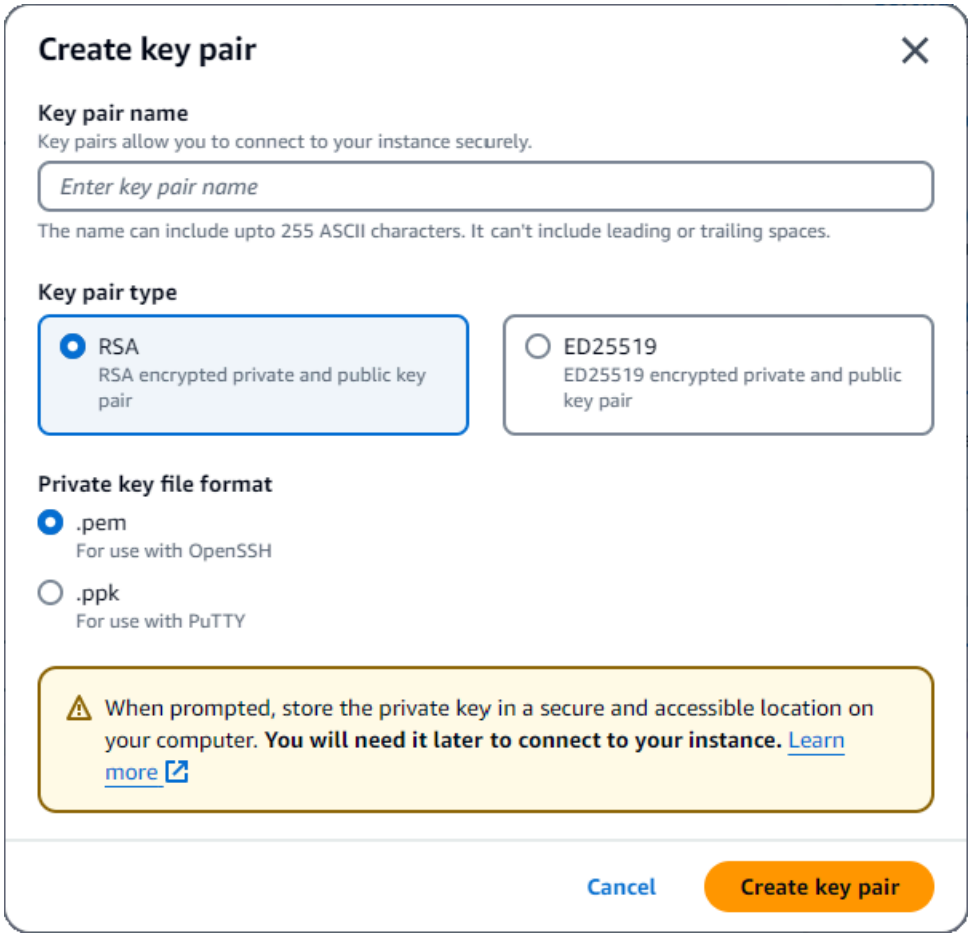
4. Localize e escolha t3.micro na lista suspensa Tipo de instância.
5. Na seção Par de chaves (login) insira o identificador de um par de chaves existente ou selecione Criar um novo par de chaves.



É necessário fornecer um par de chaves do Amazon EC2.

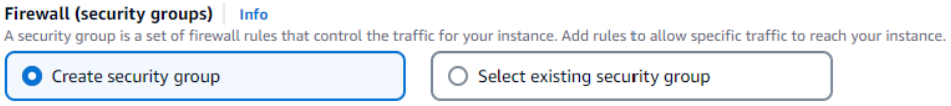
- Se você tiver um par de chaves do Amazon EC2:
  1. Selecione um par de chaves, escolha seu par de chaves na lista.
  2. Você já deve ter o arquivo de chave privada (arquivo .pem ou .ppk) disponível para fazer login na instância do Amazon EC2.

- Se você não tiver um par de chaves do Amazon EC2:
  1. Ao escolher Criar novo par de chaves, a caixa de diálogo Criar par de chaves é exibida.
  2. Insira um nome no campo Nome do par de chaves.
  3. Escolha o Tipo de par de chaves e o Formato de arquivo de chave privada.
  4. Escolha Create key pair (Criar par de chaves).



> **Note**
>
> Para fins de segurança, é altamente recomendável usar um par de chaves para conectividade de SSH e de internet com sua instância do EC2.

6. Na seção Configurações de rede, em Firewall (grupos de segurança), escolha Criar grupo de segurança ou Selecionar grupo de segurança existente.



Se você optar por selecionar um grupo de segurança existente, selecione um na lista suspensa Grupos de segurança comuns.

Se você optar por criar um novo grupo de segurança, faça o seguinte:

a. Verifique todas as regras de permissão de tráfego que se aplicam à sua conectividade do EC2.

b. No campo IP, escolha Meu IP ou selecione Personalizado para escolher em uma lista de blocos CIDR, listas de prefixos ou grupos de segurança. Não recomendamos Qualquer lugar como opção, a menos que sua instância do EC2 esteja em uma rede isolada, pois ela permite que qualquer endereço IP acesse sua instância do EC2.

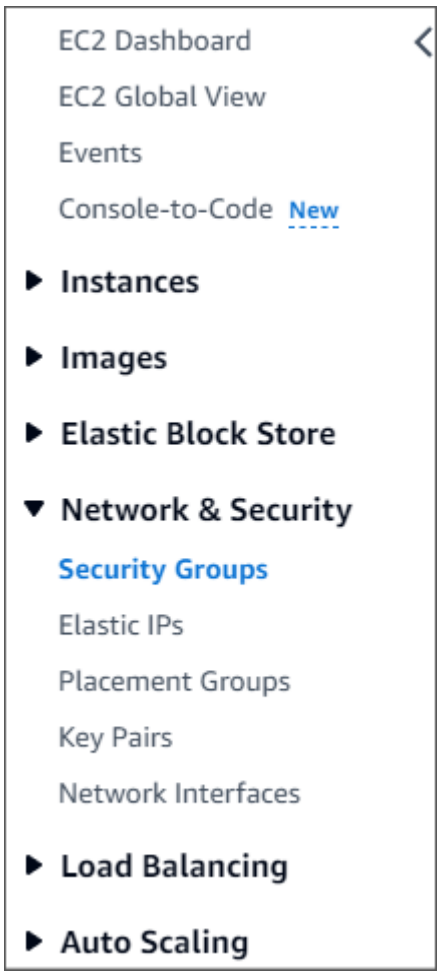


7. Na seção Resumo, revise sua configuração do EC2 e escolha Iniciar instância, se tudo estiver correto.

Etapa 2: criar um grupo de segurança

Agora você criará um novo grupo de segurança em sua Amazon VPC padrão. O grupo de segurança `demoDocDB` permite que você se conecte ao seu cluster Amazon DocumentDB na porta 27017 (a porta padrão para o Amazon DocumentDB) a partir da sua instância do Amazon EC2.

1. No console de gerenciamento do Amazon EC2, em Rede e segurança, escolha Grupos de segurança.



2. Escolha Criar grupo de segurança.



3. Na seção Detalhes básicos:

   a. Em Nome do grupo de segurança, insira `demoDocDB`.

   b. Em Descrição, insira uma descrição.

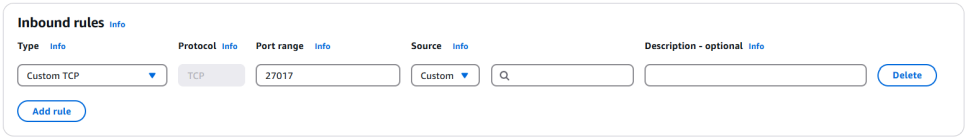
   c. Para VPC, aceite o uso da sua VPC padrão.

4. Na seção Regras de entrada, escolha Adicionar regra.

   a. Em Tipo, selecione Regra TCP personalizada (padrão).

   b. Em Intervalo de portas, insira `27017`.

   c. Em Source, escolha Custom. No campo ao lado, procure o grupo de segurança que você acabou de criar na etapa 1. Talvez seja necessário atualizar seu navegador para que o console do Amazon EC2 preencha automaticamente o nome da fonte.



5. Aceite todos os outros padrões e escolha Criar grupo de segurança.



Etapa 3: criar um cluster do Amazon DocumentDB

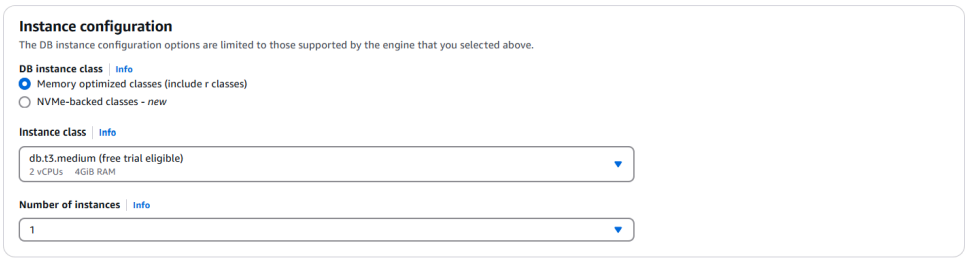
Enquanto a instância do Amazon EC2 estiver sendo provisionada, você criará seu cluster Amazon DocumentDB.

1. Navegue até o console do Amazon DocumentDB e escolha Clusters no painel de navegação.
2. Escolha Criar.
3. Deixe a configuração Tipo de cluster com o valor padrão, Cluster baseado em instância.
4. Em Configuração do cluster, em Identificador do cluster, insira um nome exclusivo. Observe que o console mudará todos os nomes de cluster para minúsculas, independentemente de como eles sejam inseridos.

   Deixe a Versão do mecanismo com o valor padrão de 5.0.0.
5. Em Configuração de armazenamento em cluster, deixe a configuração padrão Amazon DocumentDB Padrão.
6. Em Configuração da instância:

   - Em Classe de instância de banco de dados, escolha Classes otimizadas para memória (incluir classes r) (esse é o padrão).

     A outra opção de instância são NVMe-backed classes. Para saber mais, consulte NVMe-backed instâncias.
   - Em Classe de instância, escolha o tipo de instância que atende às suas necessidades. Para obter uma explicação mais detalhada das classes de instância, consulte Especificações de classe de instância.
   - Em número de casos, escolha um número que melhor reflita suas necessidades. Lembre-se de que quanto menor o número, menor o custo e menor o read/write volume que pode ser gerenciado pelo cluster.



7. Em Conectividade, mantenha a configuração padrão Não se conectar a um recurso computacional do EC2.

> **Note**
>
> A conexão com um recurso computacional do EC2 cria automaticamente grupos de segurança para a conexão com seu cluster. Como você criou manualmente esses grupos de segurança na etapa anterior, deverá selecionar Não se conectar a um recurso computacional do EC2 para não criar um segundo conjunto de grupos de segurança.

8. Na seção Autenticação, insira um nome de usuário para o usuário principal e escolha Autogerenciado. Insira uma senha e confirme-a.

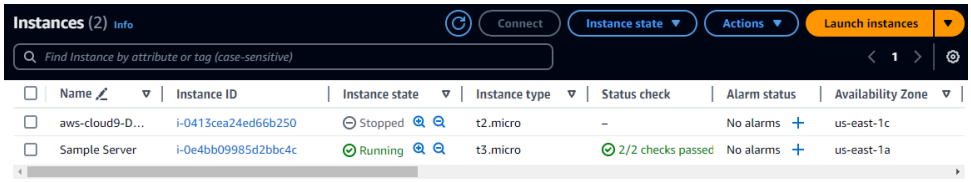
   Se, em vez disso, você escolheu Gerenciado em AWS Secrets Manager, consulte Gerenciamento de senhas com o Amazon DocumentDB e AWS Secrets Manager para obter mais informações.

9. Selecione Criar cluster.

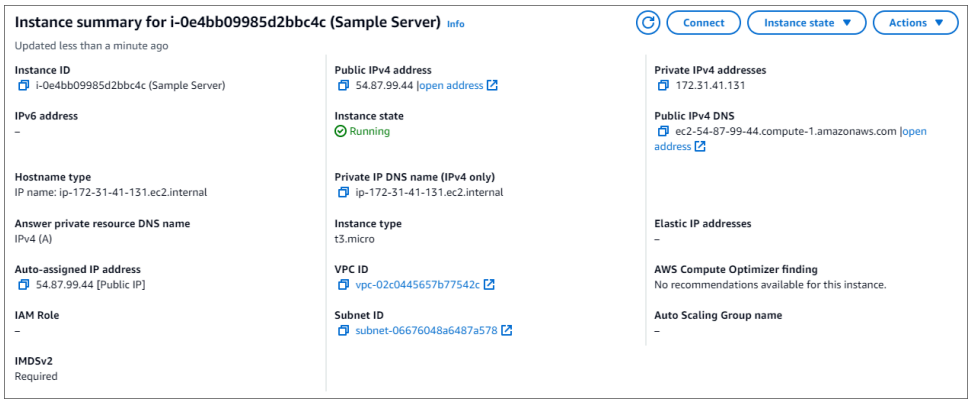
Etapa 4: conectar a sua instância do Amazon EC2

Para instalar o shell do MongoDB, você deve primeiro se conectar à sua instância do Amazon EC2. A instalação do shell do Mongo permite que você se conecte e consulte seu cluster do Amazon DocumentDB. Execute as etapas a seguir:

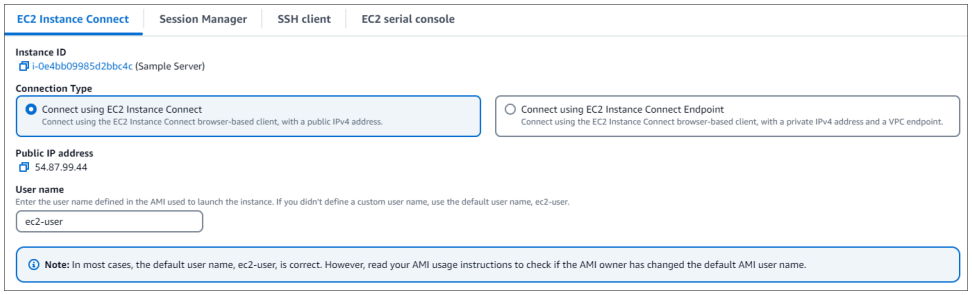
1. No console do Amazon EC2, navegue até suas instâncias e veja se a instância que você acabou de criar está em execução. Se for o caso, selecione a instância clicando no respectivo ID.



2. Selecione Conectar.



3. Você verá quatro guias de opções para seu método de conexão: Amazon EC2 Instance Connect, Session Manager, cliente SSH ou Console de Série do EC2. É necessário escolher um e seguir suas instruções. Ao concluir, escolha Conectar.



> **Note**
>
> Se seu endereço IP mudou depois que você iniciou essa demonstração ou se você estiver voltando ao seu ambiente posteriormente, atualize a regra de entrada do grupo de segurança `demoEC2` para ativar o tráfego de entrada do seu novo endereço de API.

Etapa 5: instalação do shell do MongoDB

Agora o shell do MongoDB pode ser instalado. Ele é um utilitário de linha de comando que você usa para se conectar e consultar seu cluster do Amazon DocumentDB. Atualmente, existem duas versões do shell do MongoDB: a versão mais recente, mongosh, e a versão anterior, mongo shell.

> **Important**
>
> Há uma limitação conhecida com Node.js drivers anteriores à versão 6.13.1, que atualmente não são suportados pela autenticação de identidade IAM para o Amazon DocumentDB.

Node.js drivers e ferramentas que usam Node.js driver (por exemplo, mongosh) devem ser atualizados para usar a versão 6.13.1 ou Node.js superior do driver.

Siga as instruções abaixo para instalar o shell do MongoDB em seu sistema operacional.

On Amazon Linux

Para instalar o shell do MongoDB no Amazon Linux

Se você não estiver usando o IAM e quiser usar o shell mais recente do MongoDB (mongosh) para se conectar ao seu cluster do Amazon DocumentDB, siga estas etapas:

1. Crie o arquivo do repositório. Na linha de comando da instância do EC2 que você criou, digite o comando a seguir:

```
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. Quando estiver concluído, instale o mongosh com uma das duas opções de comando a seguir no prompt de comando:

   Opção 1: se você escolheu o Amazon Linux 2023 padrão durante a configuração do Amazon EC2, insira este comando:

```
sudo yum install -y mongodb-mongosh-shared-openssl3
```

   Opção 2: se você escolheu o Amazon Linux 2 durante a configuração do Amazon EC2, insira este comando:

```
sudo yum install -y mongodb-mongosh
```

Se você estiver usando o IAM, você deve usar a versão anterior do shell do MongoDB (5.0) para se conectar ao seu cluster do Amazon DocumentDB. Siga estas etapas:

1. Crie o arquivo do repositório. Na linha de comando da instância do EC2 que você criou, digite o comando a seguir:

```
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. Quando estiver concluído, instale o shell do mongodb 5.0 com a opção de comando a seguir no prompt de comando:

```
sudo yum install -y mongodb-org-shell
```

On Ubuntu

Para instalar o mongosh no Ubuntu

1. Importe a chave pública que será usada pelo sistema de gerenciamento de pacotes.

```
curl -fsSL https://pgp.mongodb.com/server-5.0.asc | sudo gpg --dearmor -o /usr/
share/keyrings/mongodb-server-5.0.gpg
```

2. Crie o arquivo de lista `mongodb-org-5.0.list` para o MongoDB usando o comando apropriado para a versão do 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 e atualize o banco de dados de pacotes local usando o comando a seguir:

```
sudo apt-get update
```

4. Instale o mongosh.

```
sudo apt-get install -y mongodb-mongosh
```

Para obter informações sobre como instalar versões anteriores do MongoDB no sistema do Ubuntu, consulte Instalar o MongoDB Community Edition no Ubuntu.

On other operating systems

Para instalar o shell do Mongo em outros sistemas operacionais, consulte Instalar o MongoDB Community Edition na documentação do MongoDB.

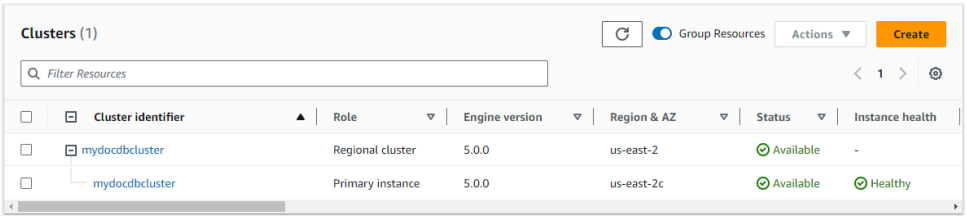
Etapa 6: gerenciar o Amazon DocumentDB TLS

Faça o download do certificado CA para o Amazon DocumentDB com o seguinte código: `wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem`

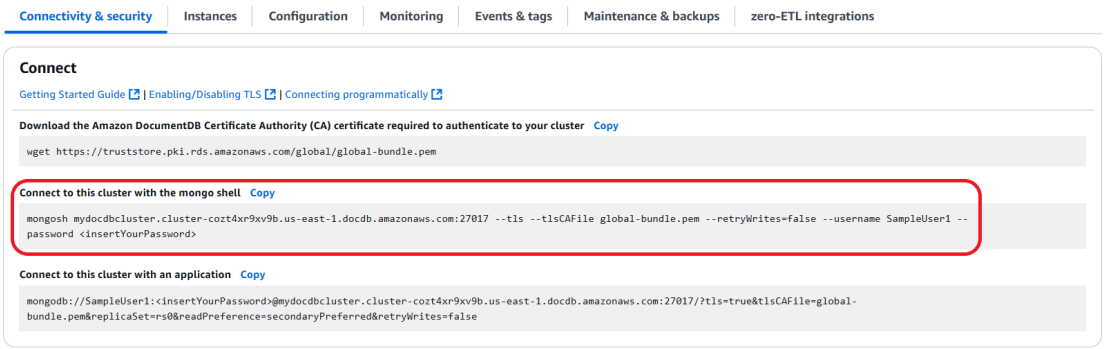
> **Note**
>
> O Transport Layer Security (TLS) está habilitado por padrão para qualquer novo cluster do Amazon DocumentDB. Para obter mais informações, consulte Criptografia de dados em trânsito.

Etapa 7: conectar ao cluster do Amazon DocumentDB

1. No console do Amazon DocumentDB, em Clusters, localize seu cluster. Escolha o cluster que você criou clicando no identificador do cluster desse cluster.



2. Na guia Conectividade e segurança, localize Conectar a este cluster com o shell mongo na caixa Conectar:



   Copie a string de conexão fornecida e cole-a no terminal.

Faça as seguintes alterações:

a. Certifique-se de que o nome de usuário correto está na string.

b. Omita `<insertYourPassword>` para que você seja solicitado a fornecer a senha pelo shell mongo ao se conectar.

c. Opcional: se você estiver usando a autenticação do IAM ou a versão anterior do shell do MongoDB, modifique sua string de conexão da forma a seguir:

```
mongo --ssl --host mydocdbcluster.cluster-cozt4xr9xv9b.us-
east-1.docdb.amazonaws.com:27017 --sslCAFile global-bundle.pem --
username SampleUser1 --password
```

Substitua `mydocdbcluster.cluster-cozt4xr9xv9b.us-east-1` pelas mesmas informações do seu cluster.

3. Pressione inserir no seu terminal. Agora você deverá fornecer sua senha. Insira a senha.
4. Ao inserir sua senha e ver o prompt `rs0 [direct: primary] <env-name>>`, você se conecta com êxito ao seu cluster do Amazon DocumentDB.

Para resolver problemas de conexão, consulteSolução de problemas do Amazon DocumentDB.

Etapa 8: inserir e consultar dados

Agora que você está conectado ao seu cluster, pode executar algumas consultas para se familiarizar com o uso de um banco de dados de documentos.

1. Para inserir um único documento, digite o seguinte:

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

Você obterá o seguinte resultado:

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

2. É possível ler o documento que escreveu com o comando `findOne()` (porque ele retorna apenas um único documento). Insira o seguinte:

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

Você obterá o seguinte resultado:

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

3. Para realizar mais algumas consultas, considere um caso de uso de perfis de jogos. Primeiro, insira algumas entradas em uma coleção intitulada `profiles`. Insira o seguinte:

```
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 }
])
```

Você obterá o seguinte resultado:

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

4. Use o `find()` comando para retornar todos os documentos na coleção de perfis. Insira o seguinte:

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

Você receberá uma saída que corresponderá aos dados digitados na Etapa 3.

5. Use uma consulta para um único documento por meio de um filtro. Insira o seguinte:

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

Você obterá o seguinte resultado:

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

6. Agora vamos tentar encontrar um perfil e modificá-lo usando o comando `findAndModify`. Atribuiremos ao usuário Matt mais 10z pontos com o código a seguir:

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

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

Você obtém o seguinte resultado (observe que a pontuação dele ainda não aumentou):

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

7. É possível verificar se a pontuação dele mudou com a seguinte consulta:

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

   Você obterá o seguinte resultado:

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

Etapa 9: explorar

Parabéns! Você concluiu com êxito o Guia de início rápido do Amazon DocumentDB.

E depois? Saiba como aproveitar totalmente esse poderoso banco de dados com alguns de seus atributos populares:

- Gerenciamento de clusters e instâncias do Amazon DocumentDB
- Escalabilidade
- Fazer backup e restaurar

> **Note**
>
> Para economizar, é possível interromper seu cluster Amazon DocumentDB para reduzir custos ou excluir o cluster. Por padrão, após 30 minutos de inatividade, seu AWS Cloud9 ambiente interromperá a instância subjacente do Amazon EC2.

Conecte-se usando o driver JDBC do Amazon DocumentDB

O driver JDBC para Amazon DocumentDB fornece uma SQL-relational interface para desenvolvedores e permite a conectividade com ferramentas de BI, como Tableau e. DbVisualizer

Para obter informações mais detalhadas, consulte a documentação do Amazon DocumentDB JDBC Driver em. GitHub

Tópicos

- Introdução
- Conecte-se ao Amazon DocumentDB a partir do Tableau Desktop
- Conecte-se ao Amazon DocumentDB a partir de DbVisualizer
- Geração automática de esquema JDBC
- Suporte e limitações do SQL
- Solução de problemas

Introdução

Etapa 1. Criar Amazon DocumentDB Cluster

Se você não tiver um cluster Amazon DocumentDB criado, crie um usando as instruções noConceitos básicos do Amazon DocumentDB.

> **Note**
>
> O Amazon DocumentDB é um serviço somente de nuvem privada virtual (VPC). Se você estiver se conectando a partir de uma máquina local, fora da VPC do cluster, precisará criar uma conexão SSH com uma instância do Amazon EC2. Nesse caso, inicie seu cluster usando as instruções emConecte usando o Amazon EC2. Consulte Usar um túnel SSH para se conectar ao Amazon DocumentDB para obter mais informações sobre tunelamento SSH e quando você precisar dele.

Etapa 2. Instalação do JRE ou JDK

Dependendo da sua aplicação de BI, talvez seja necessário garantir que uma versão 8 ou posterior da instalação do JRE ou JDK de 64 bits esteja instalada no seu computador. É possível fazer o download do Java SE Runtime Environment 8 aqui.

Etapa 3. Faça o download do driver JDBC DocumentDB

Faça o download do driver JDBC do DocumentDB aqui. O driver é empacotado como um único arquivo JAR (por exemplo, documentdb-jdbc-1.0.0-all.jar).

Etapa 4: Usando um túnel SSH para se conectar ao Amazon DocumentDB

Os clusters do Amazon DocumentDB (compatível com MongoDB) são implantados dentro de uma Amazon Virtual Private Cloud (Amazon VPC). Eles podem ser acessados diretamente pelas instâncias do Amazon EC2 ou por outros AWS serviços implantados na mesma Amazon VPC. Além disso, o Amazon DocumentDB pode ser acessado por instâncias EC2a ou outros AWS serviços em diferentes VPCs na mesma AWS região ou em outras regiões por meio do emparelhamento de VPCs.

É possível usar o tunelamento SSH (também conhecido como encaminhamento de portas) para acessar seus recursos do Amazon DocumentDB, de fora da VPC do cluster. Esse será o caso da maioria dos usuários que não executam suas aplicações em uma VM na mesma VPC do cluster DocumentDB.

Para criar um túnel SSH, você precisa de uma instância do Amazon EC2 em execução na mesma Amazon VPC que seu cluster do Amazon DocumentDB. É possível usar uma instância do EC2 existente na mesma VPC que seu cluster ou criar uma instância. É possível configurar um túnel SSH para o cluster do Amazon DocumentDB `sample-cluster.node.us-east-1.docdb.amazonaws.com` executando o seguinte comando em seu computador 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
```

O sinalizador -L é usado para encaminhar uma porta local. Esse é um pré-requisito para se conectar a qualquer ferramenta de BI em execução em um cliente fora da sua VPC. Depois de executar a etapa acima, é possível passar para as próximas etapas da ferramenta de BI de sua escolha.

Para obter mais informações sobre tunelamento SSH, consulte a documentação sobre Como usar um túnel SSH para se conectar ao Amazon DocumentDB.

Conecte-se ao Amazon DocumentDB a partir do Tableau Desktop

Tópicos

- Adicionar o driver JDBC do Amazon DocumentDB
- Conectar-se ao Amazon DocumentDB usando o Tableau - Túnel SSH

Adicionar o driver JDBC do Amazon DocumentDB

Para se conectar ao Amazon DocumentDB a partir do Tableau Desktop, você deve fazer o download e instalar o driver JDBC do Amazon DocumentDB e o conector DocumentDB Tableau.

1. Faça o download do arquivo JAR do driver JDBC do Amazon DocumentDB no repositório do driver JDBC do Amazon DocumentDB e copie-o em um desses diretórios de acordo com seu sistema operacional:
   - Windows - `C:\Program Files\Tableau\Drivers`
   - MacOS - `~/Library/Tableau/Drivers`
2. Baixe o conector DocumentDB Tableau (um arquivo TACO) do site do Tableau Exchange e copie-o para o diretório Meu Tableau. Repository/Connectors
   - Windows - `C:\Users\[user]\Documents\My Tableau Repository\Connectors`
   - MacOS - `/Users/[user]/Documents/My Tableau Repository/Connectors`

Para obter informações, consulte a Documentação do Tableau.

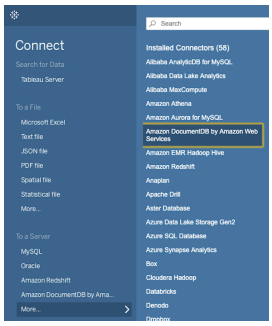
> **Note**
>
> Se você estiver usando certificados CA mais recentes, certifique-se de atualizar seu driver JDBC para a versão v1.4.5 (disponível neste repositório). AWS GitHub

Conectar-se ao Amazon DocumentDB usando o Tableau - Túnel SSH

Para se conectar ao Tableau a partir de uma máquina cliente fora da VPC do seu cluster do DocumentDB, você deve configurar um túnel SSH antes de seguir as etapas abaixo:

1. Inicie a aplicação Tableau Desktop.
2. Navegue até Connect > To A Server > More.

3. Escolha Amazon DocumentDB da Amazon Web Services em Conectores instalados.

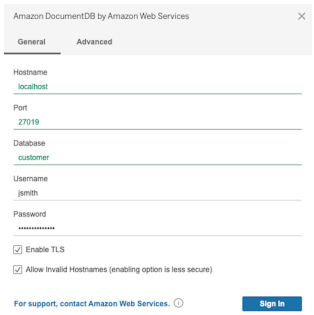


Conexão ao Amazon DocumentDB usando o Tableau - túnel SSH externo

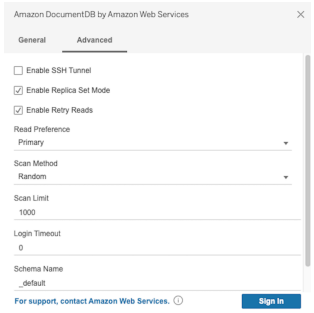
1. Insira os parâmetros de conexão necessários Nome do host, porta, banco de dados, nome de usuário e senha. Os parâmetros de conexão no exemplo a seguir são equivalentes à cadeia de conexão JDBC:

   ```
   jdbc:documentdb://localhost:27019/test?
   tls=true&tlsAllowInvalidHostnames=true&scanMethod=random&scanLimit=1000&logir
   ```

   com os parâmetros de nome de usuário e senha passados separadamente em uma coleção de propriedades. Para obter mais informações sobre os parâmetros da cadeia de conexão, consulte a documentação do github do driver JDBC do Amazon DocumentDB .



2. (Opcional) Opções mais avançadas podem ser encontradas na guia Avançado.

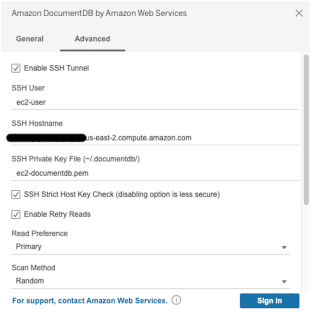


3. Escolha Logon.

Conexão ao Amazon DocumentDB usando o Tableau - túnel SSH interno

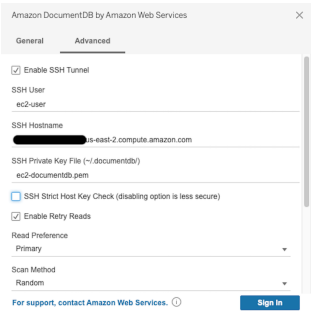
> **Note**
>
> Se você preferir não configurar o túnel SSH usando um terminal, você pode usar a GUI do Tableau para especificar os detalhes da sua instância EC2, que o driver JDBC usará inerentemente para criar um túnel SSH.

1. Na guia Avançado, escolha a opção Ativar túnel SSH para revisar outras propriedades.



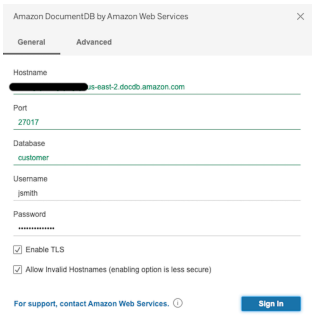
2. Insira o usuário SSH, o nome do host SSH e o arquivo de chave privada SSH.
3. (Opcional) É possível desativar a opção SSH Strict Host Key Check, que ignora a verificação da chave do host em relação a um arquivo host conhecido.

> **Note**
>
> Desativar essa opção é menos seguro, pois pode levar a um ataque intermediário.



4. Insira os parâmetros necessários; Nome do host, porta, banco de dados, nome de usuário e senha.

> **Note**
>
> Certifique-se de usar o endpoint do cluster DocumentDB e não o localhost ao usar a opção de túnel SSH interno.



5. Selecione Fazer login.

Conecte-se ao Amazon DocumentDB a partir de DbVisualizer

Tópicos

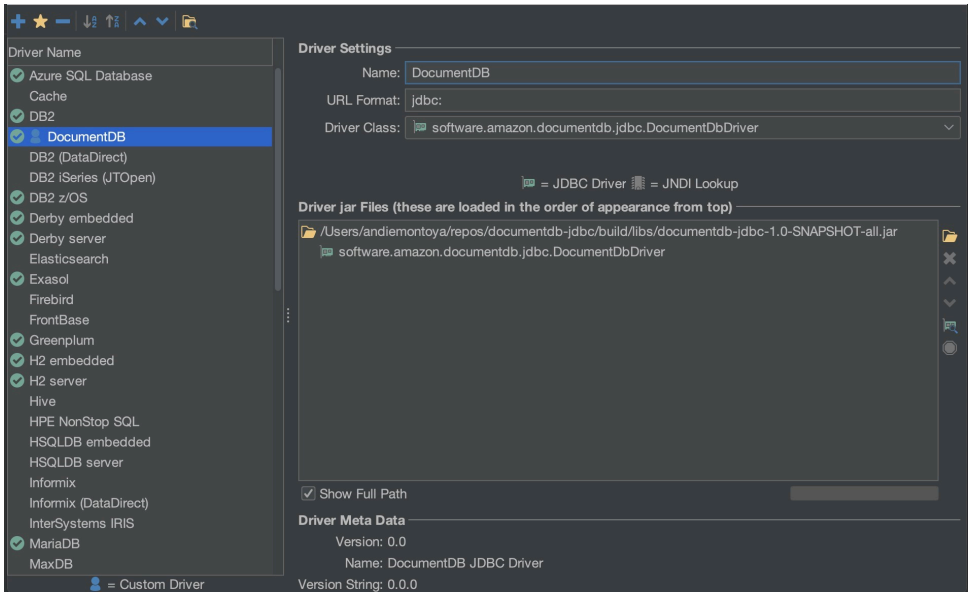
- Adicionar o driver JDBC do Amazon DocumentDB
- Conectando-se ao Amazon DocumentDB usando DbVisualizer

Adicionar o driver JDBC do Amazon DocumentDB

Para se conectar ao Amazon DocumentDB a partir do Amazon DocumentDB, DbVisualizer você deve primeiro importar o driver JDBC do Amazon DocumentDB.

1. Inicie o DbVisualizer aplicativo e navegue até o caminho do menu: Ferramentas > Gerenciador de drivers...
2. Escolha + (ou, no menu, selecione Driver > Criar driver).
3. Defina Name (Nome) como `DocumentDB`.
4. Defina o formato do URL como `jdbc:documentdb://<host>[:port]/<database>[?option=value[&option=value[...]]]`
5. Escolha o botão de pasta e, em seguida, selecione o arquivo JAR do driver JDBC Amazon DocumentDB e escolha o botão Abrir.

6. Verifique se o campo Classe do motorista está definido como`software.amazon.documentdb.jdbc.DocumentDbDriver`. As configurações do Driver Manager do DocumentDB devem ser como o exemplo a seguir.



7. Feche o diálogo. O driver JDBC do Amazon DocumentDB estará configurado e pronto para uso.

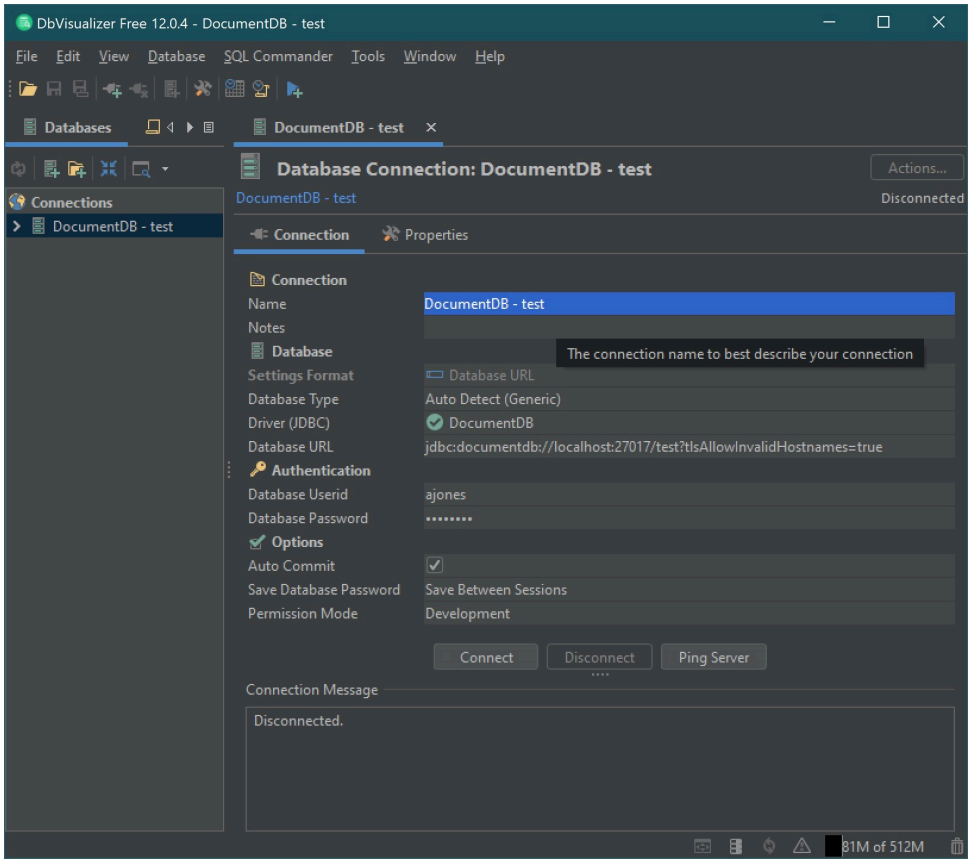
Conectando-se ao Amazon DocumentDB usando DbVisualizer

Conecte-se ao Amazon DocumentDB usando DbVisualizer

1. Se você estiver se conectando de fora da VPC do cluster Amazon DocumentDB, certifique-se de ter configurado um túnel SSH.
2. Escolha Banco de dados > Criar conexão de banco de dados no menu de nível superior.
3. Insira um nome descritivo para o campo Name.
4. Defina o Driver (JDBC) como o driver DocumentDB que você criou na seção anterior.
5. Defina o URL do banco de dados como sua string de conexão JDBC.

   Por exemplo: `jdbc:documentdb://localhost:27017/database?tlsAllowInvalidHostnames=true`
6. Defina Database Userid como sua ID de usuário do Amazon DocumentDB.
7. Defina a Senha do Banco de Dados como a senha correspondente para a ID do usuário.

   A caixa de diálogo de Conexão do Banco de Dados deve ter a seguinte aparência:



8. Selecione Conectar.

Geração automática de esquema JDBC

O Amazon DocumentDB é um banco de dados de documentos e, portanto, não tem o conceito de tabelas e esquemas. No entanto, ferramentas de BI, como o Tableau, esperam que o banco de dados conectado apresente um esquema. Especificamente, quando a conexão do driver JDBC precisar obter o esquema da coleção no banco de dados, ela pesquisará todas as coleções no banco de dados. O driver determinará se já existe uma versão em cache do esquema dessa coleção. Se uma versão em cache não existir, ela fará uma amostra da coleção para documentos e criará um esquema com base no comportamento a seguir.

Tópicos

- Limitações de geração de esquemas
- Opções do método de verificação
- Tipos de dados do Amazon DocumentDB
- Mapeamento de campos de documentos escalares

- Manipulação de tipos de dados de objetos e matrizes

Limitações de geração de esquemas

O driver JDBC DocumentDB impõe um limite no tamanho dos identificadores em 128 caracteres. O gerador de esquema pode truncar o comprimento dos identificadores gerados (nomes de tabelas e nomes de colunas) para garantir que eles se encaixem nesse limite.

Opções do método de verificação

O comportamento da amostragem pode ser modificado usando as opções de cadeia de conexão ou fonte de dados.

- scanMethod=<option>
  - random - (padrão) - Os documentos de amostra são retornados em ordem aleatória.
  - idForward - Os documentos de amostra são devolvidos na ordem de identificação.
  - idReverse - Os documentos de amostra são retornados na ordem inversa da identificação.
  - tudo - Faça uma amostra de todos os documentos da coleção.
- ScanLimit= <n>- O número de documentos a serem amostrados. O valor deve ser um inteiro positivo. O valor padrão é 1000. Se ScanMethod estiver definido como tudo, essa opção será ignorada.

Tipos de dados do Amazon DocumentDB

O servidor do Amazon DocumentDB suporta vários tipos de dados do MongoDB. Listados abaixo estão os tipos de dados compatíveis e seus tipos de dados JDBC associados.

Tipos de dados MongoDB	Compatível com o DocumentDB	Tipo de dados JDBC
Dados binários	Sim	VARBINARY
Boolean	Sim	BOOLEAN
Duplo	Sim	DOUBLE
32-bit Integer	Sim	INTEGER

Tipos de dados MongoDB	Compatível com o DocumentDB	Tipo de dados JDBC
64-bit Integer	Sim	BIGINT
String	Sim	VARCHAR
ObjectId	Sim	VARCHAR
Data	Sim	TIMESTAMP
Null	Sim	VARCHAR
Expressão Regular	Sim	VARCHAR
Timestamp	Sim	VARCHAR
MinKey	Sim	VARCHAR
MaxKey	Sim	VARCHAR
Objeto	Sim	tabela virtual
Array	Sim	tabela virtual
Decimal128	Não	DECIMAL
JavaScript	Não	VARCHAR
JavaScript (com escopo)	Não	VARCHAR
Não definido	Não	VARCHAR
Símbolo	Não	VARCHAR
DBPointer (4.0+)	Não	VARCHAR

Mapeamento de campos de documentos escalares

Ao digitalizar uma amostra de documentos de uma coleção, o driver JDBC criará um ou mais esquemas para representar as amostras na coleção. Em geral, um campo escalar no documento é

mapeado para uma coluna no esquema da tabela. Por exemplo, em uma coleção chamada team e em um único documento`{ "_id" : "112233", "name" : "Alastair", "age": 25 }`, isso seria mapeado para o esquema:

Nome da tabela	Nome da coluna	Tipo de dado	Chave
equipe	id da equipe	VARCHAR	PK
equipe	name	VARCHAR	
equipe	idade	INTEGER	

Promoção de conflitos de tipos de dados

Ao digitalizar os documentos de amostra, é possível que os tipos de dados de um campo não sejam consistentes de documento para documento. Nesse caso, o driver JDBC promoverá o tipo de dados JDBC a um tipo de dado comum adequado a todos os tipos de dados dos documentos de amostra.

Por exemplo:

```
{
"_id" : "112233",
"name" : "Alastair", "age" : 25
}

{
"_id" : "112244",
"name" : "Benjamin",
"age" : "32"
}
```

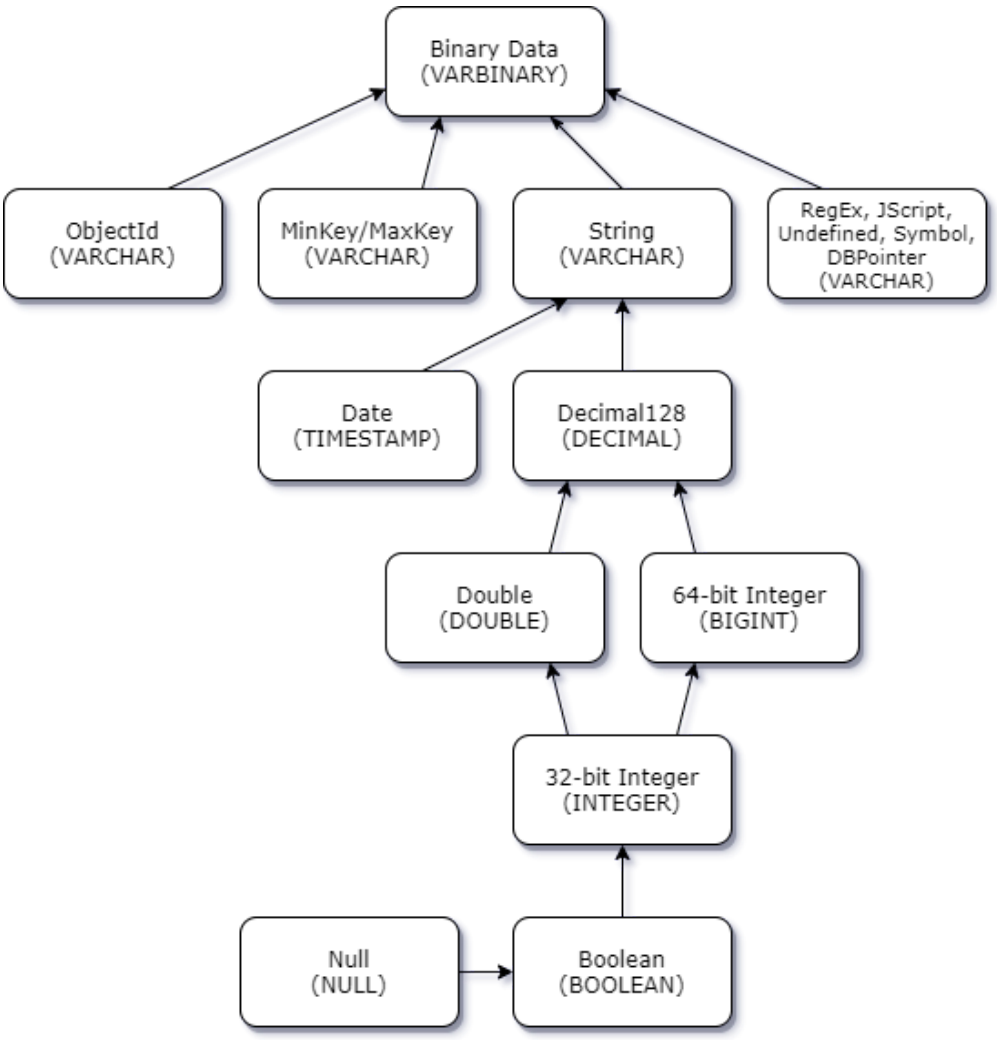
O campo de idade é do tipo inteiro de 32 bits no primeiro documento, mas string no segundo documento. Aqui, o driver JDBC promoverá o tipo de dados JDBC para VARCHAR para lidar com qualquer tipo de dados quando encontrado.

Nome da tabela	Nome da coluna	Tipo de dado	Chave
equipe	id da equipe	VARCHAR	PK

Nome da tabela	Nome da coluna	Tipo de dado	Chave
equipe	name	VARCHAR	
equipe	idade	VARCHAR	

Scalar-scalar promoção de conflitos

O diagrama a seguir mostra a maneira pela qual os conflitos de tipo de dados escalar-escalar são resolvidos.



Scalar-complex tipo de promoção de conflitos

Assim como os conflitos do tipo escalar-escalar, o mesmo campo em documentos diferentes pode ter tipos de dados conflitantes entre complexos (matriz e objeto) e escalares (inteiro, booleano etc.).

Todos esses conflitos são resolvidos (promovidos) à VARCHAR nesses campos. Nesse caso, os dados da matriz e do objeto são retornados como a representação JSON.

Matriz incorporada - Exemplo de conflito de campo de string:

```
{
   "_id":"112233",
   "name":"George Jackson",
   "subscriptions":[
      "Vogue",
      "People",
      "USA Today"
   ]
}
{
   "_id":"112244",
   "name":"Joan Starr",
   "subscriptions":1
}
```

O exemplo anterior mapeia o esquema da tabela customer2:

Nome da tabela	Nome da coluna	Tipo de dado	Chave
customer2	customer2 id	VARCHAR	PK
customer2	name	VARCHAR	
customer2	Assinatura	VARCHAR	

e a tabela virtual customer1_subscriptions:

Nome da tabela	Nome da coluna	Tipo de dado	Chave
customer1_subscriptions	customer1 id	VARCHAR	PK/FK
customer1_subscriptions	subscriptions_index_lvl0	BIGINT	PK

Nome da tabela	Nome da coluna	Tipo de dado	Chave
customer1_subscriptions	valor	VARCHAR	
customer_address	cidade	VARCHAR	
customer_address	region	VARCHAR	
customer_address	país	VARCHAR	
customer_address	código	VARCHAR	

Manipulação de tipos de dados de objetos e matrizes

Até agora, descrevemos apenas como os tipos de dados escalares são mapeados. Os tipos de dados de objeto e matriz são (atualmente) mapeados para tabelas virtuais. O driver JDBC criará uma tabela virtual para representar campos de objetos ou matrizes em um documento. O nome da tabela virtual mapeada concatenará o nome da coleção original seguido pelo nome do campo separado pelo caractere sublinhado (“_”).

A chave primária da tabela base (“_id”) assume um novo nome na nova tabela virtual e é fornecida como uma chave externa para a tabela base associada.

Para campos do tipo matriz incorporada, as colunas de índice são geradas para representar o índice na matriz em cada nível da matriz.

Exemplo de campo de objeto incorporado

Para campos de objeto em um documento, um mapeamento para uma tabela virtual é criado pelo driver JDBC.

```
{
   "Collection: customer",
   "_id":"112233",
   "name":"George Jackson",
   "address":{
      "address1":"123 Avenue Way",
      "address2":"Apt. 5",
      "city":"Hollywood",
```

```
      "region":"California",
      "country":"USA",
      "code":"90210"
   }
}
```

O exemplo anterior mapeia o esquema da tabela de clientes:

Nome da tabela	Nome da coluna	Tipo de dado	Chave
customer	customer id	VARCHAR	PK
customer	name	VARCHAR	

e a tabela virtual customer_address:

Nome da tabela	Nome da coluna	Tipo de dado	Chave
customer_address	customer id	VARCHAR	PK/FK
customer_address	address1	VARCHAR	
customer_address	address2	VARCHAR	
customer_address	cidade	VARCHAR	
customer_address	region	VARCHAR	
customer_address	país	VARCHAR	
customer_address	código	VARCHAR	

Exemplo de campo de matriz incorporada

Para campos de matriz em um documento, um mapeamento para uma tabela virtual também é criado pelo driver JDBC.

```
{
   "Collection: customer1",
```

```
   "_id":"112233",
   "name":"George Jackson",
   "subscriptions":[
      "Vogue",
      "People",
      "USA Today"
   ]
}
```

O exemplo anterior mapeia o esquema da tabela customer1:

Nome da tabela	Nome da coluna	Tipo de dado	Chave
customer1	customer1 id	VARCHAR	PK
customer1	name	VARCHAR	

e a tabela virtual customer1_subscriptions:

Nome da tabela	Nome da coluna	Tipo de dado	Chave
customer1_subscriptions	customer1 id	VARCHAR	PK/FK
customer1_subscriptions	subscriptions_index_lvl0	BIGINT	PK
customer1_subscriptions	valor	VARCHAR	
customer_address	cidade	VARCHAR	
customer_address	region	VARCHAR	
customer_address	país	VARCHAR	
customer_address	código	VARCHAR	

Suporte e limitações do SQL

O driver JDBC do Amazon DocumentDB é um driver somente para leitura que oferece suporte a um subconjunto e a algumas extensões comuns. SQL-92 Consulte a documentação de limitações de SQL e a documentação de limitações de JDBC para obter mais informações.

Solução de problemas

Se você estiver tendo problemas ao usar o driver JDBC do Amazon DocumentDB, consulte o Guia de solução de problemas.

Conecte-se usando o driver ODBC do Amazon DocumentDB

O driver ODBC para o Amazon DocumentDB fornece uma SQL-relational interface para desenvolvedores e permite a conectividade a partir de ferramentas de BI, como o Power BI Desktop e o Microsoft Excel.

Para obter informações mais detalhadas, consulte a documentação do driver ODBC do Amazon DocumentDB em. GitHub

Tópicos

- Introdução
- Configurar o driver ODBC do Amazon DocumentDB no Windows
- Conecte-se ao Amazon DocumentDB a partir do Microsoft Excel
- Conecte-se ao Amazon DocumentDB a partir do Microsoft Power BI Desktop
- Geração automática de esquemas
- Suporte e limitações do SQL
- Solução de problemas

Introdução

Etapa 1. Criar clusters do Amazon DocumentDB

Se você ainda não tem um cluster do Amazon DocumentDB, há várias maneiras de começar.

> **Note**
>
> O Amazon DocumentDB é uma nuvem privada virtual (VPC) apenas de serviço. Se você estiver se conectando de uma máquina local fora da VPC do cluster, precisará criar uma conexão SSH com uma instância do Amazon EC2. Para obter mais informações, consulte Conecte usando o Amazon EC2. Para obter mais informações sobre tunelamento SSH, consulte Usando um túnel SSH para se conectar ao Amazon DocumentDB.

Etapa 2. Instalação do JRE ou JDK

Dependendo da sua aplicação de BI, talvez seja necessário garantir uma instalação do JRE ou JDK de 64 bits, versão 8 ou posterior, instalada em seu computador. É possível fazer o download do Java SE Runtime Environment 8 aqui.

Etapa 3. Faça o download do driver ODBC do Amazon DocumentDB

Faça o download do driver ODBC do Amazon DocumentDB aqui. Escolha o instalador adequado (por exemplo, documentdb-odbc-1.0.0.msi). Siga o guia de instalação.

Etapa 4. Usar um túnel SSH para se conectar ao Amazon DocumentDB

Os clusters do Amazon DocumentDB são iniciados dentro de uma Amazon Virtual Private Cloud (Amazon VPC). Eles podem ser acessados diretamente pelas instâncias do Amazon EC2 ou por outros AWS serviços implantados na mesma Amazon VPC. Além disso, o Amazon DocumentDB pode ser acessado por instâncias do Amazon EC2 ou outros AWS serviços em diferentes VPCs na mesma Região da AWS ou em outras regiões por meio de emparelhamento de VPCs.

No entanto, suponha que seu caso de uso exija que você (ou sua aplicação) acesse seus recursos do Amazon DocumentDB de fora da VPC do cluster. Esse será o caso da maioria dos usuários que não executam suas aplicações em uma VM na mesma VPC do cluster Amazon DocumentDB. Ao se conectar de fora da VPC, é possível usar o tunelamento SSH (também conhecido como encaminhamento de portas) para acessar seus recursos do Amazon DocumentDB.

Para criar um túnel SSH, você precisa de uma instância do Amazon EC2 em execução na mesma Amazon VPC que seu cluster do Amazon DocumentDB. É possível usar uma instância do EC2 existente na mesma VPC que seu cluster ou criar uma instância. É possível configurar um túnel SSH para o cluster do Amazon DocumentDB `sample-cluster.node.us-east-1.docdb.amazonaws.com` ao executar o seguinte comando em seu computador 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
```

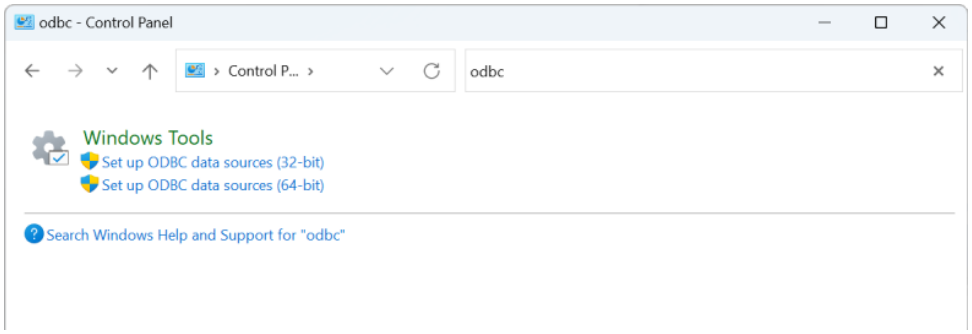
O sinalizador `-L` é usado para encaminhar uma porta local. Esse é um pré-requisito para se conectar a qualquer ferramenta de BI em execução em um cliente fora da sua VPC. Depois de executar a etapa acima, é possível passar para as próximas etapas da ferramenta de BI de sua escolha.

Para obter mais informações sobre tunelamento SSH, consulte a documentação sobre como usar um túnel SSH para se conectar ao Amazon DocumentDB.

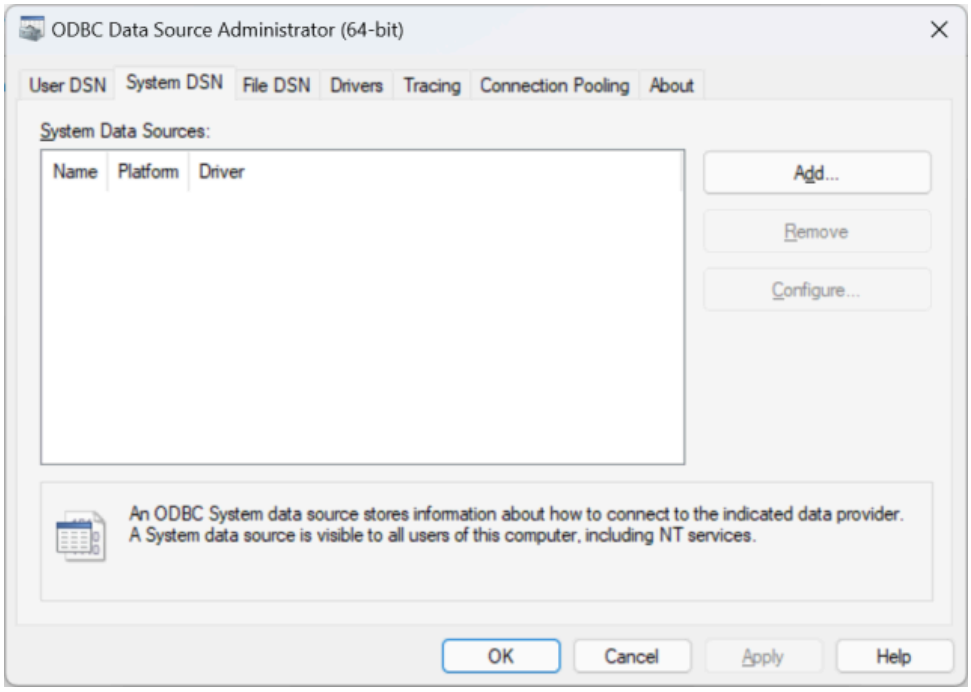
Configurar o driver ODBC do Amazon DocumentDB no Windows

Use o procedimento a seguir para configurar o driver ODBC do Amazon DocumentDB no Windows:

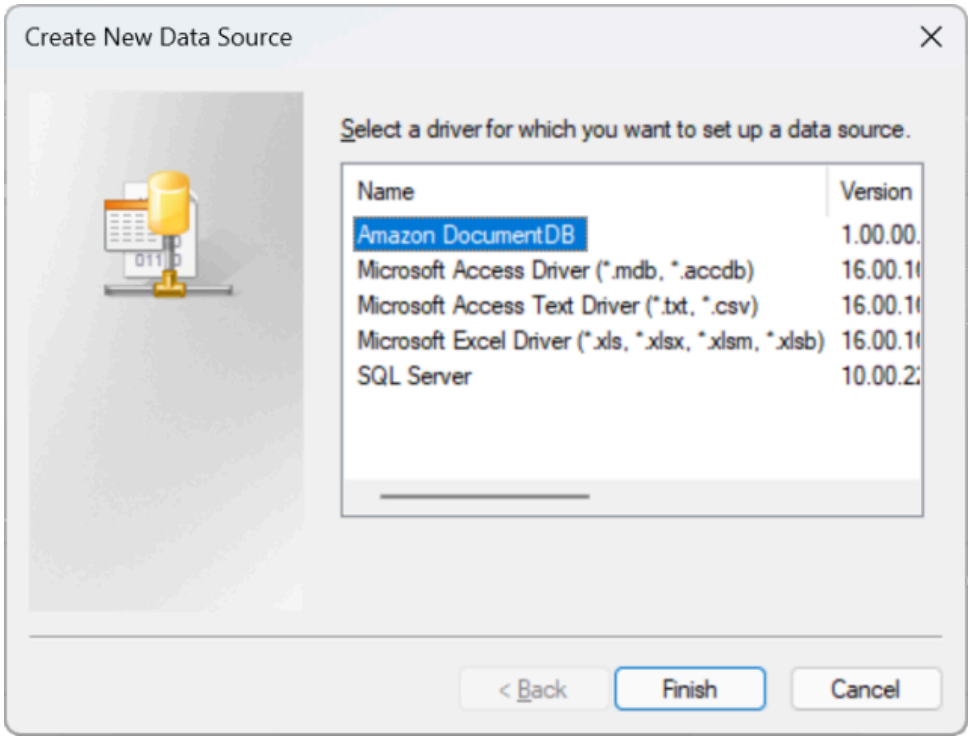
1. Abra o Painel de Controle no Windows e pesquise ODBC (ou, no menu, selecione Ferramentas do Windows > Fontes de dados ODBC (32 bits) ou Fontes de dados ODBC (64 bits)):



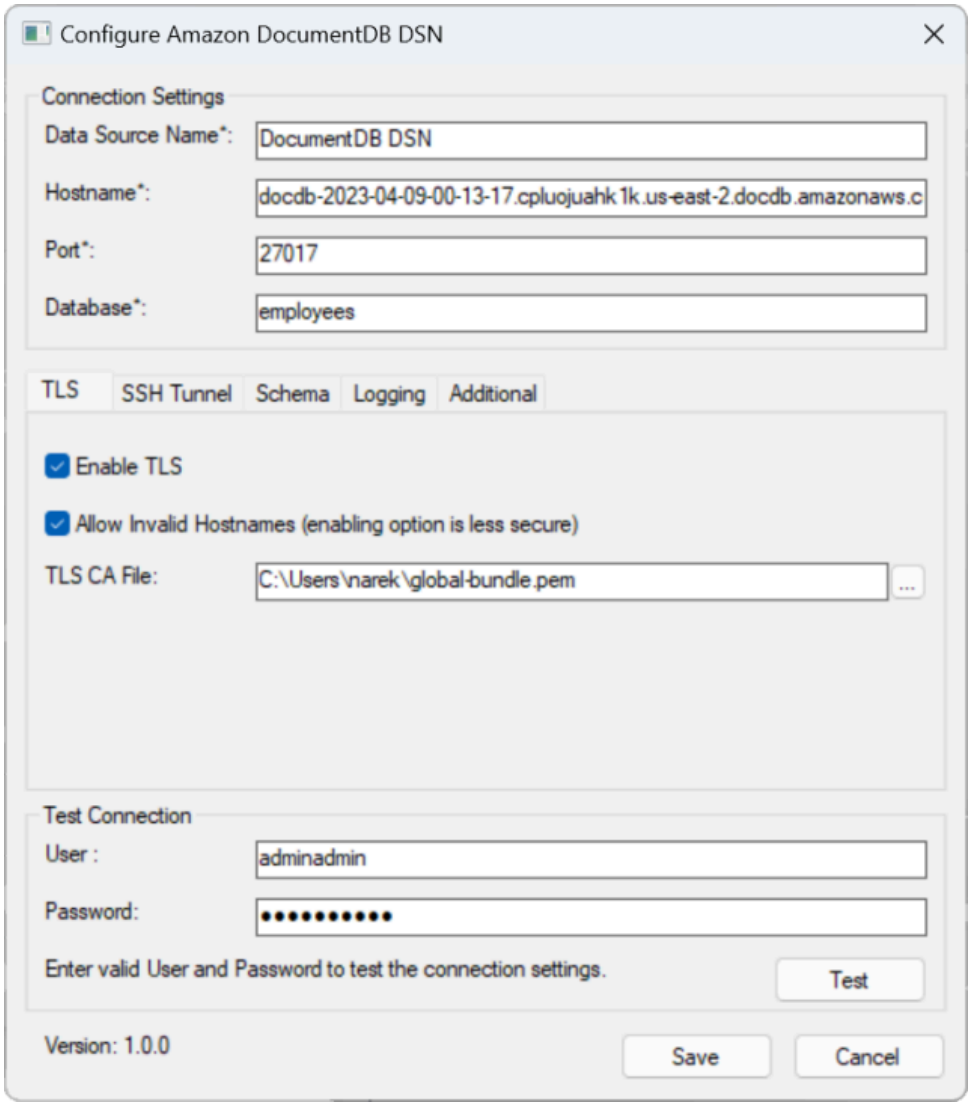
2. Selecione o administrador da fonte de dados do driver ODBC apropriado: opte pela versão de 32 bits se ela estiver instalada; caso contrário, escolha a versão de 64 bits.
3. Selecione a guia DSN do sistema e escolha Adicionar... para adicionar um novo DSN:



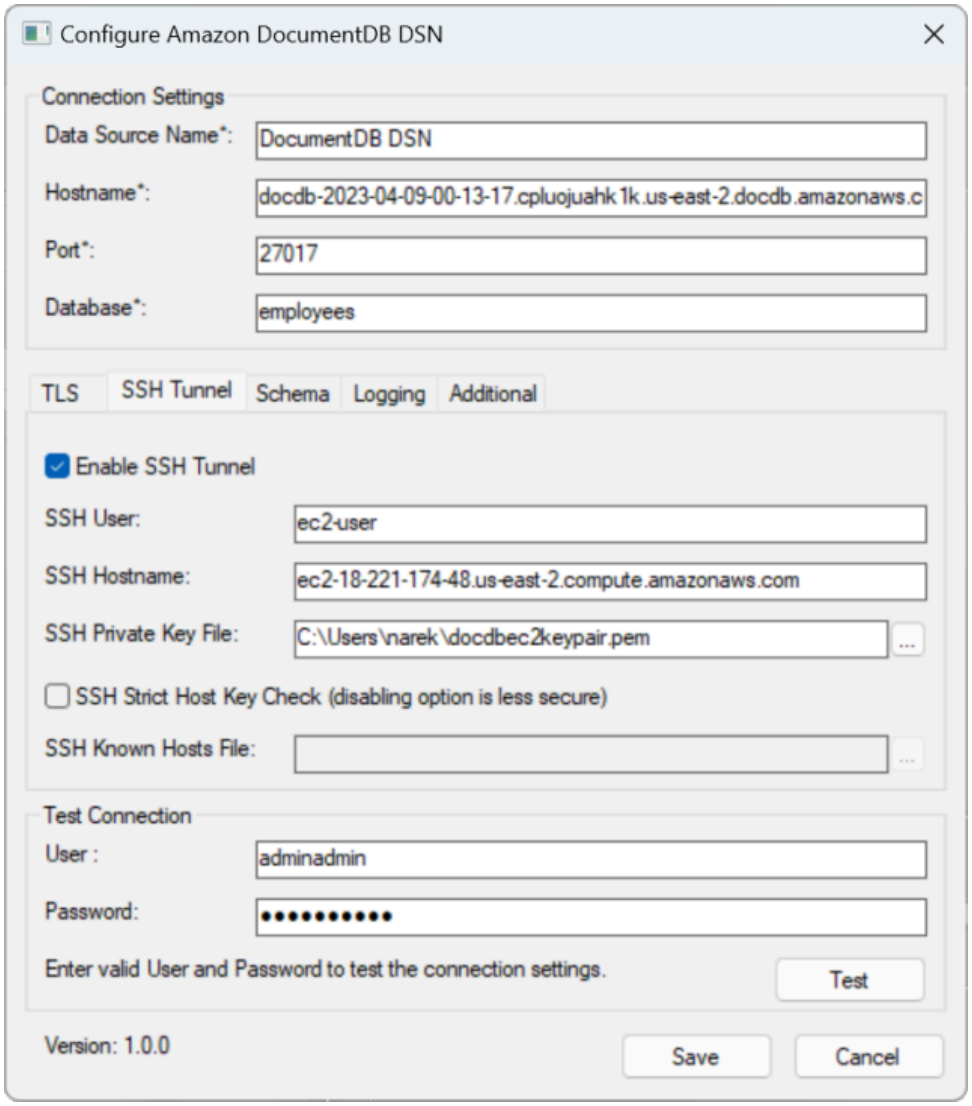
4. Escolha Amazon DocumentDB na lista de drivers da fonte de dados:



5. Na caixa de diálogo Configurar Amazon DocumentDB DSN, preencha os campos Configurações de conexão, guia TLS e Testar conexão e escolha Salvar:



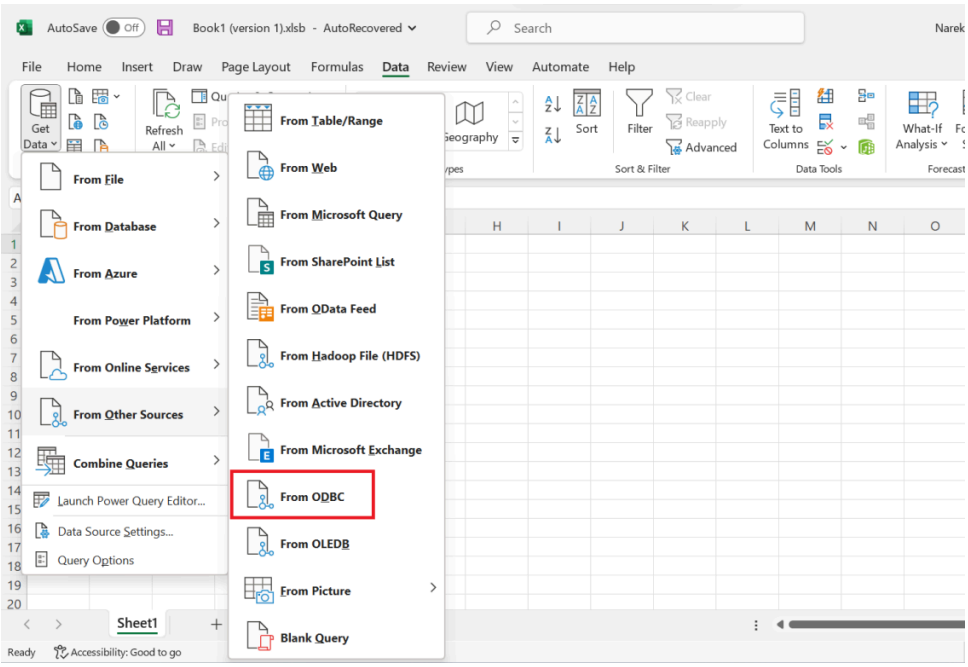
6. Certifique-se de preencher o formulário do Windows com precisão, pois os detalhes da conexão serão diferentes dependendo do método de tunelamento SSH escolhido para a instância EC2. Veja os métodos de tunelamento SSH aqui. Consulte Sintaxe e opções da cadeia de conexão para obter mais informações sobre cada propriedade.



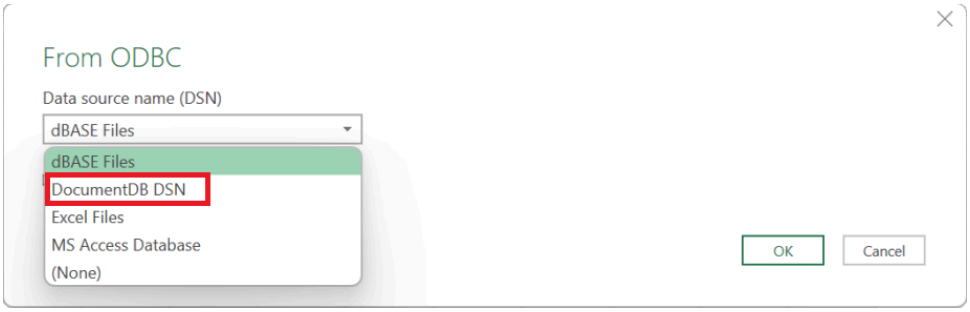
Para obter mais informações sobre como configurar o driver ODBC do Amazon DocumentDB no Windows, consulte aqui. https://github.com/aws/amazon-documentdb-odbc-driver/blob/develop/src/markdown/setup/windows-dsn-configuration.md

Conecte-se ao Amazon DocumentDB a partir do Microsoft Excel

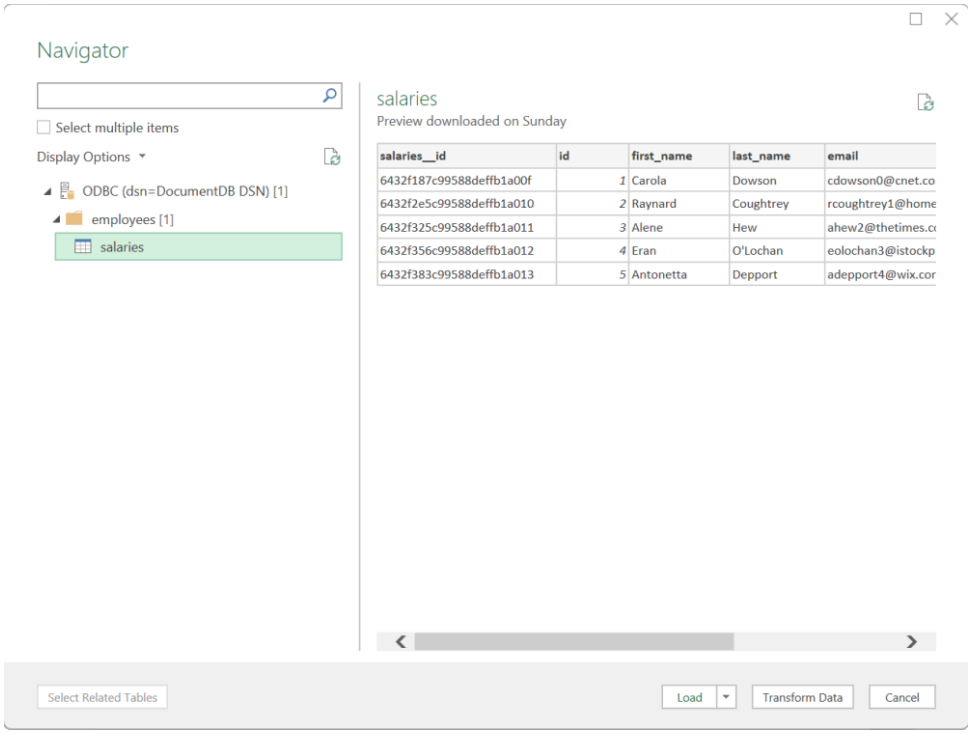
1. Certifique-se de que o driver do Amazon DocumentDB tenha sido instalado e configurado corretamente. Para obter mais informações, consulte Configurar o driver ODBC do Amazon DocumentDB no Windows.
2. Inicie o Microsoft Excel.
3. Navegue até Dados > Obter dados > De outras fontes.
4. Escolha de ODBC:



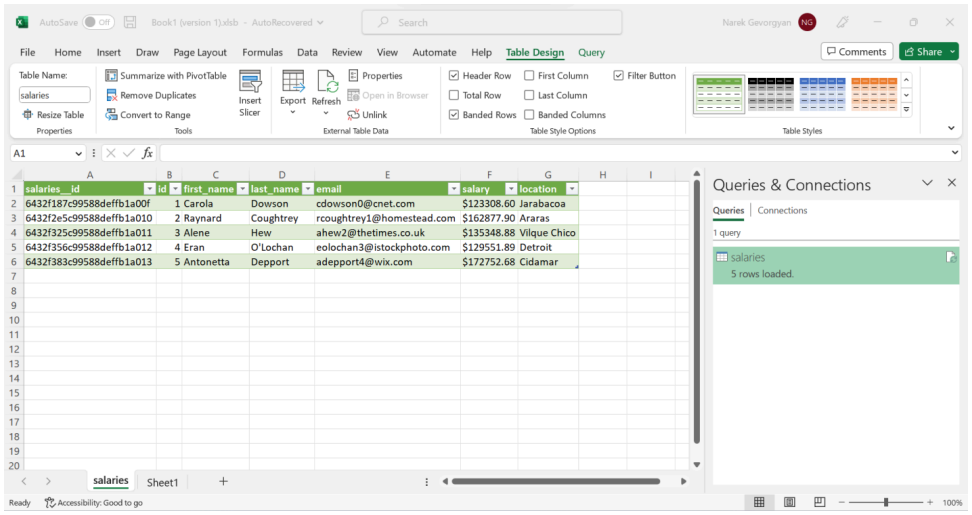
5. Selecione a fonte de dados no menu suspenso Nome da fonte de dados (DSN) associado ao Amazon DocumentDB:



6. Escolha a coleção da qual deseja carregar dados no Excel:



7. Carregar dados no Excel:



Conecte-se ao Amazon DocumentDB a partir do Microsoft Power BI Desktop

Tópicos

- Pré-requisitos
- Adicionar o conector personalizado do Microsoft Power BI Desktop
- Conexão usando o conector personalizado do Amazon DocumentDB
- Configurar o Microsoft Power BI Gateway

Pré-requisitos

Antes de começar, certifique-se de que o driver ODBC do Amazon DocumentDB esteja instalado corretamente.

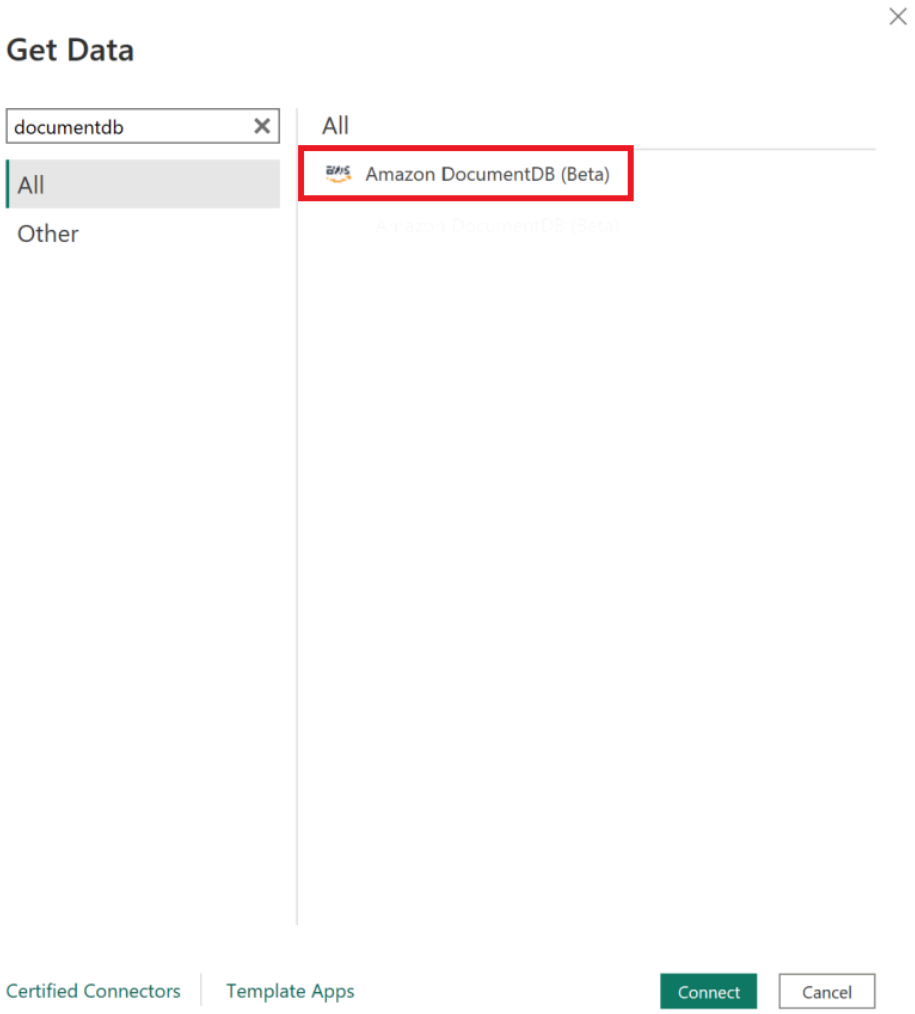
Adicionar o conector personalizado do Microsoft Power BI Desktop

Copie o `AmazonDocumentDBConnector.mez` arquivo para a `<User>\Documents\Power BI Desktop\Custom Connectors\` pasta (ou para, `<User>\OneDrive\Documents\Power BI Desktop\Custom Connectors` se estiver usando OneDrive). Isso permitirá que o Power BI acesse o conector personalizado. É possível obter o conector para o Power BI Desktop aqui. Reinicie o Power BI Desktop para garantir que o conector esteja carregado.

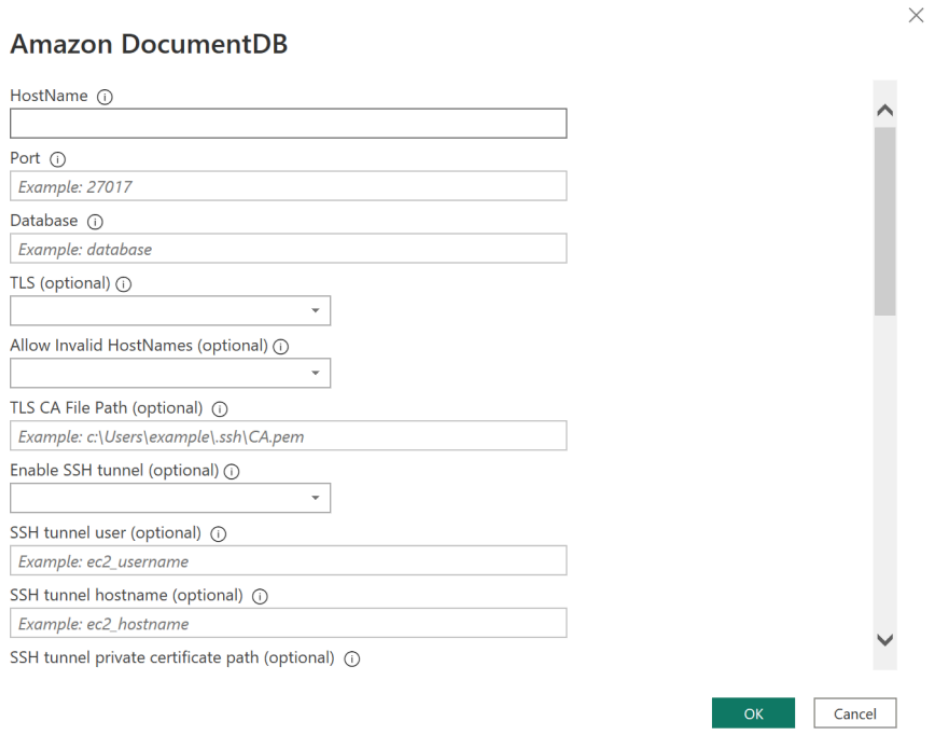
> **Note**
>
> O conector personalizado só oferece suporte ao nome de usuário e senha do Amazon DocumentDB para autenticação.

Conexão usando o conector personalizado do Amazon DocumentDB

1. Selecione Amazon DocumentDB (Beta) em Obter dados e escolha Conectar. Se você receber um aviso por usar um serviço de terceiros, escolha Continuar.



2. Insira todas as informações necessárias para se conectar ao seu cluster Amazon DocumentDB e escolha OK:



> **Note**
>
> Dependendo da configuração do nome da fonte de dados (DSN) do driver ODBC, a tela de detalhes da conexão SSH pode não ser exibida se você já tiver fornecido as informações necessárias nas configurações do DSN.

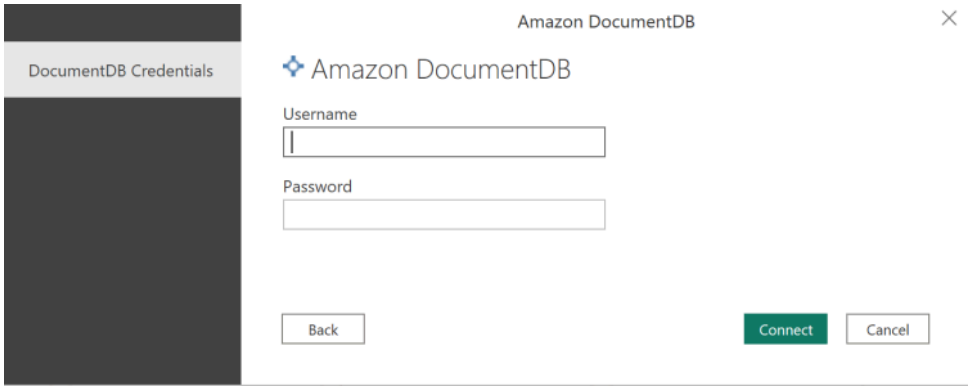
3. Escolha o modo de conectividade de dados:

   - Importar - carrega todos os dados e armazena as informações no disco. Os dados devem ser atualizados e recarregados para mostrar as atualizações de dados.
   - Consulta direta - não carrega dados, mas faz consultas em tempo real sobre os dados. Isso significa que os dados não precisam ser atualizados e recarregados para mostrar as atualizações de dados.

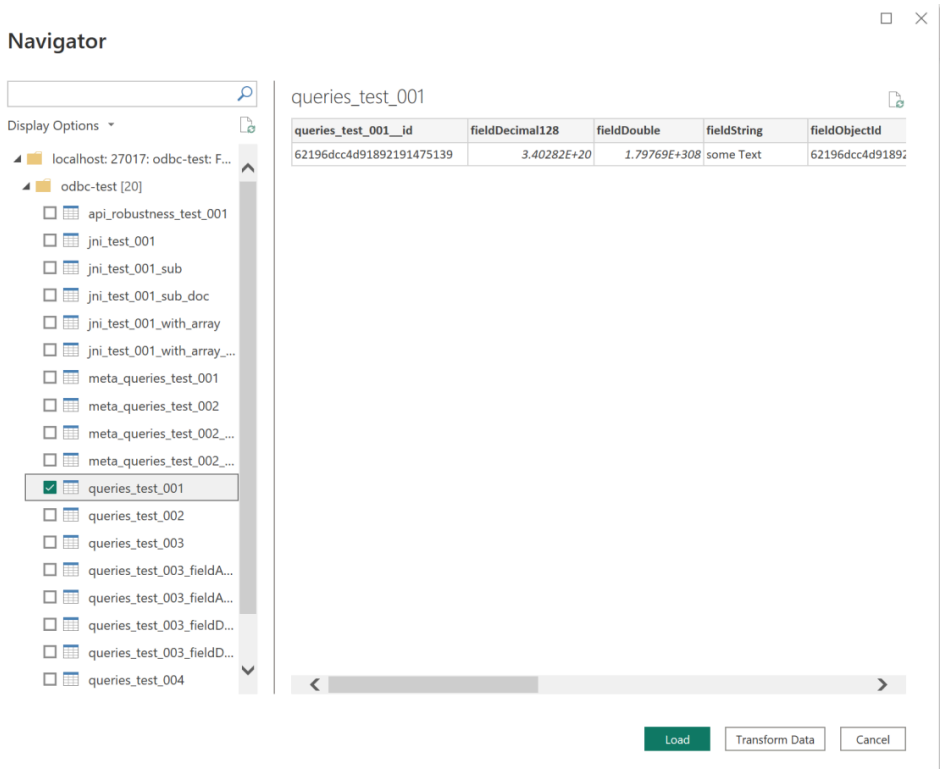


> **Note**
>
> Se você estiver usando um conjunto de dados muito grande, a importação de todos os dados pode levar mais tempo.

4. Se for a primeira vez que você se conecta a essa fonte de dados, selecione o tipo de autenticação e insira suas credenciais quando solicitado. Em seguida, escolha Conectar:



5. Na caixa de diálogo Navegador, selecione as tabelas de banco de dados desejadas e escolha Carregar para carregar os dados ou Transformar dados para continuar transformando os dados.



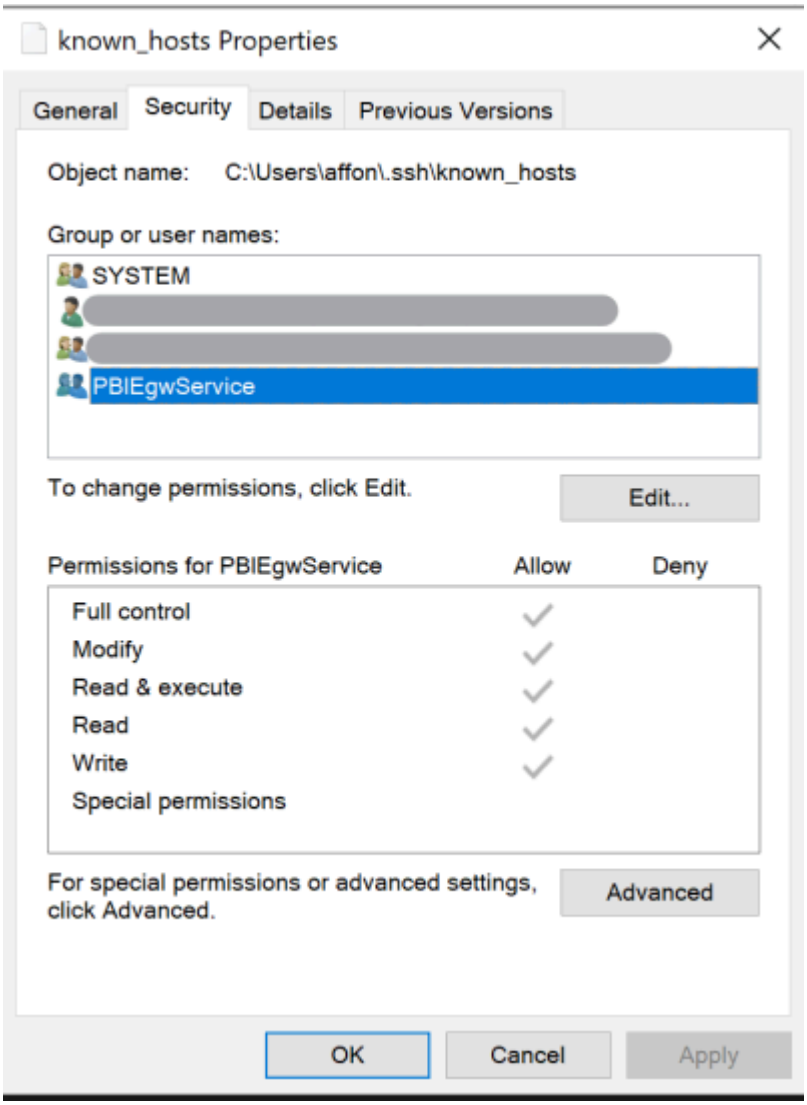
> **Note**
>
> As configurações da fonte de dados são salvas quando você se conecta. Para modificá-las, selecione Transformar dados > Configurações da fonte de dados.

Configurar o Microsoft Power BI Gateway

Pré-requisitos:

- Verifique se o conector personalizado funcionará com o Power BI Gateway.
- Verifique se o DSN ODBC foi criado nas fontes de dados ODBC na guia Sistema na máquina em que o Power BI Gateway está instalado.

Se você estiver usando o atributo de túnel SSH interno, o arquivo `known_hosts` precisará estar localizado onde a conta de serviço do Power BI tenha acesso a ele.



> **Note**
>
> Isso também se aplica a qualquer arquivo que você possa precisar para estabelecer uma conexão com seu cluster Amazon DocumentDB, como um arquivo de certificado de autoridade de certificação (CA) (arquivo pem).

Geração automática de esquemas

O driver ODBC está utilizando o driver JDBC do Amazon DocumentDB por meio da JNI (Java Native Interface), fazendo com que o atributo de geração automática de esquemas funcione de forma semelhante no driver JDBC. Para obter mais informações sobre a geração automática de esquemas,

consulteGeração automática de esquema JDBC. Para obter mais informações sobre a arquitetura do driver ODBC, consulte Arquitetura do driver ODBC do Amazon DocumentDB em. GitHub

Suporte e limitações do SQL

O driver ODBC do Amazon DocumentDB é um driver somente para leitura que suporta um subconjunto e algumas extensões comuns. SQL-92 Consulte a documentação de suporte e limitações do ODBC para obter mais informações.

Solução de problemas

Se você estiver tendo problemas ao usar o driver ODBC do Amazon DocumentDB, consulte o Guia de solução de problemas.

Programação com o Amazon DocumentDB

O serviço oferece suporte à validação do esquema JSON.

Tópicos

- Guia de programação em Java do Amazon DocumentDB
- Usando a validação do esquema JSON

Guia de programação em Java do Amazon DocumentDB

Este guia abrangente fornece instruções detalhadas para o trabalho com o Amazon DocumentDB usando os drivers de Java do MongoDB, abordando aspectos essenciais das operações e do gerenciamento do banco de dados.

Tópicos

- Introdução
- Pré-requisitos
- Modelos de dados para este guia
- Conexão com o Amazon DocumentDB com um driver de Java do MongoDB
- Execução de operações CRUD no Amazon DocumentDB com Java
- Gerenciamento de índices no Amazon DocumentDB com Java
- Event-driven programação com Amazon DocumentDB e Java

Introdução

O guia começa com a conectividade, explicando como estabelecer conexões seguras com clusters do DocumentDB usando o driver de Java do MongoDB. Ele detalha os componentes da cadeia de conexão, a SSL/TLS implementação e várias opções de conexão, incluindo autenticação do IAM e pool de conexões, além de estratégias robustas de tratamento de erros.

Na seção de operações CRUD (criar, ler, atualizar, excluir), o guia aborda detalhadamente a manipulação de documentos, demonstrando como criar, ler, atualizar e excluir documentos usando operações únicas e em massa. Ele explica o uso de filtros, consultas e várias opções de operação, ao mesmo tempo em que enfatiza as melhores práticas para tratamento de erros e implementação da lógica de repetição para melhorar a confiabilidade. O guia também aborda extensivamente o gerenciamento de índices, detalhando a criação e a manutenção de diferentes tipos de índice, incluindo índices de campo único, compostos, esparsos e de texto. Ele explica como otimizar a performance da consulta por meio da seleção adequada do índice e do uso da função `explain()` para analisar os planos de execução da consulta.

A seção final se concentra na programação orientada a eventos usando os fluxos de mudança do Amazon DocumentDB, demonstrando como implementar o monitoramento de alterações de dados em tempo real em aplicações de Java. Ele abrange a implementação de cursores de fluxo de mudança, o manuseio de tokens de retomada para operação contínua e operações baseadas em tempo para processamento de dados históricos. Ao longo do guia, exemplos práticos de código e práticas recomendadas são fornecidos, tornando-o um recurso inestimável para você na criação de aplicações robustas em Java com o Amazon DocumentDB.

Pré-requisitos

Antes de começar, você deve ter o seguinte:

- E Conta da AWS com um cluster Amazon DocumentDB configurado. Para a configuração do cluster, consulte Introdução ao Amazon DocumentDB usando o Amazon EC2 no blog do AWS banco de dados.
- O Java Development Kit (JDK) instalado (usaremos o Amazon Corretto 21 para este guia).
- Maven para o gerenciamento de dependências.

Modelos de dados para este guia

Todo o código de exemplo neste guia pressupõe uma conexão com um banco de dados de teste “ProgGuideData” que tem uma coleção “Restaurantes”. Todos os códigos de amostra neste guia funcionam em um sistema de listagem de restaurantes e, abaixo, há um exemplo da aparência de um documento nesse 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
    }
}
```

Todas as amostras de código que mostram ações CRUD, gerenciamento de índices e programação orientada a eventos pressupõem que você tenha um objeto `MongoClient` `dbClient`, um objeto `MongoDatabase` `connectionDB` e um objeto `MongoCollection` `collection`.

> **Note**
>
> Todos os exemplos de código neste guia foram testados com o driver de Java versão 5.3.0 do MongoDB.

Conexão com o Amazon DocumentDB com um driver de Java do MongoDB

Esta seção fornece um guia passo a passo para conectar-se ao Amazon DocumentDB usando drivers de Java. Isso fará com que você comece a integrar o DocumentDB às suas aplicações de Java.

Tópicos

- Etapa 1: configurar o projeto do
- Etapa 2: criação da string de conexão
- Etapa 3: escrita do código de conexão
- Etapa 4: tratamento de exceções de conexão
- Etapa 5: execução do código
- Práticas recomendadas de conexão

Etapa 1: configurar o projeto do

1. Usando o Maven, crie um projeto de java:

   ```
   mvn archetype:generate -DgroupId=com.docdb.guide -DartifactId=my-docdb-project -
   DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
   ```

2. Adicione o driver de Java do MongoDB como uma dependência para o projeto em seu arquivo 'pom.xml':

```
<dependency>
    <groupId>org.mongodb</groupId>
    <artifactId>mongodb-driver-sync</artifactId>
    <version>5.3.0</version>
</dependency>
```

Etapa 2: criação da string de conexão

A string de conexão do Amazon DocumentDB é essencial para estabelecer uma conexão entre sua aplicação e seu cluster do DocumentDB. Essa string encapsula informações cruciais, como o endpoint do cluster, a porta, os detalhes de autenticação e várias opções de conexão. Para criar uma string do DocumentDB, você normalmente começa com o formato básico:

```
"mongodb://username:password@cluster-endpoint:port/?[connection options]"
```

Você precisará substituir “nome de usuário” e “senha” por suas credenciais reais. Você pode encontrar o endpoint e o número da porta do seu cluster no e Console de gerenciamento da AWS também por meio do AWS CLI. Consulte Localizar os endpoints de um cluster para encontrar o endpoint de cluster para o seu cluster. A porta 27017 é a porta padrão para o DocumentDB.

Exemplos de cadeias de conexão

- Estabelecimento de uma conexão com o DocumentDB usando criptografia em trânsito com garantia de que as solicitações de leitura sejam enviadas para réplicas de leitura e gravem no primário:

```
"mongodb://username:password@cluster-endpoint:27017/?tls=true&
   tlsCAFile=global-bundle.pem&
   readPreference=secondaryPreferred&
   retryWrites=false"
```

- Estabelecimento de uma conexão com o DocumentDB usando a autenticação do IAM:

```
"mongodb://cluster-endpoint:27017/?tls=true&
   tlsCAFile=global-bundle.pem&
   readPreference=secondaryPreferred&
   retryWrites=false&
   authSource=%24external&
   authMechanism=MONGODB-AWS"
```

As diferentes opções disponíveis para a string de conexão são as seguintes:

- Certificado TLS
- Leitura de réplicas de leitura
- Preocupação com a gravação e diário
- RetryWrites
- Autenticação do IAM
- Grupo de conexões
- Parâmetros de tempo limite da conexão

Certificado TLS

`tls=true|false`: esta opção habilita ou desabilita o Transport Layer Security (TLS). Por padrão, a criptografia em trânsito está habilitada no cluster do Amazon DocumentDB e, portanto, a menos que o TLS esteja desabilitado no nível do cluster, o valor dessa opção deve ser `true`.

Ao usar o TLS, o código precisa fornecer um certificado SSL ao criar uma conexão com um cluster do DocumentDB. Baixe o certificado necessário para fazer a conexão segura com o cluster: `global-bundle.pem`. Há duas formas de usar o arquivo `global-bundle.pem`.

- Opção 1: extraia todos os certificados do arquivo `global-bundle.pem` e use a ferramenta de chave do Java para armazená-los em um arquivo `.jks` que possa ser usado posteriormente no código. Consulte a guia Java em Conectar-se com o TLS habilitado para ver o script que mostra como fazer isso.
- Opção 2: adicione dinamicamente o arquivo `global-bundle.pem` no código, crie um armazenamento de chaves na memória e use `SSLContext` para fornecer o certificado como parte da conexão.

Leitura de réplicas de leitura

`replicaSet=rs0&readPreference=secondaryPreferred`: a especificação dessas duas opções encaminha todas as solicitações de leitura para as réplicas de leitura e as solicitações de gravação para a instância primária. O uso de `replicaSet=rs0` na string de conexão permite que o driver do MongoDB mantenha uma visão atualizada automaticamente da topologia do cluster, permitindo que as aplicações mantenham a visibilidade das configurações atuais dos nós à medida que as instâncias são adicionadas ou removidas. Não fornecer essas opções ou especificar

`readPreference=primary` envia todas as leituras e gravações para a instância primária. Para obter mais opções para `readPreference`, consulte Opções de preferência de leitura.

Preocupação com a gravação e diário

A preocupação com a gravação determina o nível de reconhecimento solicitado do banco de dados para operações de gravação. Os drivers do MongoDB oferecem uma opção para ajustar os arquivos de preocupação com a gravação e de diário. O Amazon DocumentDB não espera que você defina a preocupação com a gravação e o diário e ignora os valores enviados para `w` e `j` (`writeConcern` e `journal`). O DocumentDB sempre grava dados com `writeConcern`: `majority` e `journal`: `true` para que as gravações sejam realizadas de forma durável na maioria dos nós antes de enviar uma confirmação para o cliente.

RetryWrites

`retryWrites`— A partir da versão 8.0.2 do mecanismo, o Amazon DocumentDB oferece suporte a gravações que podem ser repetidas. `retryWrites=true`Definido para ativar. Nas versões do mecanismo anteriores à 8.0.2, defina esse atributo `false` como. Para obter mais informações, consulte Gravações que podem ser repetidas no Amazon DocumentDB.

Autenticação do IAM

`authSource=%24external`e **`authMechanism=MONGODB-AWS`** — Esses dois parâmetros são usados para autenticar usando AWS Identity and Access Management. A autenticação do IAM está disponível no momento somente na versão 5.0 do cluster baseado em instâncias. Para obter mais informações, consulte Autenticação usando identidade do IAM.

Grupo de conexões

Essas opções estão disponíveis para agrupamento de conexões:

- **`maxPoolSize`**: define o número máximo de conexões que podem ser criadas no grupo. Quando todas as conexões estão em uso e uma nova solicitação é recebida, ela espera que uma conexão fique disponível. O padrão para drivers de Java do MongoDB é 100.
- **`minPoolSize`**: indica o número mínimo de conexões que devem ser mantidas no grupo o tempo todo. O padrão para drivers de Java do MongoDB é 0.
- **`maxIdleTimeMS`**: determina por quanto tempo uma conexão pode permanecer inativa no grupo antes de ser fechada e removida. O padrão para drivers de Java do MongoDB é de 100 milissegundos.

- **`waitQueueTimeoutMS`**: configura o tempo de espera de um thread para que uma conexão fique disponível quando o grupo estiver em seu tamanho máximo. Se uma conexão não ficar disponível dentro desse período, será lançada uma exceção. O valor padrão para drivers de Java do MongoDB é de 120.000 milissegundos (2 minutos).

Parâmetros de tempo limite da conexão

O tempo limite é um mecanismo para limitar a quantidade de tempo que uma operação ou tentativa de conexão pode levar até que seja considerada falha. Os parâmetros de tempo limite a seguir estão disponíveis para evitar esperas indefinidas e gerenciar a alocação de recursos:

- **`connectTimeoutMS`**: configura o tempo de espera do driver para estabelecer uma conexão com o cluster. O padrão é 10.000 milissegundos (10 segundos).
- **`socketTimeoutMS`**: especifica quanto tempo o driver aguardará por uma resposta do servidor para uma operação de não gravação. O padrão é de 0 (sem tempo limite ou infinito).
- **`serverSelectionTimeoutMS`**: especifica quanto tempo o driver esperará para encontrar um servidor disponível no cluster. O valor padrão dessa configuração é de 30 segundos e é suficiente para que uma nova instância primária seja eleita durante o failover.

Etapa 3: escrita do código de conexão

O exemplo de código a seguir mostra como fazer uma conexão de TLS com o Amazon DocumentDB:

- Ele cria os objetos de Java `KeyStore` e `SSLContext>`.
- Ele também cria o objeto `MongoClientSettings` ao passá-lo para o objeto `ConnectionString`. Para fazer a conexão de TLS, você deve usar o objeto `MongoClientSettings` para vincular a `connectionstring` e o `sslcontext`.
- O uso de `MongoClients` obtém um 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);
    }
}
```

Etapa 4: tratamento de exceções de conexão

Ao trabalhar com o DocumentDB em aplicações de Java, tratar exceções de conexão é crucial para manter operações de banco de dados robustas e confiáveis. Gerenciando adequadamente, essas exceções não apenas ajudam a diagnosticar problemas rapidamente, mas também garantem que sua aplicação possa tratar interrupções temporárias na rede ou indisponibilidade do servidor, melhorando a estabilidade e a experiência do usuário. Algumas das exceções críticas relacionadas ao estabelecimento de conexão incluem:

- **`MongoException`**: uma exceção geral, pode ser emitida em vários cenários não cobertos por exceções mais específicas. Certifique-se de que essa exceção seja tratada após todas as outras exceções específicas, pois essa é uma exceção geral do MongoDB que captura tudo.
- **`MongoTimeoutException`**: emitida quando uma operação expira. Por exemplo, ao consultar um endpoint de cluster inexistente.
- **`MongoSocketException`**: emitida para problemas relacionados à rede. Por exemplo, em uma desconexão repentina da rede durante uma operação.
- **`MongoSecurityException`**: emitida quando a autenticação falha. Por exemplo, ao conectar-se com credenciais incorretas.
- **`MongoConfigurationException`**: emitida quando há um erro na configuração do cliente. Por exemplo, com o uso de uma string de conexão inválida.

Etapa 5: execução do código

O exemplo de código a seguir cria uma conexão do Amazon DocumentDB e imprime todos os bancos de dados:

```
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);
    }
}
```

Práticas recomendadas de conexão

A seguir estão as práticas recomendadas a serem consideradas ao se conectar ao Amazon DocumentDB com um driver de Java do MongoDB:

- Sempre feche o seu `MongoClient` quando não precisar mais que o cliente libere recursos.
- Trate as exceções de forma adequada e implemente o log de erros adequado.
- Use variáveis de ambiente ou AWS Secrets Manager para armazenar informações confidenciais, como nomes de usuário e senhas.

Execução de operações CRUD no Amazon DocumentDB com Java

Esta seção discute a execução da operação CRUD (criar, ler, atualizar, excluir) no Amazon DocumentDB usando drivers de Java do MongoDB.

Tópicos

- Criação e inserção de documentos em uma coleção do DocumentDB
- Leitura e recuperação de dados de uma coleção do DocumentDB
- Atualização de documentos existentes em uma coleção do DocumentDB
- Remoção de documentos de uma coleção do DocumentDB
- Tratamento de erros com lógica de novas tentativas

Criação e inserção de documentos em uma coleção do DocumentDB

A inserção de documentos no Amazon DocumentDB permite que você adicione novos dados às suas coleções. Há várias maneiras de realizar inserções, dependendo das suas necessidades e do volume de dados com os quais você esteja trabalhando. O método mais básico para inserir um documento individual na coleção é `insertOne()`. Para inserir vários documentos ao mesmo tempo, é possível usar o método `insertMany()`, que permite adicionar uma matriz de documentos em uma única operação. Outro método para inserir muitos documentos em uma coleção do DocumentDB é o `bulkWrite()`. Neste guia, discutimos todos esses métodos para criar documentos em uma coleção do DocumentDB.

`insertOne()`

Vamos começar examinando como inserir um documento individual em uma coleção do Amazon DocumentDB. A inserção de um único documento é realizada usando o método `insertOne()`. Esse método usa a BsonDocument para inserção e retorna um `InsertOneResult` objeto que pode ser usado para obter a ID do objeto do novo documento inserido. O código de exemplo abaixo mostra a inserção de um documento de restaurante na coleção:

```
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;
}
```

Ao usar `insertOne()`, certifique-se de incluir o tratamento adequado de erros. Por exemplo, no código acima, "`restaurantId`" tem um índice exclusivo e, portanto, executar esse código novamente gerará a `MongoWriteException` a seguir:

```
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()

Os principais métodos usados para inserir muitos documentos em uma coleção são insertMany() e `bulkWrite()`.

O método `insertMany()` é a maneira mais simples de inserir vários documentos em uma única operação. Ele aceita uma lista de documentos e os insere na coleção. Esse método é ideal quando você insere um lote de novos documentos que são independentes uns dos outros e não exigem nenhum processamento especial ou operações mistas. O código a seguir mostra a leitura de documentos JSON de um arquivo e a inserção deles na coleção. A função `insertMany()` retorna um objeto `InsertManyResult`InsertManyResult que pode ser usado para obter os IDs de todos os documentos inseridos.

```
// 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()`

O método `bulkWrite()` permite realizar várias operações de gravação (inserir, atualizar, excluir) em um único lote. É possível usar `bulkWrite()` quando precisar realizar diferentes tipos de operações em um único lote, como inserir alguns documentos e atualizar outros. `bulkWrite()` oferece suporte a dois tipos de gravação em lote, ordenada e não ordenada:

- Operações ordenadas: (padrão) o Amazon DocumentDB processa as operações de gravação sequencialmente e para no primeiro erro encontrado. Isso é útil quando a ordem das operações é importante, como quando as operações posteriores dependem das anteriores. No entanto, as operações ordenadas geralmente são mais lentas do que as operações não ordenadas. Com as operações ordenadas, você deve resolver o caso em que o lote é interrompido no primeiro erro, potencialmente deixando algumas operações sem processamento.
- Operações não ordenadas: permite que o Amazon DocumentDB processe inserções como uma única execução no banco de dados. Se ocorrer um erro com um documento, a operação continuará com os documentos restantes. Isso é particularmente útil quando você insere grandes quantidades de dados e pode tolerar algumas falhas, como durante a migração de dados ou importações em massa, em que alguns documentos podem falhar devido a chaves duplicadas. Com operações não ordenadas, você deve abordar cenários de sucesso parcial em que algumas operações obtêm êxito e outras falham.

Ao trabalhar com o método `bulkWrite()`, algumas classes essenciais são necessárias. Primeiro, a classe `WriteModel` serve como classe base para todas as operações de gravação e com implementações específicas como `InsertOneModel`, `UpdateOneModel`, `UpdateManyModel`, `DeleteOneModel` e `DeleteManyModel` tratando diferentes tipos de operações.

A `BulkWriteOptions` classe é necessária para configurar o comportamento de operações em massa, como definir a ordered/unordered execução ou ignorar a validação de documentos. A classe `BulkWriteResult` fornece informações detalhadas sobre os resultados da execução, incluindo contagens de documentos inseridos, atualizados e excluídos.

Para o tratamento de erros, a classe `MongoBulkWriteException` é crucial, pois contém informações sobre quaisquer falhas durante a operação em massa, enquanto a classe `BulkWriteError` fornece detalhes específicos sobre falhas de operação individuais. O código a seguir mostra um exemplo de inserção de uma lista de documentos, bem como atualização e exclusão de um único documento, tudo dentro da execução de uma única chamada ao método `bulkWrite()`. O código também mostra como trabalhar com `BulkWriteOptions` e `BulkWriteResult`, bem como o tratamento adequado de erros da operação `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());
}
```

Gravações repetíveis

Diferentemente do MongoDB, o Amazon DocumentDB não oferece suporte a gravações que possam ser repetidas. Como resultado, você deve implementar uma lógica de repetição personalizada em suas aplicações, especialmente para lidar com problemas de rede ou indisponibilidade temporária de serviços. Uma estratégia de repetição bem implementada geralmente envolve aumentar o atraso entre as tentativas e limitar o número total de tentativas. Consulte Tratamento de erros com lógica de novas tentativas abaixo para obter um exemplo de código de criação de lógica de repetição com tratamento de erros.

Leitura e recuperação de dados de uma coleção do DocumentDB

A consulta de documentos no Amazon DocumentDB gira em torno de vários componentes principais que permitem recuperar e manipular dados com precisão. O método `find()` é a API de consulta fundamental nos drivers de Java do MongoDB. Ele permite a recuperação de dados complexa com várias opções para filtrar, classificar e projetar resultados. Além do método `find()`, `Filters` e `FindIterable` são dois outros componentes fundamentais que fornecem os blocos de construção para operações de consulta nos drivers de Java do MongoDB.

A classe `Filters` é uma classe utilitária no driver de Java do MongoDB que fornece uma API fluente para criar filtros de consulta. Essa classe oferece métodos estáticos de fábrica que criam instâncias de objetos `Bson` que representam várias condições de consulta. Os métodos mais usados incluem `eq()` para comparações de igualdade `gt()`, `lt()`, `gte()` e `lte()` para comparações numéricas, `and()` e `or()` para a combinação de várias condições, `in()` e `nin()` para testes de associação de matrizes e `regex()` para correspondência de padrões. A classe foi projetada para

ser segura para tipos e fornece melhor verificação de tempo de compilação em comparação com consultas brutas baseadas em documentos, tornando-a a abordagem preferida para criar consultas DocumentDB em aplicações de Java. O tratamento de erros é robusto, com claras exceções lançadas para construções de filtro inválidas.

`FindIterable` é uma interface especializada projetada para lidar com o resultado do método `find()`. Ela fornece um rico conjunto de métodos para refinar e controlar a execução de consultas, oferecendo uma API fluente para o encadeamento de métodos. A interface inclui métodos essenciais de modificação de consultas, como `limit()` para a restrição do número de documentos retornados, `skip()` para paginação, `sort()` para ordenação de resultados, `projection()` para a seleção de campos específicos e `hint()` para a seleção de índices. As operações de lote, ignorar e limitar em `FindIterable` são ferramentas essenciais de paginação e gerenciamento de dados que ajudam a controlar como os documentos são recuperados e processados do banco de dados.

O agrupamento em lotes (`batchSize`) controla quantos documentos o DocumentDB retorna ao cliente em uma única viagem de ida e volta à rede. Quando você define um tamanho de lote, o DocumentDB não retorna todos os documentos correspondentes de uma só vez, mas os retorna em grupos do tamanho de lote especificado.

Ignorar permite que você desvie o ponto de partida de seus resultados, basicamente dizendo ao DocumentDB que pule um número específico de documentos antes de começar a retornar as correspondências. Por exemplo, `skip(20)` ignorará os primeiros 20 documentos correspondentes. Isso é comumente usado em cenários de paginação em que você deseja recuperar páginas subsequentes de resultados.

O limite restringe o número total de documentos que podem ser retornados de uma consulta. Quando você especifica `limit(n)`, o DocumentDB deixará de retornar documentos depois de retornar 'n' documentos, mesmo que haja mais correspondências no banco de dados.

`FindIterable` oferece suporte a padrões de iterador e cursor ao recuperar documentos do Amazon DocumentDB. A vantagem de usar `FindIterable` como iterador é que ela permite o carregamento lento de documentos e só busca documentos quando solicitada pela aplicação. Outro benefício de usar o iterador é que você não é responsável por manter a conexão com o cluster e, portanto, nenhum fechamento explícito da conexão é necessário.

`FindIterable` também fornece suporte ao `MongoCursor`, que permite que padrões de cursor sejam usados ao trabalhar com consultas do Amazon DocumentDB. O `MongoCursor` é uma implementação específica do driver de Java do MongoDB que fornece controle sobre as operações do banco de dados e o gerenciamento de recursos. Ele implementa a interface `AutoCloseable`,

permitindo o gerenciamento explícito de recursos por meio de blocos de teste com recursos, o que é crucial para fechar adequadamente as conexões do banco de dados e liberar recursos do servidor. Por padrão, o cursor expira em 10 minutos e o DocumentDB não oferece a opção de alterar esse comportamento de tempo limite. Ao trabalhar com dados em lote, certifique-se de recuperar o próximo lote de dados antes que o cursor atinja o tempo limite. Uma consideração importante ao usar o `MongoCursor` é que ele requer um fechamento explícito para evitar vazamentos de recursos.

Nesta seção, vários exemplos são apresentados para `find()`, `Filters` e `FindIterable`.

O exemplo de código a seguir mostra como usar `find()` para recuperar um único documento usando o campo “restaurantId”:

```
Document filter = new Document("restaurantId", "REST-21G145");
Document result = collection.find(filter).first();
```

Embora o uso de `Filters` permita uma melhor verificação de erros no tempo de compilação, o driver de java também permite que você especifique um filtro `Bson` diretamente no método `find()`. O código de exemplo a seguir passa o documento `Bson` para `find()`:

```
result = collection.find(new Document("$and", Arrays.asList(
    new Document("rating.totalReviews", new Document("$gt", 1000)),
    new Document("priceRange", "$$"))))
```

O código de exemplo a seguir mostra vários exemplos de uso da classe `Filters` com `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));
```

O exemplo a seguir mostra como encadear as operações de `sort()`, `skip()`, `limit()` e `batchSize()` em um objeto `FindIterable`. A ordem de como essas operações são fornecidas influenciará a performance da sua consulta. Como prática recomendada, a ordem dessas operações deve ser `sort()`, `projection()`, `skip()`, `limit()` e `batchSize()`.

```
FindIterable < Document > results = collection.find(Filters.gt("rating.totalReviews",
 1000))
    // Sorting
    .sort(Sorts.orderBy(
        Sorts.descending("address.city"),
        Sorts.ascending("cuisine")))
    // Field selection
    .projection(Projections.fields(
        Projections.include("name", "cuisine", "priceRange"),
        Projections.excludeId()))
```

```
    // Pagination
    .skip(20)
    .limit(10)
    .batchSize(2);
```

O exemplo de código a seguir mostra como criar um iterador em `FindIterable`. Ele usa a construção `forEach` do Java para percorrer o conjunto de resultados.

```
collection.find(Filters.eq("cuisine", "American")).forEach(doc ->
 System.out.println(doc.toJson()));
```

No último exemplo de código de `find()`, ele mostra como usar `cursor()` para a recuperação de documentos. Ele cria o cursor no bloco try, o que garante que o cursor seja fechado quando o código sair do bloco try.

```
try (MongoCursor < Document > cursor = collection.find(Filters.eq("cuisine",
 "American"))
    .batchSize(25)
    .cursor()) {
    while (cursor.hasNext()) {
        Document doc = cursor.next();
        System.out.println(doc.toJson());
    }
} // Cursor automatically closed
```

Atualização de documentos existentes em uma coleção do DocumentDB

O Amazon DocumentDB fornece mecanismos flexíveis e poderosos para modificar documentos existentes e inserir novos quando eles não existam. O driver de Java do MongoDB oferece vários métodos para atualizações: `updateOne()` para atualizações de um único documento, `updateMany()` para atualizações de vários documentos e `replaceOne()` para a substituição completa de documentos. Além desses três métodos, `Updates`, `UpdateOptions` e `UpdateResult` são outros componentes fundamentais que fornecem os blocos de construção para operações de atualização nos drivers de Java do MongoDB.

A classe `Updates` no driver de Java do MongoDB é uma classe utilitária que fornece métodos estáticos de fábrica para criar operadores de atualização. Ela serve como o principal construtor para construir operações de atualização de maneira segura e legível. Métodos básicos como `set()`, `unset()`, e `inc()` permitem a modificação direta dos documentos. O poder dessa classe se torna

evidente ao combinar várias operações usando o método `Updates.combine()`, que permite que várias operações de atualização sejam executadas atomicamente, garantindo a consistência de dados.

`UpdateOptions` é uma classe de configuração poderosa no driver de Java do MongoDB, que fornece recursos essenciais de personalização para operações de atualização de documentos. Dois aspectos importantes dessa classe são fornecer suporte a filtros upsert e de matriz para operações de atualização. O recurso upsert, ativado por meio de `upsert(true)`, permite a criação de novos documentos quando nenhum documento correspondente for encontrado durante uma operação de atualização. Por meio de `arrayFilters()`, a operação de atualização pode atualizar com precisão os elementos da matriz que atendem a critérios específicos.

`UpdateResult` no driver de Java do MongoDB, fornece o mecanismo de feedback detalhando o resultado de uma operação de atualização. Essa classe encapsula três métricas principais: o número de documentos correspondidos pelos critérios de atualização (`matchedCount`), o número de documentos realmente modificados (`modifiedCount`) e informações sobre quaisquer documentos que sofreram upsert (`upsertedId`). Compreender essas métricas é essencial para o tratamento adequado de erros, a verificação das operações de atualização e a manutenção da consistência de dados nas aplicações.

Atualização e substituição de um único documento

No DocumentDB, a atualização de um único documento pode ser realizada usando o método updateOne(). Esse método usa um parâmetro de filtro, geralmente fornecido pela classe `Filters`, para identificar o documento a ser atualizado, um parâmetro `Update` que determina quais campos devem ser atualizados e um parâmetro `UpdateOptions` opcional para definir opções diferentes para a atualização. O uso do método `updateOne()` atualizará somente o primeiro documento que corresponda aos critérios de seleção. O código de exemplo a seguir atualiza um único campo de um documento:

```
collection.updateOne(Filters.eq("restaurantId", "REST-Y2E9H5"),
    Updates.set("name", "Amazing Japanese sushi"));
```

Para atualizar vários campos em um documento, use `updateOne()` com `Update.combine()` conforme mostrado no exemplo a seguir. O exemplo também mostra como adicionar um item a uma matriz no 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);
```

O exemplo de código a seguir demonstra como atualizar um documento no banco de dados. Se o documento especificado não existir, a operação o inserirá automaticamente como um novo documento. Esse código também mostra como usar as métricas disponíveis por meio do objeto `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)");
}
```

O exemplo de código a seguir demonstra como substituir completamente um documento existente por um novo usando o método `replaceOne()`, em vez de atualizar campos individuais. O método `replaceOne()` sobrescreve o documento inteiro, preservando somente o campo `_id` do original. Se vários documentos corresponderem aos critérios do filtro, somente o primeiro documento encontrado será substituído.

```
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());
```

Atualização de vários documentos

Há duas maneiras de atualizar vários documentos em uma coleção simultaneamente. É possível usar o método `updateMany()` ou usar `UpdateManyModel` com o método `bulkWrite()`. O método `updateMany()` usa um parâmetro de filtro para selecionar documentos para atualização, o parâmetro `Update` para identificar os campos a serem atualizados e um parâmetro `UpdateOptions` opcional para especificar as opções de atualização.

O exemplo de código a seguir demonstra o uso do 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", "$$$"));
```

O exemplo de código a seguir demonstra o método `bulkWrite()` usando a mesma atualização:

```
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());
```

Remoção de documentos de uma coleção do DocumentDB

O driver de Java do MongoDB oferece `deleteOne()` para a remoção de um único documento e `deleteMany()` para a remoção de vários documentos que correspondam a critérios específicos. Assim como a atualização, a operação de exclusão também pode ser usada com o método `bulkWrite()`. Tanto `deleteOne()` quanto `deleteMany()` retornam um objeto `DeleteResult` que fornece informações sobre o resultado da operação, incluindo a contagem de documentos

excluídos. A seguir há um exemplo de uso de `deleteMany()` para a remoção de vários 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());
```

Tratamento de erros com lógica de novas tentativas

Uma estratégia robusta de tratamento de erros para o Amazon DocumentDB deve implementar a categorização dos erros em passíveis de nova tentativa (como tempos limite de rede, problemas de conexão) e não passíveis de nova tentativa (como falhas de autenticação, consultas inválidas). Para falhas de operação devido a erros que devem ter nova tentativa, ele deve implementar um intervalo de tempo entre cada nova tentativa, bem como o máximo de tentativas de repetição. As operações CRUD devem estar em um bloco try-catch que capture `MongoException` e suas subclasses. Além disso, ele deve incluir monitoramento e registro de erros para visibilidade operacional. Veja a seguir um exemplo de código que mostra como implementar o tratamento de erros com nova tentativa:

```
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);
```

```
    }
}
```

Gerenciamento de índices no Amazon DocumentDB com Java

Os índices permitem a recuperação eficiente de dados de uma coleção do Amazon DocumentDB. Sem índices, o DocumentDB deve digitalizar todos os documentos da coleção para retornar resultados que satisfaçam uma determinada consulta. Este tópico fornece informações sobre como criar, eliminar e listar índices usando os drivers de Java do MongoDB. Ele também discute como determinar se um índice específico está sendo usado na consulta e como dar dicas ao Amazon DocumentDB para usar um índice específico.

Tópicos

- Criação de índices com Java
- Reduzir índices
- Determinação da seleção do índice e fornecimento de dicas de índice

O Amazon DocumentDB oferece suporte a vários tipos de índices. Para uma visão geral abrangente de todos os índices suportados, consulte Como indexar no Amazon DocumentDB no blog do AWS banco de dados.

Criação de índices com Java

Há dois mecanismos para criar índices no Amazon DocumentDB usando drivers de Java do MongoDB: por meio de `runCommand()` e pelo método `createIndex()` para um único índice ou pelo método `createIndexes()` para vários índices. Um motivo para usar os métodos `createIndex()` e `createIndexes()` é que é possível criar um melhor tratamento de erros detectando erros específicos relacionados à criação do índice. Outro motivo para usar esses métodos em vez de `runCommand()` é que o driver de Java do MongDB fornece um rico conjunto de classes de suporte para criação e manipulação de índices. Observe que essas classes de suporte só poderão ser usadas quando você estiver usando os métodos `createIndex()` ou `createIndexes()`. Há três classes de apoio:

- **`Indexes`**: esta classe serve como uma classe utilitária que oferece métodos estáticos de fábrica para criar vários tipos de índices. Ela simplifica o processo de criação de definições de índice complexas e é comumente usada em conjunto com outras classes relacionadas a índices.
- **`IndexModel`**: esta é uma classe fundamental que encapsula a definição das chaves de índice e suas opções. Ela representa uma especificação completa do índice, combinando o que indexar

(as chaves) com a forma de indexar (as opções). Essa classe é particularmente útil ao criar vários índices simultaneamente, pois permite definir uma coleção de especificações de índice que podem ser passadas para o método `createIndexes()`.

- **`IndexOptions`**: esta é uma classe de configuração abrangente que fornece um rico conjunto de métodos para personalizar o comportamento do índice. Ela inclui configurações para índices exclusivos, índices esparsos, tempo de expiração (TTL) e expressões de filtro parcial. Por meio do encadeamento de métodos, é possível configurar várias opções, como a criação de índice em segundo plano e restrições exclusivas.

Criação de um índice único

Este exemplo mostra como criar um índice único usando o método `createIndex(` em segundo plano. Para entender a criação de índices em segundo plano e em primeiro plano, consulte. Tipos de criação de índice O exemplo de código a seguir é usado `IndexOptions` para criar um índice exclusivo com o nome “unique_restaurantId_idx” em segundo plano. Esse objeto `IndexOptions` é então passado ao método `createIndex()`.

```
collection.createIndex(
    Indexes.ascending("restaurantId"),
    new IndexOptions()
        .unique(true)
        .name("unique_restaurantId_idx")
        .background(true));
```

Criação de vários índices

Este exemplo cria vários índices usando o método `createIndexes()`. Primeiro, ele cria a opção para cada índice usando o objeto `IndexModel` e, em seguida, passa uma lista de objetos `IndexModel` para o método `createIndexes()`. O exemplo de código a seguir mostra como criar um índice composto usando a classe de utilitários `Indexes`. Essa classe também é usada para especificar se você deseja criar um índice usando a ordem de classificação crescente ou decrescente. Depois de criar vários índices, ele verifica a criação do índice chamando o método `listIndexes()`.

```
// Single Field Index on cuisine
IndexModel singleIndex = new IndexModel(
    Indexes.ascending("cuisine"),
    new IndexOptions().name("cuisine_idx"));
```

```
// Compound Index
IndexModel compoundIndex = new IndexModel(
    Indexes.compoundIndex(
        Indexes.ascending("address.state"),
        Indexes.ascending("priceRange")),
    new IndexOptions().name("location_price_idx"));

// Build a list of IndexModel for the indexes
List < IndexModel > indexes = Arrays.asList(
    singleIndex,
    compoundIndex
);

collection.createIndexes(indexes);

// Verify created indexes
collection.listIndexes().forEach(index - > System.out.println("Created index: " +
 index.toJson()));
```

Criação de índices esparsos e parciais

Este exemplo mostra a criação de um índice esparso e um parcial com a criação de um `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))));
```

Criação de um índice de texto

Este exemplo mostra como criar e excluir um índice de texto. Somente um índice de texto é permitido em uma coleção, mas esse índice de texto pode ser um índice composto cobrindo vários campos. Ao usar vários campos no índice de texto, você também pode atribuir pesos a cada um dos campos no índice. Não há suporte para os índices de texto em campos de matriz no Amazon DocumentDB e, embora você possa usar até 30 campos no índice de texto composto, somente três campos podem receber um peso.

```
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());
```

Criação de um índice por meio de **`runCommand()`**

O Amazon DocumentDB oferece suporte à criação de índices paralelos para diminuir o tempo necessário para criar índices. A indexação paralela usa vários trabalhadores simultâneos. Os trabalhadores padrão usados para a criação do índice são dois. Para obter mais informações, consulte Desbloqueie o poder da indexação paralela no Amazon DocumentDB no blog do AWS banco de dados. Atualmente, os drivers de Java do MongDB não oferecem suporte à especificação da opção do trabalhador quando você estiver usando `createIndex()` ou `createIndexes()` e, portanto, a única maneira de especificar trabalhadores é por meio de `runCommand`. O exemplo de código a seguir demonstra como usar `runCommand` para criar um índice que aumente o trabalhador para quatro:

```
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);
```

Reduzir índices

O driver de Java do MongoDB fornece vários métodos para eliminar índices, atendendo a diferentes cenários e às suas preferências. É possível eliminar índices por nome, por especificação de chave ou eliminar todos os índices de uma só vez. Os métodos `dropIndex()` e `dropIndexes()` podem ser invocados em um objeto de coleção para eliminar um índice. Ao descartar um índice pelo nome, você deverá garantir que eles usem o nome correto, o que nem sempre é intuitivo, especialmente para índices compostos ou gerados automaticamente. A tentativa de eliminar um índice inexistente resultará em uma `MongoCommandException`. O índice `default _id` não pode ser descartado, pois garante a exclusividade do documento na coleção.

O exemplo de código a seguir mostra como eliminar um índice fornecendo o nome do campo em que o índice foi criado ou excluindo todos os 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();
```

Ao descartar índices usando várias chaves, verifique se há um índice composto contendo todas as chaves especificadas e se a ordem das chaves está correta. O código de exemplo de criação de índice acima mostra uma chave composta sobre “culinária” e recursos. Se você tentar descartar essa chave composta, mas a ordem não for a que foi usada para a criação, ocorrerá um MongoCommnadException erro da seguinte forma:

```
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);
}
```

O seguinte erro é exibido:

```
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
```

Determinação da seleção do índice e fornecimento de dicas de índice

O trabalho com a funcionalidade de explicação no Amazon DocumentDB é fundamental para que você entenda a performance das consultas e o uso de índices. Ao executar uma consulta, é possível acrescentar o método `explain()` para obter informações detalhadas sobre o plano da consulta, incluindo quais índices, se houver, estão sendo usados. O resultado `explain()` fornece informações sobre os estágios de execução da consulta, o número de documentos examinados e o tempo gasto em cada estágio. Essas informações são inestimáveis para identificar se um determinado índice está sendo usado de forma eficaz ou se a consulta pode se beneficiar de uma estrutura de índices diferente.

O método `explain()` pode ser encadeado com o método `find()`. O método `explain()` pode usar uma enumeração `ExplainVerbosity` opcional que determina o nível de verbosidade retornado por `explain()`. No momento, há suporte somente aos enumeradores `EXECUTION_STATS` e `QUERY_PLANNER` no DocumentDB. O exemplo de código a seguir mostra como obter um planejador de consultas para uma 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());
```

O documento JSON a seguir é retornado para o nível de verbosidade do planejador 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
    }
  }
}
```

Você tem várias opções para influenciar ou forçar o Amazon DocumentDB a usar um índice específico. Os métodos `hint()` e `hintString()` permitem que você substitua o comportamento padrão de seleção de índice do otimizador de consultas especificando explicitamente qual índice deve ser usado para uma consulta. Embora o otimizador de consultas do DocumentDB geralmente faça boas escolhas para a seleção de índices, há cenários em que forçar a aprovação de um índice específico por meio de `hint()` ou `hintString()` pode ser benéfico, como ao lidar com dados distorcidos ou ao testar a performance do índice.

O exemplo de código a seguir força o uso do índice composto “cuisine_features_idx” para a mesma consulta que foi executada no código acima:

```
// 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 programação com Amazon DocumentDB e Java

Event-driven a programação no contexto do Amazon DocumentDB representa um poderoso padrão arquitetônico em que as alterações no banco de dados servem como os principais geradores de eventos que acionam a lógica e os processos comerciais subsequentes. Quando os registros são inseridos, atualizados ou excluídos em uma coleção do DocumentDB, essas alterações agem como eventos que iniciam automaticamente vários processos posteriores, notificações ou tarefas de sincronização de dados. Esse padrão é particularmente valioso em sistemas distribuídos modernos, nos quais várias aplicações ou serviços precisam reagir às mudanças de dados em tempo real. O principal mecanismo de implementação da programação orientada a eventos no DocumentDB é por meio de fluxos de alterações.

> **Note**
>
> Este guia pressupõe que você tenha habilitado fluxos de alterações em uma coleção com a qual você esteja trabalhando. Consulte Usar fluxos de alterações com o Amazon DocumentDB para saber como habilitar fluxos de alterações na coleção.

Trabalho com fluxos de alterações da aplicação em Java

O método `watch()` no driver de Java do MongoDB é o principal mecanismo para monitorar alterações de dados em tempo real no Amazon DocumentDB. O método `watch()` pode ser chamado por objetos `MongoClient`, `MongoDatabase` e `MongoCollection`.

O método `watch()` retorna uma instância de `ChangeStreamIterable` que oferece suporte a várias opções de configuração, incluindo pesquisa completa de documentos para atualizações, fornecimento de tokens de retomada e registro de data e hora para fins de confiabilidade e estágios de agregação de pipeline para filtrar alterações.

`ChangeStreamIterable` implementa a interface Java principal `Iterable` e pode ser usada com`forEach()`. Para capturar eventos usando `forEach()`, passe uma função de retorno de chamada para `forEach()` que processe o evento alterado. O trecho de código a seguir mostra

como abrir um fluxo de alterações em uma coleção para iniciar o monitoramento de eventos de alteração:

```
ChangeStreamIterable < Document > iterator = collection.watch();
iterator.forEach(event - > {
    System.out.println("Received a change: " + event);
});
```

Outra forma de percorrer todos os eventos de mudança é abrir um cursor que mantenha uma conexão com o cluster e receba continuamente novos eventos de alteração à medida que eles ocorram. Para obter um cursor de fluxos de alterações, use o método `cursor()` do objeto `ChangeStreamIterable`. O exemplo de código a seguir mostra como monitorar eventos de alteração usando o cursor:

```
try (MongoChangeStreamCursor < ChangeStreamDocument < Document >> cursor =
 collection.watch().cursor()) {
    System.out.println(cursor.tryNext());
}
```

Como prática recomendada, crie o `MongoChangeStreamCursor` em uma instrução try-with-resource ou feche manualmente o cursor. Chamar o método `cursor()` em `ChangeStreamIterable` retornará um `MongoChangeStreamCursor` que é criado sobre um objeto `ChangeStreamDocument`.

A classe `ChangeStreamDocument` é um componente crucial que representa eventos de mudança individuais no fluxo. Ela contém informações detalhadas sobre cada modificação, incluindo o tipo de operação (inserir, atualizar, excluir, substituir), a chave do documento, as informações do namespace e o conteúdo completo do documento, quando disponível. A classe fornece métodos para acessar vários aspectos do evento de alteração, como `getOperationType()` para determinar o tipo de alteração, `getFullDocument()` para acessar o estado completo do documento e `getDocumentKey()` para identificar o documento modificado.

O objeto `ChangeStreamDocument` fornece duas informações importantes, um token de retomada e a hora do evento de mudança.

Os tokens de retomada e as operações baseadas em tempo nos fluxos de alterações do DocumentDB fornecem mecanismos cruciais para manter a continuidade e gerenciar o acesso histórico aos dados. Um token de retomada é um identificador exclusivo gerado para cada evento de alteração, servindo como um marcador que permite que as aplicações reiniciem o processamento do

fluxo de alterações a partir de um ponto específico após desconexões ou falhas. Quando um cursor de fluxo de alterações é criado, ele pode usar um token de retomada armazenado anteriormente por meio da opção `resumeAfter()`, permitindo que o fluxo continue de onde parou, em vez de começar do início ou perder eventos.

Time-based as operações em fluxos de mudança oferecem diferentes abordagens para gerenciar o ponto de partida do monitoramento de eventos de mudança. A opção `startAtOperationTime()` permite que você comece a observar as alterações que ocorreram em ou após um carimbo de data e hora específico. Esses recursos baseados em tempo são particularmente valiosos em cenários que exigem processamento histórico de dados, recuperação pontual ou sincronização entre sistemas.

O exemplo de código a seguir recupera o evento associado ao documento inserido, captura seu token de retomada e, em seguida, fornece esse token para iniciar o monitoramento de eventos após o evento de inserção. O evento é associado ao evento de atualização e, em seguida, obtém a hora do cluster em que a atualização aconteceu e usa esse carimbo de data/hora como ponto de partida para processamento adicional.

```
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());
```

```
  }
```

Por padrão, o evento de atualização de alteração não inclui o documento completo, apenas as alterações que foram feitas. Se você precisar acessar o documento completo que foi atualizado, é possível chamar o método `fullDocument()` no objeto `ChangeStreamIterable`. Lembre-se de que, quando você solicita a devolução de um documento completo para um evento de atualização, ele retorna o documento que existe no momento em que a chamada para alterar fluxos é feita.

Esse método usa uma enumeração `FullDocument` como parâmetro. No momento, o Amazon DocumentDB só oferece suporte a DEFAULT e a valores `UPDATE_LOOKUP`. O trecho de código a seguir mostra como solicitar o documento completo para eventos de atualização ao começar a observar as alterações:

```
try (MongoChangeStreamCursor < ChangeStreamDocument < Document >> cursor =
 collection.watch().fullDocument(FullDocument.UPDATE_LOOKUP).cursor())
```

Usando a validação do esquema JSON

Usando o operador de consulta de avaliação `$jsonSchema`, você pode validar documentos que estão sendo inseridos em suas coleções.

Tópicos

- Criação e uso da validação do esquema JSON
- Palavras-chave com suporte
- ignorar DocumentValidation
- Limitações

Criação e uso da validação do esquema JSON

Criação de uma coleção com validação de esquema

Você pode criar uma coleção com regras de operação e validação do `createCollection`. Essas regras de validação são aplicadas durante inserções ou atualizações de documentos do Amazon DocumentDB. O exemplo de código a seguir mostra as regras de validação para uma coleção de funcionários:

```
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"
} )
```

Inserindo um documento válido

O exemplo a seguir insere documentos que estão em conformidade com as regras de validação de esquema acima:

```
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})
```

Inserindo um documento inválido

O exemplo a seguir insere documentos que não estão em conformidade com as regras de validação de esquema acima. Neste exemplo, o valor do EmployeeID não é uma string:

```
db.employees.insert({
    "name" : { "firstName" : "Carol" , "lastName" : "Smith"},
    "employeeId": 720 ,
    "salary": 1000.0
})
```

Este exemplo mostra a sintaxe incorreta no documento.

Modifica uma coleção.

O comando `collMod` é usado para adicionar ou modificar as regras de validação da coleção existente. O exemplo a seguir adiciona um campo de salário à lista de campos obrigatórios:

```
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
      }
   }
} )
```

Documentos de endereçamento adicionados antes da alteração das regras de validação

Para endereçar documentos que foram adicionados à sua coleção antes da alteração das regras de validação, use os seguintes modificadores `validationLevel`:

- estrito: aplica regras de validação em todas as inserções e atualizações.
- moderado: aplica regras de validação a documentos válidos existentes. Durante as atualizações, os documentos inválidos existentes não são verificados.

No exemplo a seguir, depois de atualizar as regras de validação na coleção chamada “funcionários”, o campo salário é obrigatório. A atualização do seguinte documento falhará:

```
db.runCommand({
    update: "employees",
    updates: [{
        q: { "employeeId": "c721a" },
        u: { age: 25 , salary : 1000},
        upsert: true }]
})
```

O Amazon DocumentDB retorna a seguinte saída:

```
{
"n" : 0,
    "nModified" : 0,
    "writeErrors" : [
        {
"index" : 0,
            "code" : 121,
            "errmsg" : "Document failed validation"
        }
```

```
    ],
    "ok" : 1,
    "operationTime" : Timestamp(1234567890, 1)
}
```

Atualizar o nível de validação para `moderate` permitirá que o documento acima seja atualizado com sucesso:

```
db.runCommand({
    "collMod" : "employees",
    validationLevel : "moderate"
})

db.runCommand({
    update: "employees",
    updates: [{
        q: { "employeeId": "c721a" },
        u: { age: 25 , salary : 1000},
        upsert: true }]
})
```

O Amazon DocumentDB retorna a seguinte saída:

```
{
"n" : 1,
    "nModified" : 1,
    "ok" : 1,
    "operationTime" : Timestamp(1234567890, 1)
}
```

Recuperando documentos com o $jsonSchema

O operador `$jsonSchema` pode ser usado como filtro para consultar documentos que correspondam ao esquema JSON. Esse é um operador de nível superior que pode estar presente em documentos de filtro como um campo de nível superior ou usado com operadores de consulta, como `$and`, `$or` e `$nor`. Os exemplos a seguir mostram o uso de $jsonSchema como um filtro individual e com outros operadores de filtro:

Documento inserido em uma coleção de “funcionários”:

```
{ "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 }
```

Coleção filtrada somente com o operador `$jsonSchema`:

```
db.employees.find({
       $jsonSchema: { required: ["age"] } })
```

O Amazon DocumentDB retorna a seguinte saída:

```
{ "_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 }
```

Coleção filtrada com o operador `$jsonSchema` e outro operador:

```
db.employees.find({
       $or: [{ $jsonSchema: { required: ["age", "name"]}},
            { salary: { $lte:1000}}]});
```

O Amazon DocumentDB retorna a seguinte saída:

```
{ "_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 }
```

Coleção filtrada com o operador `$jsonSchema` e com o `$match` no filtro agregado:

```
db.employees.aggregate(
    [{ $match: {
        $jsonSchema: {
            required: ["name", "employeeId"],
            properties: {"salary" :{"bsonType": "double"}}
```

```
        }
       }
    }]
)
```

O Amazon DocumentDB retorna a seguinte saída:

```
{
"_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
}
```

Visualizando as regras de validação existentes

Para ver as regras de validação existentes em uma coleção, use:

```
db.runCommand({
    listCollections: 1,
    filter: { name: 'employees' }
})
```

O Amazon DocumentDB retorna a seguinte saída:

```
{
    "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)
}
```

O Amazon DocumentDB também mantém as regras de validação no estágio de agregação de `$out`.

Palavras-chave com suporte

Os seguintes campos são compatíveis com os comandos `create` e `collMod`:

- **`Validator`** — Oferece suporte ao operador `$jsonSchem`.
- **`ValidationLevel`** — Oferece suporte aos valores `off`, `strict` e `moderate`.
- **`ValidationAction`** — Oferece suporte ao valor `error`.

O operador $jsonSchema é compatível com as seguintes palavras-chave:

- 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

ignorar DocumentValidation

O Amazon DocumentDB oferece suporte a `bypassDocumentValidation` para os seguintes comandos e métodos:

- `insert`
- `update`
- `findAndModify`
- Estágio `$out` no comando `aggregate` e no método `db.collection.aggregate()`

O Amazon DocumentDB não oferece suporte aos seguintes comandos para `bypassDocumentValidation`:

- `$merge` no comando `aggregate` e no método `db.collection.aggregate()`
- Comando `mapReduce` e método `db.collection.mapReduce()`
- Comando `applyOps`

Limitações

As limitações a seguir se aplicam à validação `$jsonSchema`:

- O Amazon DocumentDB retorna o erro “Falha na validação do documento” quando uma operação falha na regra de validação.
- Os clusters elásticos do Amazon DocumentDB não suportam `$jsonSchema`.

Consultar no Amazon DocumentDB

Esta seção explica todos os aspectos da consulta com o Amazon DocumentDB.

Tópicos

- Consultar documentos
- Plano de consulta
- Explicar os resultados
- Planejador de consultas v2
- Planejador de consultas v3

- Consultar dados geoespaciais com o Amazon DocumentDB
- Índice parcial
- Realizar pesquisa de texto com o Amazon DocumentDB

Consultar documentos

Às vezes, pode ser necessário examinar o inventário da sua loja online para que os clientes possam visualizar e comprar o que você está vendendo. Consultar uma coleção é relativamente fácil, quer você queira todos os documentos na coleção ou apenas os documentos que satisfazem a um critério específico.

Para consultar documentos, use a operação `find()`. O comando `find()` tem um único parâmetro do documento que define os critérios a serem usados na escolha dos documentos a serem retornados. A saída de `find()` é um documento formatado como uma única linha de texto sem quebras de linha. Para formatar o documento de saída para facilitar a leitura, use `find().pretty()`. Todos os exemplos deste tópico usam `.pretty()` para formatar a saída.

Os exemplos de código a seguir usam os quatro documentos que você inseriu na `example` coleção nos dois exercícios anteriores — `insertOne()` e `insertMany()` que estão localizados emAdicionar documentos.

Tópicos

- Recuperar todos os documentos em uma coleção
- Recuperar documentos que correspondem a um valor de campo
- Recuperar documentos que correspondem a um documento incorporado
- Recuperar documentos que correspondem a um valor de campo em um documento incorporado
- Recuperar documentos que correspondem a uma matriz
- Recuperar documentos que correspondem a um valor em uma matriz
- Recuperar documentos usando operadores

Recuperar todos os documentos em uma coleção

Para recuperar todos os documentos em sua coleção, use a operação `find()` com um documento de consulta vazio.

A consulta a seguir retorna todos os documentos da coleção `example`.

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

Recuperar documentos que correspondem a um valor de campo

Para recuperar todos os documentos que correspondem a um campo e valor, use a operação `find()` com um documento de consulta que identifica os campos e valores a serem correspondidos.

Usando documentos anteriores, essa consulta retorna todos os documentos em que o campo "Item" será igual a "Pen".

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

Recuperar documentos que correspondem a um documento incorporado

Para localizar todos os documentos que correspondem a um documento incorporado, use a operação `find()` com um documento de consulta que especifica o nome do documento incorporado e todos os campos e valores desse documento incorporado.

Ao vincular um documento incorporado, o documento incorporado do documento deve ter o mesmo nome que na consulta. Além disso, os campos e os valores no documento incorporado devem corresponder à consulta.

A consulta a seguir retorna apenas o documento "Poster Paint". Isso ocorre porque a "Caneta" tem valores diferentes para "`OnHand`" e "`MinOnHand`", e a "Tinta spray" tem um campo a mais (`OrderQnty`) que o documento de consulta.

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

Recuperar documentos que correspondem a um valor de campo em um documento incorporado

Para localizar todos os documentos que correspondem a um documento incorporado, use a operação `find()` com um documento de consulta que especifica o nome do documento incorporado e todos os campos e valores desse documento incorporado.

Considerando os documentos anteriores, a consulta a seguir usa "notação de pontos" para especificar o documento incorporado e os campos de interesse. Qualquer documento que seja

correspondente será retornado, independentemente de quais outros campos possam estar presentes no documento incorporado. A consulta retorna "Poster Paint" e "Spray Paint", pois ambos correspondem aos campos e valores especificados.

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

Recuperar documentos que correspondem a uma matriz

Para localizar todos os documentos que correspondem a uma matriz, use a operação `find()` com o nome da matriz de interesse e todos os valores na matriz. A consulta retorna todos os documentos que têm uma matriz com esse nome, com valores idênticos aos da matriz, e na mesma ordem que na consulta.

A consulta a seguir retorna apenas o documento "Pen", pois o "Poster Paint" tem uma cor adicional (White), e "Spray Paint" tem as cores em ordem diferente.

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

Recuperar documentos que correspondem a um valor em uma matriz

Para localizar todos os documentos que possuem um valor de matriz específico, use a operação `find()` com o valor e o nome da matriz de interesse.

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

A operação anterior retorna todos os três documentos, pois cada um deles tem uma matriz chamada `Colors` e o valor `Red` em algum lugar da matriz. Se você especificar o valor "`White`", a consulta retornará apenas "Tinta de cartaz".

Recuperar documentos usando operadores

A consulta a seguir retorna todos os documentos em que o valor "`Inventory.OnHand`" é inferior a 50.

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

Para obter uma lista de operadores de consulta compatíveis, consulte Operadores de consulta e projeção.

Plano de consulta

Como posso ver o **`ExecutionStats de um`** plano de consulta?

Ao determinar por que uma consulta está sendo executada mais lentamente do que o esperado, pode ser útil entender quais são os `executionStats` desse plano de consulta. O `executionStats` fornece o número de documentos retornados de um estágio específico (`nReturned`), a quantidade de tempo de execução gasto em cada estágio (`executionTimeMillisEstimate`) e o tempo necessário para gerar um plano de consulta (`planningTimeMillis`). Você pode determinar os estágios mais demorados da sua consulta para ajudar a concentrar seus esforços de otimização a partir da saída de`executionStats`, conforme mostrado nos exemplos de consulta a seguir. No momento, o parâmetro `executionStats` não é compatível com os comandos `delete` e `update`.

> **Note**
>
> O Amazon DocumentDB emula a API do MongoDB 3.6 em um mecanismo de banco de dados criado para fins específicos que utiliza um sistema de armazenamento distribuído, tolerante a falhas e autorrecuperável. Como resultado, os planos de consulta e a saída de `explain()` podem diferir entre o Amazon DocumentDB e o MongoDB. Os clientes que desejam ter controle sobre seu plano de consulta podem usar o operador `$hint` para impor a seleção de um índice preferencial.

Execute a consulta que você deseja melhorar com o comando `explain()` da seguinte forma.

```
db.runCommand({explain: {query document}}).
explain("executionStats").executionStats;
```

Veja a seguir um exemplo de operação.

```
db.fish.find({}).limit(2).explain("executionStats");
```

A saída dessa operação é semelhante à seguinte.

```
{
    "queryPlanner" : {
        "plannerVersion" : 1,
```

```
        "namespace" : "test.fish",
        "winningPlan" : {
            "stage" : "SUBSCAN",
            "inputStage" : {
                "stage" : "LIMIT_SKIP",
                "inputStage" : {
                    "stage" : "COLLSCAN"
                }
            }
        }
    },
    "executionStats" : {
        "executionSuccess" : true,
        "executionTimeMillis" : "0.063",
        "planningTimeMillis" : "0.040",
        "executionStages" : {
            "stage" : "SUBSCAN",
            "nReturned" : "2",
            "executionTimeMillisEstimate" : "0.012",
            "inputStage" : {
                "stage" : "LIMIT_SKIP",
                "nReturned" : "2",
                "executionTimeMillisEstimate" : "0.005",
                "inputStage" : {
                    "stage" : "COLLSCAN",
                    "nReturned" : "2",
                    "executionTimeMillisEstimate" : "0.005"
                }
            }
        }
    },
    "serverInfo" : {
        "host" : "enginedemo",
        "port" : 27017,
        "version" : "3.6.0"
    },
    "ok" : 1
}
```

Se estiver interessado em ver apenas o `executionStats` da consulta acima, você pode usar o seguinte comando. Para coleções pequenas, o processador de consultas do Amazon DocumentDB pode optar por não usar um índice se os ganhos de desempenho forem insignificantes.

```
db.fish.find({}).limit(2).explain("executionStats").executionStats;
```

Cache do plano de consultas

Para otimizar o desempenho e reduzir a duração do planejamento, o Amazon DocumentDB armazena internamente os planos de consulta em cache. Isso permite que consultas com a mesma forma sejam executadas diretamente usando um plano em cache.

No entanto, esse armazenamento em cache às vezes pode causar um atraso aleatório para a mesma consulta; por exemplo, uma consulta que normalmente leva um segundo para ser executada pode, ocasionalmente, levar dez segundos. Isso ocorre porque, com o tempo, a instância do leitor armazenou em cache várias formas da consulta, consumindo memória. Se você tiver essa lentidão aleatória, não é necessário fazer nada para liberar a memória. O sistema gerenciará o uso da memória para você e, quando a memória atingir um determinado limite, ela será liberada automaticamente.

Explicar os resultados

Se você quiser retornar informações sobre planos de consulta, o Amazon DocumentDB oferece suporte ao modo de verbosidade `queryPlanner`. Os resultados do `explain` retornam o plano de consulta selecionado escolhido pelo otimizador em um formato semelhante ao seguinte:

```
{
   "queryPlanner" : {
      "plannerVersion" : <int>,
      "namespace" : <string>,
      "winningPlan" : {
         "stage" : <STAGE1>,
         ...
         "inputStage" : {
            "stage" : <STAGE2>,
            ...
            "inputStage" : {
               ...
            }
         }
      }
   }
}
```

As seções a seguir definirão os resultados `explain` comuns.

Tópicos

- Estágio de digitalização e filtragem
- Interseção de índices
- União de índices
- Índice múltiplo intersection/union
- Índice composto
- Fase de classificação
- Fase de grupos

Estágio de digitalização e filtragem

O otimizador pode escolher um dos seguintes escaneamentos:

COLLSCAN

Esse estágio é um escaneamento de coleta sequencial.

```
{
    "stage" : "COLLSCAN"
}
```

IXSCAN

Esse estágio verifica as teclas de índice. O otimizador pode recuperar o documento nesse estágio e isso pode resultar em um estágio FETCH anexado posteriormente.

```
db.foo.find({"a": 1})
{
    "stage" : "IXSCAN",
    "direction" : "forward",
    "indexName" : <idx_name>
}
```

FETCH

Se o otimizador recuperou documentos em um estágio diferente do IXSCAN, o resultado incluirá um estágio FETCH. Por exemplo, a consulta IXSCAN acima pode resultar em uma combinação dos estágios FETCH e IXSCAN:

```
db.foo.find({"a": 1})
{
    "stage" : "FETCH",
    "inputStage" : {
        "stage" : "IXSCAN",
        "indexName" : <idx_name>
    }
}
```

O IXONLYSCAN verifica somente a chave de índice. Criar índices compostos não evitará o FETCH.

Interseção de índices

IXAND

O Amazon DocumentDB pode incluir um estágio IXAND com uma matriz InputStages do IXSCAN se puder utilizar a interseção de índices. Por exemplo, podemos ver resultados como:

```
{
    "stage" : "FETCH",
    "inputStage" : {
        "stage" : "IXAND",
        "inputStages" : [
            {
                "stage" : "IXSCAN",
                "indexName" : "a_1"
            },
            {
                "stage" : "IXSCAN",
                "indexName" : "b_1"
            }
        ]
    }
}
```

União de índices

IXOR

Semelhante à interseção de índices, o Amazon DocumentDB pode incluir o estágio `IXOR` com uma matriz `inputStages` para o operador `$or`.

```
db.foo.find({"$or": [{"a": {"$gt": 2}}, {"b": {"$lt": 2}}]})
```

Para a consulta acima, a saída de explicação pode ter a seguinte aparência:

```
{
    "stage" : "FETCH",
    "inputStage" : {
        "stage" : "IXOR",
        "inputStages" : [
            {
                "stage" : "IXSCAN",
                "indexName" : "a_1"
            },
            {
                "stage" : "IXSCAN",
                "indexName" : "b_1"
            }
        ]
    }
}
```

Índice múltiplo intersection/union

O Amazon DocumentDB pode combinar vários estágios de interseção ou união de índices e, em seguida, buscar o resultado. Por exemplo:

```
{
    "stage" : "FETCH",
    "inputStage" : {
        "stage" : "IXOR",
        "inputStages" : [
            {
                "stage" : "IXSCAN",
                ...
```

```
            },
            {
                "stage" : "IXAND",
                "inputStages" : [
                    {
                        "stage" : "IXSCAN",
                        ...
                    },
                    {
                        "stage" : "IXSCAN",
                        ...
                    }
                ]
            }
        ]
    }
}
```

O uso de estágios de interseção ou união de índices não é afetado pelo tipo de índice (esparso, composto, etc.).

Índice composto

O uso do índice composto do Amazon DocumentDB não está limitado aos subconjuntos iniciais dos campos indexados; ele pode usar o índice com a parte do sufixo, mas pode não ser muito eficiente.

Por exemplo, o índice composto de `{ a: 1, b: -1 }` pode oferecer suporte às três consultas abaixo:

```
db.orders.find( { a: 1 } )
```

```
db.orders.find( { b: 1 } )
```

```
db.orders.find( { a: 1, b: 1 } )
```

Fase de classificação

Se houver um índice na(s) chave(s) de classificação solicitada(s), o Amazon DocumentDB poderá usar o índice para obter a ordem. Nesse caso, o resultado não incluirá um estágio `SORT`, mas sim um estágio `IXSCAN`. Se o otimizador preferir uma classificação simples, ele incluirá um estágio como este:

```
{
    "stage" : "SORT",
    "sortPattern" : {
        "a" : 1,
        "b" : -1
    }
}
```

Fase de grupos

O Amazon DocumentDB oferece suporte a duas estratégias de grupo diferentes:

- `SORT_AGGREGATE`: agregação de classificação no disco.
- `HASH_AGGREGATE`: agregação de hash na memória.

Planejador de consultas v2

O novo planejador de consultas do Amazon DocumentDB (planejador versão 2.0) apresenta recursos avançados de otimização de consultas e performance aprimorada. A versão 2.0 do planejador para Amazon DocumentDB 5.0 oferece uma melhoria de performance de até 10 vezes em relação à versão anterior no uso dos operadores `find` e `update` com índices. As melhorias de performance vêm principalmente do uso de planos de indexação mais otimizados e da ativação do suporte à varredura de índice para operadores como operadores de negação (`$neq`, `$nin`) e `$elementMatch` aninhados. As consultas da versão 2.0 do planejador são executadas mais rapidamente por meio de melhores técnicas de estimativa de custos, algoritmos otimizados e estabilidade aprimorada. A versão 2.0 do planejador também oferece suporte a APIs de filtro de cache do plano, o que melhora a estabilidade do planejador. Com esse recurso, o Amazon DocumentDB 5.0 agora oferece a capacidade de selecionar entre as diferentes versões do planejador de consultas.

Tópicos

- Pré-requisitos
- Seleção do planejador versão 2.0 como o planejador de consultas padrão
- Práticas recomendadas
- Limitações
- Melhorias para localizar e atualizar operadores

- Planejar filtros de cache
- Diferenças potenciais de comportamento entre o planejador versão 1.0, 2.0 e o MongoDB
- A versão 2.0 do planejador preenche a lacuna de comportamento com o MongoDB

Pré-requisitos

Os pré-requisitos a seguir se aplicam à versão 2.0 do planejador:

- A versão 2.0 do Planner está disponível em todas as áreas em Regiões da AWS que a versão 5.0 do motor está disponível.
- Para optar por usar a versão 2.0 como planejador de consultas padrão, seu cluster precisa estar na versão de patch do mecanismo 3.0.15902 ou posterior do Amazon DocumentDB versão 5.0. Para ver as etapas de atualização para o patch da versão mais recente do mecanismo, consulte Executar uma atualização de patch para a versão do mecanismo de um cluster.
- Para definir o planejador versão 2.0 como o planejador de consultas padrão, você precisa de permissões do IAM para atualizar os grupos de parâmetros do cluster.
- Para o Amazon DocumentDB 8.0, o planejador versão 3.0 é o planejador de consultas padrão. Para obter mais detalhes, consulte Planejador de consultas v3.

Seleção do planejador versão 2.0 como o planejador de consultas padrão

Use as etapas a seguir para selecionar a versão 2.0 como o planejador de consultas padrão no console ou na CLI:

- Siga as etapas em Modificando os parâmetros de cluster do Amazon DocumentDB para modificar o grupo de parâmetros do seu cluster.
- Para o parâmetro intitulado 'plannerVersion', altere o valor para 2.0 indicando a versão 2.0 do planejador.
- Selecione Aplicar imediatamente (selecionar Aplicar na reinicialização tornará a seleção ineficaz até a próxima reinicialização do cluster).

Práticas recomendadas

Para obter os resultados esperados, use as práticas recomendadas a seguir ao aplicar a versão 2.0 do planejador:

- Em um cluster global, selecione o mesmo `plannerVersion` valor (1,0 ou 2,0) nos grupos de parâmetros do cluster para ambas as regiões. Observe que selecionar diferentes versões do planejador nas regiões primária e secundária pode causar comportamento e desempenho inconsistentes da consulta.
- A atualização para a versão 2.0 do planejador durante uma janela de manutenção programada ou durante períodos de tráfego reduzido será a menos perturbadora, pois pode haver um aumento nas taxas de erro se a versão do planejador for alterada quando as workloads estiverem em execução ativa.
- A versão 2.0 do planejador funciona de forma mais otimizada com a versão 5.0 do shell do MongoDB.

Limitações

As limitações a seguir se aplicam ao planejador versão 2.0:

- Não há suporte para a versão 2.0 do planejador em clusters elásticos, que retornarão à versão 1.0 do planejador.
- Não há suporte para a versão 2.0 do planejador para agregação e comandos distintos, que retornarão à versão 1.0 do planejador.
- Algumas formas de consulta não podem ter um filtro de cache de plano definido nelas. Para ver a lista atual, consulteFormas de consulta suportadas.

Melhorias para **`localizar`** e **`atualizar`** operadores

A versão 2.0 do Planner otimiza as operações fundamentais, incluindo os comandos `find`, `update`, `delete` e `find-and-modify`. As seções com guias a seguir mostram recursos aprimorados para índices, bem como as melhorias na performance da consulta com a versão 2.0 do planejador:

Enhanced index support

- A versão 2.0 do planejador adiciona suporte de índices para operadores de negação, incluindo `$nin`, `$ne`, `$not {eq}` e `$not {in}`, bem como para `$type` e `$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 }}}}})
```

- A versão 2.0 do planejador utiliza índices esparsos ou parciais mesmo quando `$exists` não está presente na expressão 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" })
```

- A versão 2.0 do planejador utilizará índices parciais mesmo quando a condição de consulta não corresponder exatamente à expressão de filtro do 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}})
```

- A versão 2.0 do planejador utiliza varredura parcial de índices com consultas $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}}})
```

- A versão 2.0 do planejador inclui suporte à varredura de índices para `$regex`, sem a necessidade de fornecer `$hint` no código da sua aplicação. `$regex` oferece suporte a índices somente em pesquisas de prefixo.

```
Sample Document: { "x": [1, 2, 3], "y": "apple" }
db.foo.createIndex({ "x": 1, "y": 1 })
db.foo.find({"y":{ $regex: "^a" }})
```

- A versão 2.0 do planejador melhora a performance das consultas envolvendo índices de várias chaves, com condições de igualdade no campo de várias chaves.

```
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)
```

- A versão 2.0 do planejador melhora a performance das consultas que envolvem vários filtros, especialmente em coleções com documentos maiores que 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}}]})
```

- A versão 2.0 do planejador melhora a performance das consultas ao usar o operador `$in` com um índice composto, eliminando o estágio de classificação.

```
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})
```

  Também melhora o desempenho de consultas que usam índices de várias chaves com elementos. `$in`

```
Sample Document: {"x": [1, 2, 3]}
db.foo.createIndex({"x": 1})
db.foo.find("x":{$in:[>100 elements]})
```

Query performance improvements

- A versão 2.0 do planejador melhora a performance das consultas que envolvem vários filtros, especialmente em coleções com documentos maiores que 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}}]})
```

- A versão 2.0 do planejador melhora a performance das consultas ao usar o operador `$in` com um índice composto, eliminando o estágio de classificação.

```
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})
```

  Ela também melhora a performance das consultas que usam índices de várias chaves com elementos `$in`.

```
Sample Document: {"x": [1, 2, 3]}
db.foo.createIndex({"x": 1})
db.foo.find("x":{$in:[>100 elements]})
```

Planejar filtros de cache

A versão 2.0 do Planner adiciona suporte aos filtros de cache do plano, também chamados de filtros de índice, que permitem restringir o conjunto de índices que o planejador considera para uma forma de consulta específica. Os filtros são definidos com um comando de banco de dados e aplicados no servidor. Portanto, se você tiver uma regressão de consulta, poderá atenuá-la sem modificar o código do aplicativo.

Na versão 2.0 do planejador, os filtros de cache do plano se aplicam aos `count` comandos `find` e e, a partir do Amazon DocumentDB 5.0.2 e 8.0.2, também aos comandos`update`, e. `delete` `findAndModify` Os `aggregate` comandos `distinct` e exigem a versão 3.0 do planejador.

Para obter mais informações sobre a referência do comando, formas de consulta suportadas, exemplos resolvidos e como os filtros interagem com as dicas, consultePlanejar filtros de cache.

Diferenças potenciais de comportamento entre o planejador versão 1.0, 2.0 e o MongoDB

Em alguns casos extremos, é possível que a versão 2.0 do planejador produza resultados ligeiramente diferentes dos resultados do MongoDB. Esta seção mostra alguns exemplos dessas possibilidades.

$(update) and $(projection)

- Em alguns casos, os operadores `$(update)` e `$(projection)` do MongoDB podem se comportar de forma diferente da versão 1.0 do planejador do Amazon DocumentDB. Veja adiante alguns exemplos:

```
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 } }
```

  - Planejador versão 1.0: campo de atualizações 2022
  - MongoDB: campo de atualizações 2022
  - Planejador versão 2.0: campo de atualizações 2021

-
```
db.col.insert({x:[1,2,3]})
db.col.update({$and:[{x:1},{x:3}]},{$set:{"x.$":500}})
```

  - Planejador versão 1.0: atualiza aleatoriamente o primeiro elemento correspondente
  - MongoDB: atualiza aleatoriamente o primeiro elemento correspondente
  - Planejador versão 2.0: não faz atualizações

-
```
db.col.insert({x:[1,2,3]})
```

```
db.col.find()
```

- Planejador versão 1.0: seleciona aleatoriamente o elemento correspondente
- MongoDB: seleciona aleatoriamente o elemento correspondente
- Planejador versão 2.0: não faz uma seleção

- ```
  db.col.insert({x:100})
  db.col.update({x:100},{x:100})
  ```

  - Planejador versão 1.0: a contagem nModified é alterada
  - MongoDB: a contagem nModified é alterada
  - Planejador versão 2.0: a contagem nModified não é alterada quando atualizada com o mesmo valor.
- Quando o operador `$(update)` é usado com `$setOnInsert`, o planejador versão 1.0 e o MongoDB geram um erro, mas a versão 2.0 do planejador não.
- Renomear um campo inexistente para `$field` gera um erro na versão 2.0 do planejador, mas não produz atualizações na versão 1.0 do planejador e no MongoDB.

Index behavior

- A versão 2.0 do planejador gera um erro quando `$hint` é aplicado com um índice inadequado, enquanto a versão 1.0 do planejador e o MongoDB não.

  ```
  // 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"
}
```

- `$near` não pode usar `$hint({“$natural”:1})` na versão 2.0 do planejador.

```
// 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" }
```

- Enquanto o MongoDB oferece suporte a varreduras completas de índices regex, a versão 2.0 do planejador oferece suporte à varredura de índices regex somente em campos de prefixo.

```
// 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)
```

- Existem algumas diferenças inerentes ao uso de filtros de cache de plano com o planejador versão 2.0 em comparação com o MongoDB. Embora a versão 2.0 do planejador não ofereça suporte à especificação de “projeção” e “agrupamento” com filtros de cache de plano, o MongoDB oferece. No entanto, o filtro de índices do MongoDB está somente na memória e é perdido após a reinicialização. A versão 2.0 do planejador persiste os filtros de índice ao longo de reinicializações e aplicações de patches.

Others

- O formato dos logs de auditoria do DML ao usar a versão 2.0 do planejador varia um pouco em relação à versão 1.0 do planejador.

```
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}}
```

- A condição do índice como parte do plano de explicação:

```
rs0:PRIMARY> db.col.createIndex({index1:1})
{
 "createdCollectionAutomatically" : false,
 "numIndexesBefore" : 1,
 "numIndexesAfter" : 2,
 "ok" : 1,
 "operationTime" : Timestamp(1761149251, 1)
}
```

A versão 2.0 do planejador explica a saída do plano exibindo a condição e o filtro do índice:

```
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)
}
```

A versão 1.0 do planejador explica a saída do plano:

```
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)
}
```

A versão 2.0 do planejador preenche a lacuna de comportamento com o MongoDB

Há algumas áreas em que a versão 2.0 do planejador preenche as lacunas de comportamento do MongoDB:

- A versão 2.0 do planejador permite a pesquisa de índice numérico em matrizes achatadas 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}}})
```

- Enquanto a versão 1.0 do planejador excluiu strings na projeção, o comportamento do planejador versão 2.0 se alinha ao MongoDB e as trata como valores literais

```
// 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 }
```

- A versão 2.0 do planejador, como o MongoDB, não permite projeção nos mesmos campos “x” e “x.a”:

```
// Planner version 2/MongoDB will error out
> db.col.find()
{ "_id" : ObjectId("68153da2012265816bc9ba23"), "x" : [ { "a" : 1 }, 3 ] }
 db.col.find({},{"x.a":1,"x":1}) // error

// Planner Version 1 does not error out
db.col.find()
{ "_id" : ObjectId("68153da2012265816bc9ba23"), "x" : [ { "a" : 1 }, 3 ] }

db.col.find({},{"x.a":1,"x":1})
{ "_id" : ObjectId("68153d60143af947c720d099"), "x" : [ { "a" : 1 }, 3 ] }
```

- A versão 2.0 do planejador, como o MongoDB, permite a projeção em 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"
}
```

- Na versão 2.0 do planejador, como o MongoDB, a projeção não oferece suporte a campos após o 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 } ] }
```

- A versão 2.0 do planejador, como o MongoDB, permite o 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 }
```

Planejador de consultas v3

A versão 3 do Planner no Amazon DocumentDB 8.0 suporta 21 estágios de agregação, incluindo 6 novos estágios. O Planner V3 vem com suporte embutido para comandos distintos. Ele oferece uma melhoria de desempenho geral de até duas vezes em relação ao Planner v2 no Amazon DocumentDB 5.0. Todos os novos recursos e operadores do Amazon DocumentDB 8.0 são compatíveis com o Planner v3. Os novos estágios de agregação no Amazon DocumentDB 8.0 compatíveis com o Planner v3 incluem $replaceWith, $vectorSearch, $merge, $set, $unset, $bucket. O Planner v3 também oferece suporte a novos recursos e operadores no Amazon DocumentDB 8.0, incluindo Collation, Views, $merge, $pow, $rand, $dateTrunc, $date, $date. ToParts FromParts

Tópicos

- Pré-requisitos
- Selecionar o planejador versão 3.0 como o planejador de consultas padrão
- Práticas recomendadas

- Limitações
- Melhorias em operadores agregados e distintos
- Planejar filtros de cache
- Possíveis diferenças de comportamento entre o planejador versão 1.0, 3.0 e o MongoDB

Pré-requisitos

Os seguintes pré-requisitos se aplicam ao planejador versão 3.0:

- A versão 3.0 do Planner está disponível em todas as áreas em Regiões da AWS que a versão 8.0 do motor está disponível.
- O Planner versão 3.0 é o planejador de consultas padrão quando a versão 8.0 do mecanismo é selecionada.

Selecionar o planejador versão 3.0 como o planejador de consultas padrão

Se você alterou seu planejador de consultas padrão no Amazon DocumentDB 8.0 e precisa reverter para o planejador v3, você pode fazer isso a partir do console ou da CLI:

- Siga as etapas em Modificando os parâmetros de cluster do Amazon DocumentDB para modificar o grupo de parâmetros do seu cluster.
- Para o parâmetro intitulado 'PlannerVersion', altere o valor para 3.0, indicando a versão 3.0 do planejador.
- Selecione Aplicar imediatamente (selecionar Aplicar na reinicialização tornará a seleção ineficaz até a próxima reinicialização do cluster).

Práticas recomendadas

Para obter os resultados esperados, use as seguintes práticas recomendadas ao aplicar o planejador versão 3.0:

- Em um cluster global, selecione o mesmo `plannerVersion` valor (1,0, 2,0 ou 3,0) nos grupos de parâmetros do cluster para ambas as regiões. Observe que selecionar diferentes versões do planejador nas regiões primária e secundária pode causar comportamento e desempenho inconsistentes da consulta.

- A atualização para a versão 3.0 do planejador durante uma janela de manutenção programada ou durante períodos de tráfego reduzido será a menos disruptiva, pois pode haver um aumento nas taxas de erro se a versão do planejador for alterada quando as cargas de trabalho estiverem em execução ativa.

Limitações

As seguintes limitações se aplicam ao planejador versão 3.0:

- A versão 3.0 do Planner não é suportada em clusters elásticos, que retornarão à versão 1.0 do planejador.

- Enquanto o Planner v1 permite o uso do 'PlanHint' para garantir que um plano de consulta específico seja selecionado pelo otimizador de consultas, o Planner v3 não permite o uso do 'PlanHint' e depende de otimizações internas para escolher o melhor plano para a consulta em questão.

Melhorias em **`operadores agregados`** e **`distintos`**

A versão 3.0 do Planner introduz melhorias nos estágios $aggregate e no comando $distinct. A seguir estão algumas das melhorias mais notáveis..

- O planejador move os estágios $match mais cedo no pipeline quando possível, reduzindo o número de documentos processados pelos estágios subsequentes.

```
//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 } }
])
```

- O planejador combina automaticamente os estágios $lookup e $unwind quando eles operam no mesmo campo, reduzindo o processamento intermediário de dados e melhorando o desempenho.

```
                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
```

- A versão 3.0 do Planner apresenta uma nova estratégia de execução do Distinct Scan que melhora significativamente o desempenho de operações distintas em índices de baixa cardinalidade.

```
                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"
                        }
                }
        }
}
```

Planejar filtros de cache

A versão 3.0 do Planner suporta filtros de cache de plano, também chamados de filtros de índice, que permitem restringir o conjunto de índices que o planejador considera para uma forma de consulta específica. Os filtros são definidos com um comando de banco de dados e aplicados no servidor. Portanto, se você tiver uma regressão de consulta, poderá atenuá-la sem modificar o código do aplicativo.

A versão 3.0 do Planner aplica filtros de cache do plano a mais comandos do que a versão 2.0 do planejador:

- O `distinct` comando requer a versão 3.0 do planejador.
- O `aggregate` comando exige a versão 3.0 do planejador e o Amazon DocumentDB 8.0.2. Um filtro se aplica quando o pipeline lê a coleção em um `$match` estágio inicial.
- Um filtro definido na coleção externa de um `$graphLookup` estágio `$lookup` ou conduz o índice usado para escanear essa coleção externa. Isso também requer a versão 3.0 do planejador e o Amazon DocumentDB 8.0.2.

Para obter mais informações sobre a referência do comando, formas de consulta suportadas, exemplos resolvidos e como os filtros interagem com as dicas, consultePlanejar filtros de cache.

Possíveis diferenças de comportamento entre o planejador versão 1.0, 3.0 e o MongoDB

Em alguns casos extremos, é possível que a versão 3.0 do planejador produza resultados ligeiramente diferentes da versão 1.0 do planejador. Esta seção mostra alguns exemplos dessas possibilidades.

Feature Differences

- Em uma coleção vazia ou quando estágios anteriores, como match, filtram todos os documentos, o Planner v1 fornece o “campo” de saída :0. O Planner v3 e o MongoDB não fornecerão nenhuma saída.

- Com o Plannerv1, {"$skip” :n} não pula se houver menos de n documentos retornados pelo estágio anterior. O Plannerv3 e o MongoDB ignoram corretamente, independentemente do número de documentos retornados.
- Quando uma coleção estrangeira referenciada em $lookup não existe, o plannerv1 gera um erro. O Planner v3 e o MongoDB tratam a coleção externa como uma coleção vazia e executam $lookup.

```
db.coll.aggregate([
    {$lookup: {from: "does_not_exist", localField: "a", foreignField: "a", as:
 "c"}}
])
```

- Somente o Planner v1 permitirá o uso de vários $search em um pipeline. O Planner v2/v3 gerará um erro e o MongoDB não o suportará.

```
VectorSearch = { "$search": { "vectorSearch": {
   "vector": [0.2, 0.5, 0.8],
   "path": "vectorEmbedding",
   "similarity": "cosine",
   "k": 2,
   "efSearch": 1
   }}}
db.coll.aggregate([VectorSearch, VectorSearch])
```

- Somente o Planner v3 funciona quando o estágio VectorSearch não é o primeiro estágio. Nesse caso, o MongoDB lançará um erro e o Planner v1 não suporta o estágio $vectorSearch.

```
VectorSearch = { {"$vectorSearch": {
    "queryVector": [0.2, 0.5, 0.8],
    "path": "vectorEmbedding",
    "similarity": "euclidean",
    "limit": 4,
    "numCandidates": 100} }

db.coll.aggregate([$match:{}, VectorSearch])
```

Consultar dados geoespaciais com o Amazon DocumentDB

Esta seção aborda como você pode consultar dados geoespaciais com o Amazon DocumentDB. Depois de ler esta seção, você poderá responder como armazenar, consultar e indexar dados geoespaciais no Amazon DocumentDB.

Tópicos

- Visão geral do
- Indexar e armazenar dados geoespaciais
- Consultar dados geoespaciais
- Limitações

Visão geral do

Casos de uso comuns do Geospatial envolvem análise de proximidade de seus dados. Por exemplo, “encontrar todos os aeroportos em um raio de 50 milhas de Seattle” ou “encontrar os restaurantes mais próximos de um determinado local”. O Amazon DocumentDB usa a especificação GeoJSON para representar dados geoespaciais. GeoJSON é uma especificação de código aberto para formas em um espaço JSON-formatting de coordenadas. As coordenadas GeoJSON capturam tanto a longitude quanto a latitude, representando posições em uma esfera semelhante à Terra.

Indexar e armazenar dados geoespaciais

O Amazon DocumentDB usa o tipo GeoJSON “Point” para armazenar dados geoespaciais. Cada documento GeoJSON (ou subdocumento) geralmente é composto por dois campos:

- tipo - a forma que está sendo representada, que informa ao Amazon DocumentDB como interpretar o campo “coordenadas”. No momento, o Amazon DocumentDB só oferece suporte a pontos
- coordenadas — um par de latitude e longitude representado como um objeto em uma matriz — [longitude, latitude]

O Amazon DocumentDB também usa índices 2dsphere para indexar dados geoespaciais. O Amazon DocumentDB oferece suporte a pontos de indexação. O Amazon DocumentDB suporta consultas de proximidade com indexação 2dsphere.

Vamos considerar um cenário em que você está criando uma aplicação para serviço de entrega de comida. Você deseja armazenar pares de latitude e longitude de restaurantes no Amazon DocumentDB. Para fazer isso, primeiro crie um índice no campo geoespacial que contenha o par de latitude e longitude.

```
use restaurantsdb
db.usarestaurants.createIndex({location:"2dsphere"})
```

A saída desse comando será semelhante a esta:

```
{
 "createdCollectionAutomatically" : true,
 "numIndexesBefore" : 1,
 "numIndexesAfter" : 2,
 "ok" : 1
}
```

Depois de criar um índice, você pode começar a inserir dados em sua coleção do 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
      ]
   }
});
```

Consultar dados geoespaciais

O Amazon DocumentDB suporta consultas de proximidade, inclusão e interseção de dados geoespaciais. Um bom exemplo de consulta de proximidade é encontrar todos os pontos (todos os aeroportos) que estão a menos de uma certa distância e a mais do que uma distância de outro ponto (cidade). Um bom exemplo de consulta de inclusão é encontrar todos os pontos (todos os aeroportos) localizados em um determinado area/polygon (estado de Nova York). Um bom exemplo de consulta de interseção é encontrar um polígono (estado) que cruza com um ponto (cidade). Você pode usar os seguintes operadores geoespaciais para obter informações sobre seus dados.

- **`$nearSphere`**- `$nearSphere` é um operador de busca que suporta encontrar pontos do mais próximo ao mais distante de um ponto GeoJSON.
- **`$geoNear`**- `$geoNear` é um operador de agregação que suporta o cálculo da distância em metros a partir de um ponto GeoJSON.
- **`$minDistance`**- `$minDistance` é um operador de busca usado em conjunto com `$nearSphere` ou `$geoNear` para filtrar documentos que estejam pelo menos na distância mínima especificada do ponto central.
- **`$maxDistance`**- `$maxDistance` é um operador de busca usado em conjunto com `$nearSphere` ou `$geoNear` para filtrar documentos que estejam no máximo na distância máxima especificada do ponto central.

- **`$geoWithin`**- `$geoWithin` é um operador de busca que suporta a localização de documentos com dados geoespaciais que existem inteiramente dentro de uma forma especificada, como um polígono.
- **`$geoIntersects`**- `$geoIntersects` é um operador de busca que suporta a localização de documentos cujos dados geoespaciais se cruzam com um objeto GeoJSON especificado.

> **Note**
>
> `$geoNeare $nearSphere` exija um índice 2dsphere no campo GeoJSON que você usa em sua consulta de proximidade.

Exemplo 1

Neste exemplo, você aprenderá como encontrar todos os restaurantes (pontos) classificados pela distância mais próxima de um endereço (ponto).

Para realizar essa consulta, você pode usar `$geoNear` para calcular a distância do conjunto de pontos de outro ponto. Você também pode adicionar o `distanceMultiplier` para medir a distância em quilômetros.

```
db.usarestaurants.aggregate([
   {
      "$geoNear":{
         "near":{
            "type":"Point",
            "coordinates":[
               -122.3516,
               47.6156
            ]
         },
         "spherical":true,
         "distanceField":"DistanceKilometers",
         "distanceMultiplier":0.001
      }
   }
])
```

O comando acima retornaria os restaurantes ordenados pela distância (mais próxima para a mais distante) do ponto especificado. A saída desse comando será semelhante a esta

```
{ "_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**
>
> numAs opções `limit` e para o `$geoNear` estágio, bem como o limite padrão de 100 documentos, estão obsoletas no Amazon DocumentDB 8.0 (patch de motor versão 4.0.5748) e versões posteriores. Os exemplos a seguir usam `limit` e num se aplicam somente ao Amazon DocumentDB 8.0 (patch de motor versão 4.0.4448) e versões anteriores.

Para limitar o número de resultados em uma consulta, use a opção `limit` ou 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
      }
   }
])
```

Exemplo 2

Neste exemplo, você aprenderá como encontrar todos os restaurantes (pontos) dentro de 2 quilômetros de um endereço específico (ponto). Para realizar tal consulta, você pode usar `$nearSphere` dentro de um mínimo `$minDistance` e máximo `$maxDistance` de um ponto GeoJSON

```
db.usarestaurants.find({
   "location":{
      "$nearSphere":{
         "$geometry":{
            "type":"Point",
            "coordinates":[
               -122.3516,
               47.6156
            ]
         },
         "$minDistance":1,
         "$maxDistance":2000
      }
   }
},
{
   "name":1
```

```
})
```

O comando acima retornaria restaurantes a uma distância máxima de 2 quilômetros do ponto especificado. A saída desse comando será semelhante a esta

```
{ "_id" : ObjectId("611f3da985009a81ad38e74b"), "name" : "Noodle House" }
```

Limitações

O Amazon DocumentDB não oferece suporte à consulta ou indexação de polígonos,, LineString, e. MultiPoint MultiPolygon MultiLineString GeometryCollection

Índice parcial

Um índice parcial indexa documentos em uma coleção que atende a um critério de filtro especificado. O recurso parcial de índice é aceito em clusters baseados em instância do Amazon DocumentDB 5.0.

Tópicos

- Crie um índice parcial
- Operadores compatíveis
- Consultar usando um índice parcial
- Funcionalidades parciais de índice
- Limitações parciais do índice

Crie um índice parcial

Para criar um índice parcial, use o método `createIndex()` com a opção `partialFilterExpression`. Por exemplo, a operação a seguir cria um índice composto exclusivo na coleção de pedidos que indexa documentos com um `OrderID` e com o campo `isDelivered` verdadeiro:

```
db.orders.createIndex(
  {"category": 1, "CustomerId": 1, "OrderId": 1},
  {"unique": true, "partialFilterExpression":
    {"$and": [
      {"OrderId": {"$exists": true}},
```

```
      {"isDelivered": {"$eq": false}}
    ]}
  }
)
```

Operadores compatíveis

- $eq
- $exists
- $and (somente no nível superior)
- $gt/$gte/$lt/$lte (a varredura de índice só é usada quando o filtro, predicado na consulta, corresponde exatamente à expressão do filtro parcial) (consulte Limitações)

Consultar usando um índice parcial

Os seguintes padrões de consulta são possíveis usando índices parciais:

- O predicado da consulta corresponde exatamente à expressão do filtro de índice parcial:

```
db.orders.find({"$and": [
    {"OrderId": {"$exists": true}},
    {"isDelivered": {"$eq": false}}
  ]}).explain()
```

- O resultado esperado do filtro de consulta é um subconjunto lógico do filtro parcial:

```
db.orders.find({"$and": [
    {"OrderId": {"$exists": true}},
    {"isDelivered": {"$eq": false}},
    {"OrderAmount": {"$eq": "5"}}
  ]}).explain()
```

- Um subpredicado da consulta pode ser usado em conjunto com outros índices:

```
db.orders.createIndex({"anotherIndex":1})
db.orders.find({ "$or": [
      {"$and": [
        {"OrderId": {"$exists": true}},
        {"isDelivered": {"$eq": false}}
```

```
      ]},
      {"anotherIndex": {"$eq": 5}}
    ]
  }).explain()
```

> **Note**
>
> Um planejador de consultas pode optar por usar uma varredura de coleção em vez de uma varredura de índice, se for eficiente fazer isso. Isso geralmente é visto em coleções ou consultas muito pequenas que retornariam uma grande parte de uma coleção.

Funcionalidades parciais de índice

Listar índices parciais

Liste índices parciais com parciais FilterExpression usando a `getIndex` operação. Por exemplo, a operação `getIndex` emitida lista índices parciais com os campos key, name e partialFilterExpressions:

```
db.orders.getIndexes()
```

Este exemplo retorna a saída a seguir:

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

Expressão múltipla de filtro parcial na mesma chave:ordem

Diferentes índices parciais podem ser criados para as mesmas combinações de campos (chave:ordem). Esses índices devem ter nomes diferentes.

```
db.orders.createIndex(
  {"OrderId":1},
  {
    name:"firstPartialIndex",
    partialFilterExpression:{"OrderId":{"$exists": true}}
  }
)
```

```
db.orders.createIndex(
  {"OrderId":1},
  {
    name:"secondPartialIndex",
    partialFilterExpression:{"OrderId":{"$gt": 1000}}
  }
)
```

Execute a operação `getIndexes` para listar todos os índices na coleção:

```
db.orders.getIndexes()
```

Este exemplo retorna a saída a seguir:

```
[
```

```
  {
    "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**
>
> Os nomes dos índices devem ser diferentes e devem ser excluídos somente pelo nome.

Índices com propriedades parciais e TTL

Você também pode criar índices com propriedades parciais e TTL especificando as opções `partialFilterExpression` e `expireAfterSeconds` durante a criação do índice. Isso permite que você forneça mais controle sobre quais documentos agora são removidos de uma coleção.

Por exemplo, você pode ter um índice TTL que identifica documentos a serem excluídos após um determinado período de tempo. Agora você pode fornecer condições adicionais sobre quando excluir documentos usando a opção de índice parcial:

```
db.orders.createIndex(
    { "OrderTimestamp": 1 },
    {
        expireAfterSeconds: 3600 ,
        partialFilterExpression: { "isDelivered": { $eq: true } }
    }
)
```

Este exemplo retorna a saída a seguir:

```
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1,
        "operationTime" : Timestamp(1234567890, 1)
}
```

Execute a operação `getIndexes` para listar os índices presentes na coleção:

```
db.orders.getIndexes()
[
    {
        "v" : 4,
        "key" : {
            "_id" : 1
        },
        "name" : "_id_",
        "ns" : "test.orders"
    }
```

Este exemplo retorna a saída a seguir:

```
[
    {
        "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
    }
]
```

Limitações parciais do índice

As seguintes limitações se aplicam ao recurso de índice parcial:

- As consultas de desigualdade no Amazon DocumentDB só usarão um índice parcial quando o predicado do filtro de consulta corresponder exatamente a `partialFilterExpression` e for do mesmo tipo de dados.

  > **Note**
  >
  > Nem mesmo `$hint` pode ser usado para forçar o IXSCAN no caso acima.

  No exemplo a seguir, `partialFilterExpression` é aplicado somente a `field1`, mas não 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
```

- Um `partialFilterExpression` com operadores de matriz não é compatível. A operação a seguir gerará um erro:

```
db.orders.createIndex(
  {"CustomerId":1},
  {'partialFilterExpression': {'OrderId': {'$eq': [1000, 1001, 1002]}}}
)
```

- Os seguintes operadores não são suportados em FilterExpression campo parcial:
  - `$all` (operador de matriz)
  - `$mod` (operador de matriz)
  - `$or`
  - `$xor`
  - `$not`
  - `$nor`
- O tipo de dados da expressão do filtro e do filtro deve ser o mesmo.

Realizar pesquisa de texto com o Amazon DocumentDB

O recurso nativo de pesquisa de texto completo do Amazon DocumentDB (Text Index v1) permite que você realize pesquisas de texto em grandes conjuntos de dados textuais usando índices de texto para fins especiais. Esta seção descreve as funcionalidades do recurso de índice de texto e fornece etapas sobre como criar e usar índices de texto no Amazon DocumentDB. As limitações da pesquisa de texto também estão listadas.

Tópicos

- Funções com suporte
- Usar o índice de texto do Amazon DocumentDB
- Diferenças com o MongoDB
- Práticas recomendadas e diretrizes
- Índice de texto V2
- Limitações

Funções com suporte

A pesquisa de texto do Amazon DocumentDB suporta as seguintes funcionalidades compatíveis com a API MongoDB:

- Criar índices de texto em um único campo.
- Criar índices de texto composto que incluam mais de um campo de texto.
- Fazer pesquisas com uma ou várias palavras.
- Controlar os resultados da pesquisa usando pesos.
- Classificar os resultados da pesquisa por pontuação.
- Usar o índice de texto no pipeline de agregação.
- Pesquisar a frase exata.

Usar o índice de texto do Amazon DocumentDB

Para criar um índice de texto em um campo contendo dados de string, especifique a string “text” conforme mostrado abaixo:

Índice de campo único:

```
db.test.createIndex({"comments": "text"})
```

Esse índice suporta consultas de pesquisa de texto no campo da string “comments” na coleção especificada.

Crie um índice de texto composto em mais de um campo de string:

```
db.test.createIndex({"comments": "text", "title":"text"})
```

Esse índice suporta consultas de pesquisa de texto nos campos de caracteres “comentários” e “título” na coleção especificada. É possível especificar até 30 campos ao criar um índice de texto composto. Depois de criadas, suas consultas de pesquisa de texto consultarão todos os campos indexados.

> **Note**
>
> Somente um índice de texto é permitido em cada coleção.

Listar um índice de texto em uma coleção do Amazon DocumentDB

Você pode usar `getIndexes()` em sua coleção para identificar e descrever índices, incluindo índices de texto, conforme mostrado no exemplo a seguir:

```
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
   }
]
```

Depois de criar um índice, você pode começar a inserir dados em sua coleção do 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"}])
```

Executar consultas de pesquisa de texto

Executar uma consulta de pesquisa de texto com uma única palavra

Você precisará usar os operadores `$text` e `$search` para realizar pesquisas de texto. O exemplo a seguir retorna todos os documentos em que seu campo indexado de texto contém a string “apple” simplesmente ou em outros formatos, como “apples”:

```
db.test.find({$text: {$search: "apple"}})
```

Saída:

A saída desse comando será semelhante a esta:

```
{ "_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" }
```

Executar uma pesquisa de texto com várias palavras

Você também pode realizar pesquisas de texto com várias palavras nos seus dados do Amazon DocumentDB. O comando abaixo retorna documentos com um campo de texto indexado contendo “apple” ou “pie”:

```
db.test.find({$text: {$search: "apple pie"}})
```

Saída:

A saída desse comando será semelhante a esta:

```
{ "_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" }
```

Executar uma pesquisa de texto com várias palavras

Para uma pesquisa com várias palavras, use este exemplo:

```
db.test.find({$text: {$search: "\"apple pie\""}})
```

Saída:

O comando acima retorna documentos com campo de texto indexado contendo a frase exata “apple pie”. A saída desse comando será semelhante a esta:

```
{ "_id" : 4, "star_rating" : 2, "comments" : "bake the apple pie in the oven" }
```

Executar uma pesquisa de texto com filtros

Você também pode combinar a pesquisa de texto com outros operadores de consulta para filtrar os resultados com base em critérios adicionais:

```
db.test.find({$and: [{star_rating: 5}, {$text: {$search: "interest"}}]})
```

Saída:

O comando acima retorna documentos com um campo indexado de texto contendo qualquer forma de “interest” e uma “star_rating” igual a 5. A saída desse comando será semelhante a esta:

```
{ "_id" : 5, "star_rating" : 5, "comments" : "interesting couch" }
{ "_id" : 6, "star_rating" : 5, "comments" : "interested in couch for sale, year
 2022" }
```

Limitar o número de documentos retornados em uma pesquisa de texto

É possível optar por restringir o número de documentos devolvidos usando `limit`:

```
db.test.find({$and: [{star_rating: 5}, {$text: {$search: "couch"}}]}).limit(1)
```

Saída:

O comando acima retorna um resultado que satisfaz o filtro:

```
{ "_id" : 5, "star_rating" : 5, "comments" : "interesting couch" }
```

Classificar resultados por pontuação de texto

O exemplo a seguir classifica os resultados da pesquisa de texto por pontuação de texto:

```
db.test.find({$text: {$search: "apple"}}, {score: {$meta: "textScore"}}).sort({score:
 {$meta: "textScore"}})
```

Saída:

O comando acima retorna documentos com um campo de texto indexado contendo “apple” simplesmente ou em outros formatos, como “apples”, e classifica o resultado com base na relevância do documento em relação ao termo de pesquisa. A saída desse comando será semelhante a esta:

```
{ "_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 }
```

`$text` e `$search` também têm suporte para os comandos `aggregate`, `count`, `findAndModify`, `update` e `delete`.

Operadores de agregação

Pipeline de agregação usando **`$match`**

```
db.test.aggregate(
   [{ $match: { $text: { $search: "apple pie" } } }]
)
```

Saída:

O comando acima retorna os seguintes 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" }
```

Uma combinação de outros operadores de agregação

```
db.test.aggregate(
   [
      { $match: { $text: { $search: "apple pie" } } },
```

```
      { $sort: { score: { $meta: "textScore" } } },
      { $project: { score: { $meta: "textScore" } } }
   ]
)
```

Saída:

O comando acima retorna os seguintes resultados:

```
{ "_id" : 4, "score" : 0.6079270860936958 }
{ "_id" : 1, "score" : 0.3039635430468479 }
{ "_id" : 2, "score" : 0.3039635430468479 }
{ "_id" : 3, "score" : 0.3039635430468479 }
```

Especifique vários campos ao criar um índice de texto

É possível atribuir pesos a até três campos em seu índice de texto composto. O peso padrão atribuído a um campo em um índice de texto é um (1). O peso é um parâmetro opcional e deve estar na faixa de 1 a 100000.

```
db.test.createIndex(
   {
     "firstname": "text",
     "lastname": "text",
     ...
   },
   {
     weights: {
       "firstname": 5,
       "lastname":10,
       ...
     },
     name: "name_text_index"
   }
 )
```

Diferenças com o MongoDB

O atributo de índice de texto do Amazon DocumentDB usa índice invertido com um algoritmo de frequência de termos. Os índices de texto são esparsos por padrão. Devido às diferenças na lógica de análise, delimitadores de tokenização e outros, o mesmo conjunto de resultados do MongoDB pode não ser retornado para o mesmo conjunto de dados ou formato de consulta.

Existem as seguintes diferenças adicionais entre o índice de texto do Amazon DocumentDB e o MongoDB:

- Não há suporte para índices compostos usando índices que não sejam de texto.
- Os índices de texto do Amazon DocumentDB não diferenciam maiúsculas de minúsculas.
- Somente o idioma inglês tem suporte com o índice de texto.
- A indexação de texto de campos de matriz (ou de várias chaves) não tem suporte. Por exemplo, a criação de um índice de texto em “a“ com o documento {“a”: [“apple”, “pie”]} falhará.
- A indexação de texto curinga não tem suporte.
- Índices de texto exclusivos não têm suporte.
- A exclusão de um termo não tem suporte.

Práticas recomendadas e diretrizes

- Para obter o desempenho ideal em consultas de pesquisa de texto envolvendo classificação por pontuações de texto, crie o índice de texto antes de carregar os dados.
- Os índices de texto exigem armazenamento adicional para uma cópia interna otimizada dos dados indexados. Isso tem implicações adicionais de custo.

Índice de texto V2

O Amazon DocumentDB 8.0 apresenta uma nova versão do índice de texto (V2) que altera o analisador de pesquisa de texto subjacente para trazer mais compatibilidade com o MongoDB.

Além das funcionalidades fornecidas pelos índices de texto V1, os índices de texto V2 também fornecem o seguinte suporte:

- O planejador move os estágios $match mais cedo no pipeline quando possível, reduzindo o número de documentos processados pelos estágios subsequentes.
- Tokenização aprimorada de caracteres especiais em campos de texto, como e-mails, URLs e caminhos de arquivo. A V2 pode analisar e combinar tokens individuais dentro dessas strings, enquanto a V1 não pode.

```
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" }
```

Limitações

A pesquisa de texto tem as seguintes limitações no Amazon DocumentDB:

- Os índices de texto armazenam lexemas e suas informações de posição. O tamanho combinado de todos os lexemas e suas informações de posição, em um único documento, é limitado a 1 MB.

Índices

Os índices melhoram o desempenho das consultas ao permitir que o Amazon DocumentDB localize documentos rapidamente sem digitalizar todos os documentos em uma coleção. Você pode criar índices nos campos que seu aplicativo consulta com frequência para reduzir a latência e os custos. I/O

Para ver outros tópicos relacionados ao índice, consulte:

- Gerenciar índices do Amazon DocumentDB— criação e construção de índices
- Trabalhar com índices— melhores práticas de indexação
- Solução de problemas de índices— resolvendo falhas de compilação de índices e inchaço
- Como analiso o uso do índice e identifico índices não utilizados?— encontrar índices não utilizados e ausentes

Tópicos

- Gerenciamento de índices
- Tipos de índice
- Propriedades de índice

Gerenciamento de índices

Esta seção aborda tópicos de gerenciamento de índices para o Amazon DocumentDB.

Tópicos

- Compilação de índice
- Manutenção de índices
- Práticas recomendadas para trabalhar com índices

Compilação de índice

A criação de índices no Amazon DocumentDB exige que várias decisões sejam tomadas:

- Com que rapidez ele precisa ser concluído?
- A coleção pode ficar inacessível enquanto a construção está ocorrendo?
- Quanto da capacidade computacional de uma instância pode ser alocada para a compilação?
- Que tipo de índice deve ser criado?

Esta seção ajuda você a responder a essas perguntas e fornece os comandos e exemplos de monitoramento para criar um índice Amazon DocumentDB em sua coleção de clusters baseada em instâncias.

Diretrizes

As diretrizes a seguir incluem limites básicos e compensações de configuração ao criar novos índices:

- Suporte à versão do Amazon DocumentDB: embora a indexação de um único trabalhador seja suportada em todas as versões do Amazon DocumentDB, a indexação de vários trabalhadores é suportada somente nas versões 4.0 e 5.0 do Amazon DocumentDB.

- Compensação de desempenho - Aumentar o número de trabalhadores no processo de criação do índice aumenta a utilização da CPU e a IO de leitura na instância primária do seu banco de dados Amazon DocumentDB. Os recursos necessários para criar um novo índice não estarão disponíveis para sua workload em execução.
- Clusters elásticos: a indexação paralela não tem suporte nos clusters elásticos do Amazon DocumentDB.
- Clusters sem servidor - A indexação paralela em clusters sem servidor exige uma capacidade mínima de pelo menos 32 DCUs. Abaixo desse limite, o `workers` parâmetro é aceito, mas a construção do índice é executada com um único trabalhador. O número máximo de trabalhadores paralelos é dimensionado de acordo com sua capacidade mínima configurada: aproximadamente `min_DCU / 8` trabalhadores estão disponíveis. Por exemplo, em 32 DCUs você pode usar até 4 trabalhadores e em 64 DCUs você pode usar até 8 trabalhadores.

  > **Note**
  >
  > É necessário reiniciar o cluster após alterar a configuração de capacidade mínima para que o novo limite de trabalhadores paralelos entre em vigor.

- Máximo de trabalhadores (com base na instância) - O número máximo de trabalhadores que você pode configurar depende do tamanho da sua instância primária no cluster de banco de dados. É metade do número total de vCPUs na instância primária do seu cluster de banco de dados. Por exemplo, você pode executar no máximo 32 trabalhadores em uma instância db.r6g.16xlarge que tenha 64 vCPUs.

> **Note**
>
> A criação de índices paralelos é compatível com instâncias 2xlarge e superiores.

- Mínimo de trabalhadores - O número mínimo de trabalhadores que você pode configurar é um. A configuração padrão para criação de índices em clusters baseados em instâncias é de dois trabalhadores. No entanto, você pode reduzir o número de trabalhadores para um usando a opção “segmentos de trabalho”. Isso executará o processo com um único trabalhador.
- Compactação de índice - O Amazon DocumentDB não oferece suporte à compactação de índices. O tamanho dos dados para dados e índices armazenados poderá ser maior do que ao usar outras opções.

- Indexação de várias coleções - Metade das vCPUs na instância primária do seu cluster de banco de dados pode ser usada para trabalhadores configurados que realizam a criação de índices em várias coleções.
- Tipos de índice — Consulte Tipos de índice a lista de tipos de índices compatíveis no Amazon DocumentDB.

Tipos de criação de índice

Os quatro tipos de criação de índices são:

- Primeiro plano: a construção do índice em primeiro plano bloqueia todas as outras operações do banco de dados na coleção em que o índice é criado, até que a criação do índice seja concluída. A construção em primeiro plano do Amazon DocumentDB é composta por cinco estágios.
- Primeiro plano (exclusivo): compilações de índice de primeiro plano de documento único (exclusivo) bloqueiam outras operações de banco de dados, como compilações regulares de primeiro plano. Ao contrário da construção básica em primeiro plano, a compilação exclusiva usa um estágio adicional (chaves de classificação 2) para procurar chaves duplicadas. A construção em primeiro plano (exclusiva) é composta por seis estágios.
- Plano de fundo: a construção do índice em segundo plano permite que outras operações do banco de dados sejam executadas em primeiro plano enquanto o índice está sendo criado. A construção em segundo plano do Amazon DocumentDB é composta por oito estágios.
- Plano de fundo (exclusivo): as compilações de índice em segundo plano de documento único (exclusivo) permitem que outras operações de banco de dados sejam executadas em primeiro plano enquanto o índice está sendo criado. Ao contrário da construção básica em segundo plano, a compilação exclusiva usa um estágio adicional (chaves de classificação 2) para procurar chaves duplicadas. A construção de fundo (única) é composta por nove estágios.

O tipo de construção de índice padrão é primeiro plano.

Para criar um índice em segundo plano, você adiciona o parâmetro “background”: true ao criar o índice.

Introdução

Para iniciar a criação do índice em uma coleção, use o comando CreateIndexes. Por padrão, o comando executará dois trabalhadores paralelos, o que aumenta a velocidade do processo de criação do índice em duas vezes.

Por exemplo, o processo de comando a seguir demonstra como criar um índice para o campo “user_name” em um documento e aumentar a velocidade do processo de indexação para quatro trabalhadores:

1. Crie índices em segundo plano usando dois trabalhadores paralelos no cluster:

```
db.runCommand({"createIndexes":"test","indexes":[{"key": {"user_name":1},
 "name":"username_idx", "background":true }]})
```

2. Para otimizar a velocidade do processo de criação do índice, você pode especificar o número de trabalhadores usando a opção “threads de trabalho” (“trabalhadores”:<number>) no comando db.runCommand createIndexes.

   Aumente a velocidade do processo para quatro trabalhadores paralelos:

```
db.runCommand({"createIndexes":"test","indexes":[{"key": {"user_name":1},
 "name":"username_idx", "workers":4}]})
```

> **Note**
>
> Quanto maior o número de trabalhadores, mais rápido a criação do índice progride. No entanto, quanto maior o número de trabalhadores, maior o aumento da carga nas vCPUs e na E/S de leitura da sua instância primária. Garanta que seu cluster esteja suficientemente provisionado para lidar com o aumento da carga sem degradar outras workloads.

Status do andamento da indexação

O processo de criação de índices funciona inicializando, escaneando coleções, classificando chaves e, finalmente, inserindo chaves por meio de um construtor de índices. O processo tem até seis estágios quando você o executa em primeiro plano e até nove estágios quando você o executa em segundo plano. É possível visualizar métricas de status, como percentual de conclusão, número total de blocos de armazenamento digitalizados, chaves classificadas e chaves inseridas, etapa por etapa.

Monitore o progresso do processo de indexação usando o comando db.currentOp () no shell mongo. Uma conclusão de 100% da última etapa mostra que todos os índices foram criados com sucesso:

```
db.currentOp({"command.createIndexes": { $exists : true } })
```

> **Note**
>
> Só há suporte para a visualização do status do andamento da indexação no Amazon DocumentDB 5.0.

Etapas de construção do índice

Estágio	Primeiro plano	Primeiro plano (exclusivo)	Contexto	Plano de fundo (exclusivo)
Inicializando	1	1	1	1
índice de construção: inicializando	2	2	2	2
índice de construção: varredura da coleção	3	3	3	3
índice de construção: chaves de classificação 1	4	4	4	4
índice de construção: chaves de classificação 2		5		5

Estágio	Primeiro plano	Primeiro plano (exclusivo)	Contexto	Plano de fundo (exclusivo)
índice de construção: inserindo chaves	5	6	5	6
validação: índice de digitalização			6	7
validação: classificação de tuplas			7	8
validação: coleta de digitalização			8	9

- inicialização - createIndex está preparando o construtor de índices. Essa fase deve ser muito breve.
- índice de construção: inicialização - O criador de índices está se preparando para criar o índice. Essa fase deve ser muito breve.
- índice de construção: coleção de varredura - O construtor de índices está realizando uma varredura de coleção para coletar chaves de índice. A unidade de medida é “blocos”.

> **Note**
>
> Se mais de um trabalhador estiver configurado para a criação do índice, ele será exibido nesse estágio. O estágio de “coleta de digitalização” é o único estágio que usa vários trabalhadores durante o processo de criação do índice. Todos os outros estágios exibirão um único trabalhador.

- índice de construção: chaves de classificação 1 - O construtor de índices está classificando as chaves de índice coletadas. A unidade de medida é “chaves”.

- índice de construção: chaves de classificação 2 - O construtor de índices está classificando as chaves de índice coletadas que correspondem às tuplas mortas. Essa fase existe apenas para a criação de índices exclusivos. A unidade de medida é “chaves”.
- índice de construção: inserindo chaves - O criador de índices está inserindo chaves de índice no novo índice. A unidade de medida é “chaves”.
- validando: índice de varredura - createIndex está examinando o índice para encontrar chaves que precisam ser validadas. A unidade de medida é “blocos”.
- validando: classificando tuplas - createIndex está classificando a saída da fase de varredura do índice.
- validando: escaneando a coleção - CreateIndex está escaneando a coleção para validar as chaves de índice encontradas nas duas fases anteriores. A unidade de medida é “blocos”.

Exemplo de saída de compilação de índice

No exemplo de saída a seguir (construção do índice em primeiro plano), o status da criação do índice é mostrado. O campo “msg” resume o progresso da construção indicando o estágio e a porcentagem de conclusão da construção. O campo “trabalhadores” indica o número de trabalhadores usados durante esse estágio da criação do índice. O campo “progresso” mostra os números reais usados para calcular a porcentagem de conclusão.

> **Note**
>
> Os campos IndexBuildName “current”, “msg” e “progress” não são suportados no Amazon DocumentDB versão 4.0.

```
{
    "inprog" : [{
    …
        "command": {
            "createIndexes": "test",
            "indexes": [{
                "v": 2,
                "key": {
                    "user_name": 1
                },
```

```
                "name": "user_name_1"
            }],
            "lsid": {
                "id": UUID("094d0fba-8f41-4373-82c3-7c4c7b5ff13b")
            },
            "$db": "test"
        },
        "currentIndexBuildName": user_name_1,
        "msg": "Index Build: building index number_1, stage 6/6 building index:
 656860/1003520 (keys) 65%",
        "workers": 1,
        "progress": {
            "done": 656861,
            "total": 1003520
        },
    …
    ],

    "ok" : 1
}
```

Recursos de solução de problemas

Veja os seguintes recursos para solucionar problemas de criação de índices:

- Falha na construção do índice
- Problemas e falhas de latência de compilação do índice em segundo plano

Manutenção de índices

Sobrecarga de índices

O Amazon DocumentDB usa o Controle de Multi-Version Concorrência (MVCC) para gerenciar transações simultâneas. Quando os documentos são excluídos ou atualizados, suas versões anteriores permanecem em coleções e índices como versões “inativas”. O processo de coleta de resíduos recupera automaticamente o espaço dessas versões inativas para futuras operações.

A sobrecarga de índices ocorre quando os índices de uma coleção se tornam maiores devido ao acúmulo de entradas de índice inativas ou obsoletas ou à fragmentação nas páginas. O percentual relatado representa a quantidade de espaço de índices que pode ser usada por futuras entradas de

índice. Essa sobrecarga consome espaço no cache e no armazenamento do buffer. Se você quiser remover a sobrecarga, precisará reconstruir os índices.

Exemplo: execute o comando a seguir para determinar o armazenamento não utilizado para seu índice:

```
db.coll.aggregate({$indexStats:{}});
```

O resultado deve ser semelhante a este:

```
{
    "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
    }
}
```

Você pode reconstruir índices sem tempo de inatividade usando o comando reIndex, que exige uma varredura de toda a coleção. Consulte Manutenção do índice usando reIndex.

Manutenção do índice usando reIndex

ReIndex é um comando usado para reconstruir um índice. Geralmente é usado quando um índice se torna corrompido ou ineficiente. Com o tempo, os índices podem acumular espaço não utilizado devido a muitas atualizações, inserções ou exclusões, levando à degradação do desempenho. A reindexação ajuda a remover esse espaço não utilizado e a restaurar a eficiência do índice.

Diretrizes do ReIndex

- O reIndex só é compatível com o Amazon DocumentDB 5.0 e superior
- O reIndex sempre é executado em segundo plano.
- O Amazon DocumentDB suporta a reindexação de um único índice em segundo plano, permitindo a utilização de vários trabalhadores. O índice antigo pode ser usado por consultas quando o processo de reindexação está em execução.
- O Amazon DocumentDB oferece suporte à indexação de relatórios de progresso por meio do CurrentOp. Você pode ver os estágios de construção do índice semelhantes aos estágios de construção do índice visualizados durante a criação do índice. A única diferença é que o reIndex sempre tem oito estágios, independentemente de ser único ou não. Não há um estágio de “índice de construção: chaves de classificação 2".
- O reIndex pode ser executado simultaneamente com qualquer comando, exceto comandos relacionados ao índice na mesma coleção: createIndexes, dropIndexes, collMod e RenameCollection.
- Atualmente, o reIndex não é compatível com índices de texto, geoespaciais e vetoriais.
- O suporte de reIndex para índices parciais está disponível na versão 8.0.2 e posterior do mecanismo.

Construção de reindexação

Use o seguinte comando para recriar seu índice:

```
db.runCommand({ reIndex: "collection-name", index: "index-name"})
```

Opcionalmente, você também pode controlar o número de operadores atribuídos ao processo de reconstrução:

```
db.runCommand({ reIndex: "collection-name", index: "index-name", workers: number })
```

Para obter informações específicas sobre o gerenciamento de índices com Java, consulteGerenciamento de índices no Amazon DocumentDB com Java.

Práticas recomendadas para trabalhar com índices

Criação de índices

Ao importar dados para o Amazon DocumentDB, você deve criar seus índices antes de importar grandes conjuntos de dados. Você pode usar a ferramenta de índice Amazon DocumentDB para extrair índices de uma instância ou diretório mongodump do MongoDB em execução e criar esses índices em um cluster do Amazon DocumentDB. Para obter mais orientações sobre migrações, consulte Migração e atualização.

Seletividade do índice

Limite a criação de índices aos campos em que o número de valores duplicados seja inferior a 1% do número total de documentos na coleção. Por exemplo, se sua coleção contiver 100.000 documentos, crie índices somente em campos em que o mesmo valor ocorra 1.000 vezes ou menos.

A escolha de um índice com um alto número de valores exclusivos (ou seja, uma alta cardinalidade) garante que as operações de filtro retornem um pequeno número de documentos, gerando, assim, um bom desempenho durante as varreduras de índice. Um exemplo de índice de alta cardinalidade é um índice exclusivo, que garante que predicados de igualdade retornem, no máximo, um documento. Exemplos de baixa cardinalidade incluem um índice sobre um campo booliano e um índice sobre o dia da semana. Devido ao baixo desempenho, é improvável que índices de baixa cardinalidade sejam escolhidos pelo otimizador de consultas do banco de dados. Ao mesmo tempo, índices de baixa cardinalidade continuam consumindo recursos como espaço em disco e. I/Os Como regra geral, você deve direcionar índices a campos em que a frequência típica do valor é de 1%, ou menos, do tamanho total da coleção.

Além disso, é recomendável criar apenas índices em campos comumente usados como filtro e procurar regularmente índices não usados. Para obter mais informações, consulte the section called “Analise o uso do índice e identifique índices não utilizados”.

Impacto dos índices na gravação de dados

Embora os índices possam melhorar o desempenho da consulta evitando a necessidade de digitalizar todos os documentos em uma coleção, essa melhoria tem com uma compensação. Para

cada índice em uma coleção, sempre que um documento é inserido, atualizado ou excluído, o banco de dados deve atualizar a coleção e gravar os campos em cada um dos índices da coleção. Por exemplo, se uma coleção tiver nove índices, o banco de dados deverá executar dez gravações antes de confirmar a operação para o cliente. Assim, cada índice adicional gera latência de gravação adicional e aumento no armazenamento geral utilizado. I/Os

As instâncias de cluster precisam ser dimensionadas adequadamente para manter toda a memória do conjunto de trabalho. Isso evita a necessidade de ler continuamente as páginas de índice do volume de armazenamento, o que afeta negativamente o desempenho e gera I/O custos mais altos. Para obter mais informações, consulte Dimensionamento de instância.

Para obter um melhor desempenho, minimize o número de índices em suas coleções, adicionando apenas os índices necessários para melhorar o desempenho de consultas comuns. Embora as workloads variem, uma boa orientação é manter cada coleção com cinco índices ou menos.

Identificar índices ausentes

Como melhor prática, identifique os índices ausentes regularmente. Para obter mais informações, consulte the section called “Identifique índices ausentes”.

Identificar índices não utilizados

Como melhor prática, identifique e remova índices não utilizados regularmente. Para obter mais informações, consulte the section called “Analise o uso do índice e identifique índices não utilizados”.

Tipos de índice

Tópicos

- Índices de campo único
- Índices compostos
- Multi-key índices
- Índice de texto
- Indexação geoespacial
- Índices vetoriais

Índices de campo único

Índices de campo único armazenam informações de qualquer campo em uma coleção de documentos. Por padrão, todas as coleções têm um índice no `_id` campo. Você pode adicionar índices adicionais para acelerar consultas e operações importantes.

Índices de campo único são benéficos quando:

- Seu aplicativo frequentemente consulta ou filtra documentos com base nos valores de um campo específico.
- Você precisa classificar documentos de forma eficiente por um campo específico.
- Você quer garantir a exclusividade de um campo específico em todos os documentos de uma coleção criando um índice exclusivo.

Propriedades de índice suportadas

Opção	3.6	4,0	5,0	8.0	Cluster elástico
name	Sim	Sim	Sim	Sim	Sim
único	Sim	Sim	Sim	Sim	Sim
esparso *	Sim	Sim	Sim	Sim	Sim
parcialFilterExpression *	Não	Não	Sim	Sim	Não
expirarAfterSeconds	Sim	Sim	Sim	Sim	Sim

* As `partialFilterExpression` opções `sparse` e não podem ser usadas juntas na mesma definição de índice. Se você tentar criar um índice com essas opções, ele falhará com o seguinte erro:

```
Error in specification: cannot mix partialFilterExpression and sparse options
```

Criação de um índice de campo único

Use o `createIndex()` método para criar um índice de campo único. A sintaxe é: `db.collection.createIndex(<key>, <options>)`.

O `key` parâmetro é um documento JSON que especifica o campo e a ordem de classificação do índice:

```
{
  "<field>": <1 (ascending)|-1 (descending)>
}
```

O `options` parâmetro é um documento JSON que especifica as opções para o índice:

```
{
  "name": "<name>",
  "unique": <true | false>,
  "sparse": <true | false>,
  "partialFilterExpression": <filter expression>,
  "expireAfterSeconds": <seconds before expiry>
}
```

Consulte Propriedades do índice para ver exemplos de criação de índices de campo único.

Índices compostos

Os índices compostos armazenam informações de dois ou mais campos em uma coleção de documentos, permitindo consultas no primeiro campo ou em qualquer campo de prefixo. Esses índices otimizam o desempenho de consultas que filtram em vários campos simultaneamente ou combinam filtragem com operações de classificação. Eles também são eficazes para consultas com uma única condição nos campos indexados mais à esquerda. O banco de dados aproveita essas entradas de índice para localizar com eficiência documentos correspondentes sem realizar digitalizações completas da coleção.

Os índices de campo composto são benéficos quando:

- Você precisa filtrar em vários campos simultaneamente.
- Você precisa combinar a filtragem com as operações de classificação.

Propriedades de índice suportadas

Opção	3.6	4,0	5,0	8.0	Cluster elástico
name	Sim	Sim	Sim	Sim	Sim
único	Sim	Sim	Sim	Sim	Sim
esparso *	Sim	Sim	Sim	Sim	Sim
parcialFilterExpression *	Não	Não	Sim	Sim	Não

* As `partialFilterExpression` opções `sparse` e não podem ser usadas juntas na mesma definição de índice. Se você tentar criar um índice com essas opções, ele falhará com o seguinte erro:

```
Error in specification: cannot mix partialFilterExpression and sparse options
```

Criação de um índice composto

Use o `createIndex()` método para criar um índice composto. A sintaxe do método é: `db.collection.createIndex(<keys>, <options>)`

O `keys` parâmetro é um documento JSON que especifica os campos e a ordem de classificação do índice:

```
{
  "<field 1>": <1 (ascending)|-1 (descending)>,
  "<field 2>": <1 (ascending)|-1 (descending)>,
  ...
}
```

Observe que somente um campo pode ser uma matriz em um índice composto. Se você tentar criar um índice composto em dois ou mais campos de matriz, ele falhará com o seguinte erro:

```
multiple fields of compound index cannot be arrays
```

O `options` parâmetro é um documento JSON que especifica as opções para o índice:

```
{
  "name": "<name>",
  "unique": <true | false>,
  "sparse": <true | false>,
  "partialFilterExpression": <filter expression>
}
```

Consulte Propriedades do índice para ver exemplos de criação de índices compostos.

Multi-key índices

Para campos que têm um valor de matriz, um índice de várias chaves permite criar uma chave de índice para cada elemento na matriz. A indexação de uma matriz cria uma entrada de índice para cada elemento da matriz.

Multi-key índices são benéficos quando seu aplicativo consulta ou filtra documentos com frequência com base em valores em matrizes.

Propriedades de índice suportadas

Opção	3.6	4,0	5,0	8.0	Cluster elástico
name	Sim	Sim	Sim	Sim	Sim
único	Sim	Sim	Sim	Sim	Sim
esparso *	Sim	Sim	Sim	Sim	Sim
parcialFilterExpression *	Não	Não	Sim	Sim	Não
expirarAfterSeconds	Sim	Sim	Sim	Sim	Sim

* As `partialFilterExpression` opções `sparse` e não podem ser usadas juntas na mesma definição de índice. Se você tentar criar um índice com essas opções, ele falhará com o seguinte erro:

```
Error in specification: cannot mix partialFilterExpression and sparse options
```

Criação de um índice com várias chaves

Use o `createIndex()` método para criar um índice com várias chaves. A sintaxe do método é: `db.collection.createIndex(<key>, <options>)`

O `key` parâmetro é um documento JSON que especifica o campo e a ordem de classificação do índice:

```
{
  "<field>": <1 (ascending)|-1 (descending)>
}
```

O `options` parâmetro é um documento JSON que especifica as opções para o índice:

```
{
  "name": "<name>",
  "unique": <true | false>,
  "sparse": <true | false>,
  "partialFilterExpression": <filter expression>,
  "expireAfterSeconds": <seconds before expiry>
}
```

O exemplo a seguir cria um índice de várias chaves no `categories` campo classificado em ordem crescente com o nome: `book_categories`

```
db.collection.createIndex(
  {
    "categories": 1
  },
  {
    "name": "book_categories"
  }
)
```

Consulte Propriedades do índice para ver exemplos de criação de índices com várias chaves.

Índice de texto

Os índices de texto são úteis na pesquisa de palavras ou frases em campos de texto em documentos, permitindo que você execute a funcionalidade de pesquisa de texto completo. Você pode criar índices de texto de campo único ou índices de texto compostos com mais de um campo de texto. No entanto, você só pode ter um índice de texto por coleção. Para obter mais informações, consulte Realizar pesquisa de texto com o Amazon DocumentDB.

Propriedades de índice suportadas

Opção	3.6	4,0	5,0	8.0	Cluster elástico
name	Não	Não	Sim	Sim	Não

Criação de um índice de texto

Use o `createIndex()` método para criar um índice de texto. A sintaxe do método é: `db.collection.createIndex(<keys>, <options>)`

O `keys` parâmetro é um documento JSON que especifica o (s) campo (s) e o tipo de índice de texto:

```
{
  "<field 1>": "text",
  "<field 2>": "text"
}
```

O `options` parâmetro é um documento JSON que especifica o nome do índice:

```
{
  "name": "<name>"
}
```

Consulte Propriedades do índice para ver exemplos de criação de índices de texto.

Indexação geoespacial

Os índices geoespaciais são um tipo especializado de índice projetado para consultar e gerenciar com eficiência dados geoespaciais armazenados em uma coleção de documentos. O Amazon

DocumentDB oferece suporte a índices 2dsphere, que são projetados especificamente para lidar com dados geoespaciais em uma esfera (como a Terra). Isso permite cálculos e consultas precisos com base na geometria esférica.

Os índices geoespaciais são benéficos quando seus aplicativos precisam realizar consultas baseadas em localização, como:

- encontrar pontos de interesse próximos,
- determinar se um local está dentro de uma área específica
- cálculo de distâncias entre locais

Propriedades de índice suportadas

Opção	3.6	4,0	5,0	8.0	Cluster elástico
name	Sim	Sim	Sim	Sim	Sim

Criação de um índice geoespacial

Use o `createIndex()` método para criar um índice geoespacial. A sintaxe do método é: `db.collection.createIndex(<key>, <options>)`

O `key` parâmetro é um documento JSON que especifica o campo e o tipo de índice 2dsphere:

```
{
  "<field>": "2dsphere"
}
```

O `options` parâmetro é um documento JSON que especifica as opções para o índice:

```
{
  "name": "<name>"
}
```

Consulte Propriedades do índice para ver exemplos de criação de índices geoespaciais.

Índices vetoriais

Os índices vetoriais são um tipo especializado de índice projetado para consultar e gerenciar com eficiência dados vetoriais armazenados em uma coleção de documentos. O Amazon DocumentDB oferece suporte aos índices Hierarchical Navigable Small World (HNSW) e Inverted File with Flat Compression (IVFFlat).

Para obter mais informações, consulte Pesquisa vetorial para Amazon DocumentDB.

Os índices vetoriais são benéficos para casos de uso de aprendizado de máquina e IA generativa, como:

- pesquisa semântica
- recomendação de produto
- personalização
- chatbots
- detecção de fraudes
- detecção de anomalias

Propriedades de índice suportadas

Opção	3.6	4,0	5,0	8.0	Cluster elástico
name	Não	Não	Sim	Sim	Não

Criar um índice vetorial

Use o comando CreateIndex com o `runCommand()` método para criar um índice vetorial. A sintaxe é:

```
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]
    }
  }]
})
```

O parâmetro chave é um documento JSON que especifica o campo e o tipo de índice vetorial:

```
{
  "<field>": "vector"
}
```

Consulte Propriedades do índice para ver exemplos de criação de índices vetoriais.

Propriedades de índice

As propriedades do índice definem o comportamento e as características dos índices no Amazon DocumentDB. Compreender essas propriedades ajuda você a criar índices eficientes que atendam aos requisitos específicos do seu aplicativo.

Tópicos

- Propriedade do índice: nome
- Propriedade do índice: única
- Propriedade do índice: esparsa
- Propriedade do índice: parcial FilterExpression
- Propriedade do índice: expirar AfterSeconds
- Propriedade do índice: oculta

Propriedade do índice: nome

Tipos de índice compatíveis

Opção	3.6	4,0	5,0	8.0	Cluster elástico
campo único	Sim	Sim	Sim	Sim	Sim
compound	Sim	Sim	Sim	Sim	Sim
chave múltipla	Sim	Sim	Sim	Sim	Sim
texto	Não	Não	Sim	Sim	Não
Geoespacial	Sim	Sim	Sim	Sim	Sim
vetor	Não	Não	Sim	Sim	Não

Use a opção name para fornecer um nome opcional para o índice.

Todos os exemplos usam o seguinte documento de amostra:

```
{
  "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"
  }
)
```

Composta

```
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"
  }
)
```

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
    }
```

```
  }]
})
```

Propriedade do índice: única

Tipos de índice compatíveis

Opção	3.6	4,0	5,0	8.0	Cluster elástico
campo único	Sim	Sim	Sim	Sim	Sim
compound	Sim	Sim	Sim	Sim	Sim
chave múltipla	Sim	Sim	Sim	Sim	Sim

Use a opção exclusiva para garantir a exclusividade dos campos em todos os documentos de uma coleção.

Exemplos

Os exemplos a seguir mostram como criar índices exclusivos no seguinte documento de exemplo:

```
{
  "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

Crie um índice de campo único exclusivo no ProductID para garantir que o mesmo ProductID não exista em mais de um documento:

```
db.collection.createIndex(
  {
    "productId": 1
  },
  {
    name: "productId_unique",
    unique: true
  }
)
```

Composto

Crie um índice composto exclusivo no sku e no fabricante para garantir que a mesma combinação de sku e fabricante não exista em mais de um documento:

```
db.collection.createIndex(
  {
    "sku": 1,
    "manufacturer": 1
  },
  {
    name: "sku_and_manufacturer_unique",
    unique: true
  }
)
```

chave múltipla

Crie um índice exclusivo de várias chaves em códigos de barras para garantir que qualquer valor na matriz de códigos de barras não exista em mais de um documento:

```
db.collection.createIndex(
  {
    "barcodes": 1
  },
  {
    name: "barcodes_unique",
    unique: true
  }
)
```

A indexação de uma matriz cria uma entrada de índice para cada elemento da matriz. Por exemplo, se uma matriz tiver 50 itens, ela terá 50 entradas de índice. Por isso, índices exclusivos de várias chaves impõem exclusividade em todos os itens individuais. Por exemplo, os documentos a seguir violarão a restrição exclusiva no índice do campo da matriz de valores:

```
{ "values": [ 1, 2, 3] }
{ "values": [ 3, 2 ] }   --> 3 and 2 already exist
{ "values": [ 1 ] }      --> 1 already exists
```

Observe o seguinte comportamento com índices exclusivos:

1. Se você criar um índice exclusivo em dados existentes em que dois (ou mais) documentos tenham os mesmos valores para os campos indexados, a criação do índice falhará com o seguinte erro: `could not create unique index: <collection> index: <index name>`
2. Se você inserir um documento em que o valor do campo indexado corresponda ao valor desse campo em outro documento, a inserção falhará com o seguinte erro: `E11000 duplicate key error collection: <collection> index: <index name>`
3. Se você atualizar um documento existente para que o novo valor do campo indexado corresponda ao valor desse campo em outro documento, a atualização falhará com o seguinte erro: `E11000 duplicate key error collection: <collection> index: <index name>`
4. Se o campo indexado estiver ausente em um documento, o valor será tratado como nulo. As compilações, inserções e atualizações do índice falharão conforme descrito acima se o campo indexado estiver ausente em dois (ou mais) documentos.

Propriedade do índice: esparsa

Tipos de índice compatíveis

Opção	3.6	4,0	5,0	8.0	Cluster elástico
campo único	Sim	Sim	Sim	Sim	Sim
compound	Sim	Sim	Sim	Sim	Sim
chave múltipla	Sim	Sim	Sim	Sim	Sim

Use a opção esparsa para ignorar a indexação de documentos sem os campos indexados, reduzindo o tamanho do índice e economizando espaço na memória. Como o tamanho do índice é menor, as consultas que o usam são mais eficientes. Para que uma consulta utilize um índice esparso, você deve usar a cláusula $exists nos campos indexados. Se você omitir a cláusula $exists, o Amazon DocumentDB não usará o índice esparso.

Exemplos

Os exemplos a seguir mostram como criar índices esparsos no seguinte documento de exemplo:

```
{
  "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')
}
```

Observe que os campos de resenhas, material e cor não existem em todos os documentos.

campo único

Crie um índice de campo único esparso no campo de material:

```
db.collection.createIndex(
  {
    "material": 1
  },
  {
    "name": "material_sparse",
    "sparse": true
  }
)
```

Esse índice será usado ao encontrar todos os produtos que tenham um valor para o material:

```
db.collection.find({
  "material": {
    $exists: true
  }
})
```

composto

Crie um índice composto esparso nos campos de material e cor:

```
db.collection.createIndex(
```

```
  {
    "material": 1,
    "color": 1
  },
  {
    "name": "material_and_color_sparse",
    "sparse": true
  }
)
```

Esse índice será usado ao encontrar todos os produtos que tenham qualquer combinação de valores de material e cor:

```
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
      }
    }
  ]
})
```

chave múltipla

Crie um índice esparso de várias chaves na matriz de avaliações:

```
db.collection.createIndex(
  {
    "reviews": 1
  },
  {
    "name": "reviews_sparse",
    "sparse": true
  }
)
```

Esse índice será usado ao encontrar todos os produtos que tenham uma matriz de avaliações:

```
db.collection.find({
  "reviews": {
    $exists: true
  }
})
```

Propriedade do índice: parcial FilterExpression

Tipos de índice compatíveis

Opção	3.6	4,0	5,0	8.0	Cluster elástico
campo único	Não	Não	Sim	Sim	Não
compound	Não	Não	Sim	Sim	Não
chave múltipla	Não	Não	Sim	Sim	Não
Geoespacial	Não	Não	Não	Não	Não

Use a FilterExpression opção parcial para criar um índice parcial que inclua somente documentos que atendam a uma condição de filtro especificada. Isso permite criar índices mais eficientes indexando somente um subconjunto de documentos em uma coleção, em vez de indexar todos os documentos, reduzindo o tamanho do índice e economizando espaço na memória. Como o tamanho do índice é menor, as consultas que o usam são mais eficientes. O Amazon DocumentDB usará o índice parcial nos seguintes cenários:

- O predicado da consulta corresponde exatamente à expressão do filtro de índice parcial.
- O resultado esperado do filtro de consulta é um subconjunto lógico do filtro parcial.
- Um subpredicado da consulta pode ser usado em conjunto com outros índices.

Para obter mais informações, consulte Índice parcial.

Exemplos

Os exemplos a seguir mostram como criar índices parciais no seguinte documento de exemplo:

```
{
  "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

Crie um índice parcial de campo único no fabricante para produtos que tenham uma cor de cinza espacial:

```
db.collection.createIndex(
  {
    "manufacturer": 1
  },
  {
    "name": "manufacturer_space_gray",
    "partialFilterExpression": {
      "color": {
        $eq: "Space Gray"
      }
    }
  }
)
```

Esse índice será usado ao encontrar produtos na cor Space Grey pelo fabricante:

```
db.collection.find({
  "manufacturer": "The Manufacturer",
  "color": {
    $eq: "Space Gray"
  }
})
```

composto

Crie um índice composto parcial no fabricante e na cor para produtos que tenham um material de policarbonato:

```
db.collection.createIndex(
  {
    "manufacturer": 1,
    "color": 1
  },
  {
    "name": "manufacturer_and_color_polycarbonate",
    "partialFilterExpression": {
      "material": {
        $eq: "Polycarbonate"
      }
    }
  }
)
```

Esse índice será usado ao encontrar produtos de material de policarbonato por fabricante e cor:

```
db.collection.find({
  "manufacturer": "The Manufacturer",
  "color": "Space Gray",
  "material": {
    $eq: "Polycarbonate"
  }
})
```

chave múltipla

Crie um índice parcial de várias teclas nas etiquetas de produtos que têm um fabricante do fabricante The Manufacturer:

```
db.collection.createIndex(
  {
    "tags": 1
  },
  {
    "name": "tags_the_manufacturer",
    "partialFilterExpression": {
      "manufacturer": {
        $eq: "The Manufacturer"
      }
    }
  }
)
```

Esse índice será usado ao encontrar produtos marcados com impressoras que tenham um fabricante de The Manufacturer:

```
db.collection.find({
  "tags": "printer",
  "manufacturer": {
    $eq: "The Manufacturer"
  }
})
```

Propriedade do índice: expirar AfterSeconds

Tipos de índice compatíveis

Tipo de índice	3.6	4,0	5,0	8.0	Cluster elástico
campo único	Sim	Sim	Sim	Sim	Sim
chave múltipla	Sim	Sim	Sim	Sim	Sim

Use a AfterSeconds opção expirar para criar um índice de tempo de vida (TTL). Os índices TTL permitem que você exclua documentos com base na idade, com base na condição de tempo limite

de cada documento. Quando um documento atinge a idade TTL especificada, ele é excluído da coleção.

Para obter as melhores práticas com exclusões de TTL, consulte. Workloads de TTL e séries temporais

Exemplos

Os exemplos a seguir mostram como criar índices TTL no seguinte documento de exemplo:

```
{
  "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

Crie um índice TTL em LastUpdated para excluir documentos que não foram atualizados em 90 dias:

```
db.collection.createIndex(
  {
    "lastUpdated": 1
  },
  {
    "name": "lastUpdated_ttl",
    "expireAfterSeconds": 7776000
  }
)
```

chave múltipla

Crie um índice TTL em reviews.review_date para excluir documentos que não foram revisados no ano passado:

```
db.collection.createIndex(
  {
    "reviews.review_date": 1
  },
  {
    "name": "reviews_review_date_ttl",
    "expireAfterSeconds": 31536000
  }
)
```

Observe que, se o índice TTL estiver em um campo de matriz, todos os itens na matriz serão verificados. Se algum carimbo de data/hora na matriz atender à condição de tempo limite, o documento será excluído.

Propriedade do índice: oculta

O Amazon DocumentDB oferece suporte à propriedade de `hidden` índice começando com Amazon DocumentDB 8.0.1+ e Amazon DocumentDB 5.0.2+.

Os índices ocultos não são visíveis para o planejador de consultas e não podem ser usados para dar suporte a uma consulta. Ao ocultar um índice do planejador, você pode avaliar o impacto potencial de descartar o índice sem realmente eliminá-lo. Se o impacto for negativo, você poderá exibir o índice em vez de precisar recriar um índice descartado.

A `hidden` propriedade é compatível com todos os tipos de índice, incluindo índices de campo único, composto, multichave, texto, geoespacial e vetorial.

Comportamento

Além de ficarem ocultos do planejador, os índices ocultos são totalmente mantidos e aplicados. Por exemplo:

- Se um índice oculto for exclusivo, o índice ainda aplicará sua restrição exclusiva aos documentos.
- Se um índice oculto for um índice TTL, o índice ainda expirará os documentos.
- Índices ocultos são incluídos em `listIndexes` e nos `db.collection.getIndexes()` resultados.
- Os índices ocultos são atualizados nas operações de gravação na coleção e continuam consumindo espaço em disco e memória. Como resultado, eles são incluídos em operações estatísticas, como `db.collection.stats()` `$indexStats` e.
- Ocultar um índice não oculto ou exibir um índice oculto redefine seu. `$indexStats` Ocultar um índice já oculto ou exibir um índice já não oculto não é redefinido. `$indexStats`

Restrições

- Você não pode ocultar o `_id` índice.
- Você não pode usar `cursor.hint()` para forçar o uso de um índice oculto.
- `$merge`falhará se o índice exclusivo necessário nos on campos estiver oculto. Torne o índice visível antes de executar`$merge`.

Exemplos

Crie um índice oculto

Para criar um índice no estado oculto, use o `db.collection.createIndex()` método e defina a `hidden` opção como `true` na definição do índice.

Por exemplo, a operação a seguir cria um índice ascendente oculto no `productId` campo:

```
db.collection.createIndex(
  { "productId": 1 },
  { "name": "productId_1", "hidden": true }
)
```

Para verificar, executar `db.collection.getIndexes()` ou `listIndexes` acessar a coleção. Um índice oculto inclui `"hidden": true` em sua definição.

```
db.collection.getIndexes()
[
  {
    "v": 4,
    "key": { "_id": 1 },
    "name": "_id_"
  },
  {
    "v": 4,
    "key": { "productId": 1 },
    "name": "productId_1",
    "hidden": true
  }
]
```

A opção de índice só `hidden` é retornada se o valor for`true`.

Ocultar um índice existente

Use o `collMod` comando para ocultar um índice existente. Identifique o índice por its `name` ou por its `keyPattern` e defina-o como `hiddentrue`.

O exemplo a seguir oculta um índice por nome.

```
db.runCommand({
```

```
  collMod: "collection",
  index: {
    name: "productId_1", // Specify the index name
    hidden: true
  }
})
```

Você também pode identificar o índice por seu padrão de chave.

```
db.runCommand({
  collMod: "collection",
  index: {
    keyPattern: { productId: 1 }, // Specify the index key specification document
    hidden: true
  }
})
```

Exibir um índice existente

Para tornar um índice oculto visível novamente, use o `collMod` comando com `hidden` set to`false`.

```
db.runCommand({
  collMod: "collection",
  index: { name: "productId_1", hidden: false }
})
```

Como os índices são totalmente mantidos enquanto estão ocultos, o índice fica imediatamente disponível para uso quando não está oculto.

Usar fluxos de alterações com o Amazon DocumentDB

O atributo de fluxos de alterações do Amazon DocumentDB (compatível com MongoDB) fornece uma sequência ordenada por tempo das alterações que ocorrem nas coleções do cluster. É possível ler eventos de um fluxo de alterações para implementar muitos casos de uso diferentes, incluindo o seguinte:

- Notificação de alterações
- Full-text pesquise com o Amazon OpenSearch Service (OpenSearch Service)
- Analytics com Amazon Redshift

As aplicações podem usar os fluxos de alterações para assinar as alterações de dados em coleções individuais. Os eventos dos fluxos de alterações são ordenados à medida que ocorrem no cluster e são armazenados por 3 horas (por padrão) após a gravação do evento. O período de retenção pode ser estendido até 7 dias usando o parâmetro `change_stream_log_retention_duration`. Para modificar o período de retenção do fluxo de alterações, consulteModificar a duração da retenção do log do fluxo de alterações.

Tópicos

- Operações compatíveis
- Eventos expandidos
- Faturamento
- Limitações
- Ativar fluxos de alterações
- Exemplo: usar fluxos de alterações com Python
- Pesquisa completa de documentos
- Usando $change StreamSplitLargeEvent para dividir grandes eventos em fragmentos
- Retomar um fluxo de alterações
- Retomando um fluxo de mudanças com start AtOperationTime
- Retomando um fluxo de alterações com a postagem BatchResumeToken
- Transações em fluxos de alterações
- Modificar a duração da retenção do log do fluxo de alterações
- Usar fluxos de alterações em instâncias secundárias

Operações compatíveis

O Amazon DocumentDB oferece suporte às seguintes operações para fluxos de alterações:

- Todos os eventos de alteração compatíveis na API `db.collection.watch()`, `db.watch()` e `client.watch()` do MongoDB.
- Pesquisa completa de documentos para atualizações.
- Estágios de agregação: `$match` `$project``$redact`,,`$addFields`,`$replaceRoot`, e. `$changeStreamSplitLargeEvent`
- Retomando um fluxo de alterações a partir de um token de currículo

- Retomar um fluxo de alterações de um carimbo de data/hora usando `startAtOperation` (aplicável ao Amazon DocumentDB 4.0+)
- Notificação de evento DDL para `createCollection` usar o `showExpandedEvents` parâmetro (aplicável ao Amazon DocumentDB 8.0 a partir da versão 8.0.2 do mecanismo).
- Notificação de evento DDL para `createIndexes` usar o `showExpandedEvents` parâmetro (aplicável ao Amazon DocumentDB 8.0 a partir da versão 8.0.2 do mecanismo).

Eventos expandidos

> **Note**
>
> Eventos expandidos estão disponíveis no Amazon DocumentDB 8.0 a partir da versão 8.0.2 do mecanismo.

Os fluxos de alteração suportam notificações de eventos de linguagem de definição de dados (DDL) e. `createCollection` `createIndexes` Para incluir eventos expandidos, abra um cursor de fluxo de alterações com a `showExpandedEvents` opção.

Por exemplo:

mongosh Method

```
let cur = db.names.watch( [ ], {
   showExpandedEvents: true
} )
cur.next()
```

Aggregation Stage

```
let cur = db.names.aggregate( [
   { $changeStream: { showExpandedEvents: true } }
] )
cur.next()
```

Faturamento

O atributo de fluxos de alterações do Amazon DocumentDB é desativado por padrão e não incorre em cobranças adicionais até ser ativado e usado. O uso dos fluxos de alterações em um cluster resulta em custos adicionais de ler e escrever IOs e de armazenamento. É possível usar a operação `modifyChangeStreams` de API para habilitar esse atributo para seu cluster. Para obter mais informações sobre preços, consulte Preços do Amazon DocumentDB.

Limitações

Os fluxos de alterações têm as seguintes limitações no Amazon DocumentDB:

- No Amazon DocumentDB 3.6 e no Amazon DocumentDB 4.0, os fluxos de alterações só podem ser abertos de uma conexão com a instância primária de um cluster do Amazon DocumentDB. A leitura de fluxos de alteração em uma instância de réplica não é compatível com o Amazon DocumentDB 3.6. e no Amazon DocumentDB 4.0. Ao chamar a operação de API `watch()`, é necessário especificar uma preferência de leitura `primary` para garantir que todas as leituras sejam direcionadas à instância principal (consulte a seção Exemplo).
- No Amazon DocumentDB 5.0, os fluxos de alterações podem ser abertos tanto da instância primária quanto da instância secundária, incluindo clusters globais. É possível especificar uma preferência de leitura secundária para redirecionar os fluxos de alteração para instâncias secundárias. Consulte Usar fluxos de alterações em instâncias secundárias para saber sobre outras práticas recomendadas e limitações.
- Os eventos gravados em um fluxo de alterações para uma coleção estão disponíveis por até 7 dias (o padrão é 3 horas). Os dados de fluxos de alterações são excluídos após a janela de duração de retenção de log, mesmo que nenhuma nova alteração tenha ocorrido.
- Uma operação de gravação de longa duração em uma coleção como `updateMany` ou `deleteMany` pode interromper temporariamente a gravação dos eventos dos fluxos de alterações até que ela seja concluída.
- O Amazon DocumentDB não oferece suporte ao log de operações do MongoDB (`oplog`).
- Com o Amazon DocumentDB, é necessário ativar explicitamente os fluxos de alterações em determinada coleção.
- Se o tamanho total de um evento de fluxos de alteração (incluindo os dados de alteração e o documento completo, se solicitado) for maior que`16 MB`, o cliente experimentará uma falha de leitura no fluxo de alterações. Para evitar isso, use o `$changeStreamSplitLargeEvent` palco para dividir o evento em fragmentos.

- Atualmente, o driver Ruby não é aceito ao usar `db.watch()` e `client.watch()` com o Amazon DocumentDB 3.6.
- A saída do comando `updateDescription` nos fluxos de alterações é diferente no Amazon DocumentDB e no MongoDB quando o valor atualizado do campo é o mesmo do anterior:
  - O Amazon DocumentDB não retornará um campo na saída de `updateDescription` se o campo fornecido for especificado no comando `$set` e seu valor de destino já for igual ao valor de origem.
  - O MongoDB retorna o campo na saída, mesmo que o valor especificado seja igual ao valor atual.

Ativar fluxos de alterações

É possível habilitar os fluxos de alterações do Amazon DocumentDB em todas as coleções em um determinado banco de dados ou apenas em coleções específicas. Veja a seguir os exemplos de como habilitar os fluxos de alterações em diferentes casos de uso com o shell do Mongo. As strings vazias são tratadas como curingas na especificação de nomes de banco de dados e coleções.

```
//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});
```

Os fluxos de alterações serão ativados em uma coleção se qualquer uma destas opções for verdadeira:

- O banco de dados e a coleção estão explicitamente ativados.
- O banco de dados que contém a coleção está ativado.
- Todos os bancos de dados estão ativados.

Eliminar uma coleção de um banco de dados não desativará os fluxos de alterações dessa coleção se o banco de dados pai também tiver fluxos de alterações ativados, ou se todos os bancos de dados do cluster estiverem ativados. Se uma coleção for criada com o mesmo nome da coleção excluída, os fluxos de alterações serão ativados para essa coleção.

É possível listar todos os fluxos de alterações ativados para o cluster usando o estágio de agregação do pipeline `$listChangeStreams`. Todas as etapas de agregação compatíveis com o Amazon DocumentDB podem ser usadas no pipeline para processamento adicional. Se uma coleção ativada anteriormente tiver sido desativada, ela não aparecerá na saída `$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:{}}));
```

Exemplo: usar fluxos de alterações com Python

Veja a seguir um exemplo do uso de um fluxo de alterações do Amazon DocumentDB com Python no nível da coleção.

```
import os
import sys
from pymongo import MongoClient, ReadPreference

username = "DocumentDBusername"
password = <Insert your password>

clusterendpoint = "DocumentDBClusterEndpoint”
client = MongoClient(clusterendpoint, username=username, password=password, tls='true',
 tlsCAFile='global-bundle.pem')

db = client['bar']

#While ‘Primary’ is the default read preference, here we give an example of
#how to specify the required read preference when reading the change streams
coll = db.get_collection('foo', read_preference=ReadPreference.PRIMARY)
#Create a stream object
stream = coll.watch()
#Write a new document to the collection to generate a change event
coll.insert_one({'x': 1})
#Read the next change event from the stream (if any)
print(stream.try_next())

"""
Expected Output:
{'_id': {'_data': '015daf94f600000002010000000200009025'},
'clusterTime': Timestamp(1571788022, 2),
'documentKey': {'_id': ObjectId('5daf94f6ea258751778163d6')},
'fullDocument': {'_id': ObjectId('5daf94f6ea258751778163d6'), 'x': 1},
'ns': {'coll': 'foo', 'db': 'bar'},
'operationType': 'insert'}
"""
```

```
#A subsequent attempt to read the next change event returns nothing, as there are no
 new changes
print(stream.try_next())

"""
Expected Output:
None
"""

#Generate a new change event by updating a document
result = coll.update_one({'x': 1}, {'$set': {'x': 2}})
print(stream.try_next())

"""
Expected Output:
{'_id': {'_data': '015daf99d40000000101000000100009025'},
'clusterTime': Timestamp(1571789268, 1),
'documentKey': {'_id': ObjectId('5daf9502ea258751778163d7')},
'ns': {'coll': 'foo', 'db': 'bar'},
'operationType': 'update',
'updateDescription': {'removedFields': [], 'updatedFields': {'x': 2}}}
"""
```

Veja a seguir um exemplo do uso de um fluxo de alterações do Amazon DocumentDB com Python no nível do banco de dados.

```
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'}
"""
```

Pesquisa completa de documentos

O evento de alteração de atualização não inclui o documento completo, apenas a alteração que foi feita. Se o seu caso de uso exigir o documento completo afetado por uma atualização, é possível ativar a pesquisa completa do documento na abertura do fluxo.

O documento `fullDocument` de um evento de fluxos de alterações de atualização representa a versão mais atual do documento atualizado no momento em que ele é pesquisado. Se ocorrerem alterações entre a operação de atualização e a pesquisa do `fullDocument`, o documento `fullDocument` poderá não representar o estado dele no momento da atualização.

Para criar um objeto de fluxo com a pesquisa de atualização ativada, use este exemplo:

```
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()
```

A saída do objeto de fluxo será semelhante ao seguinte:

```
{'_id': {'_data': '015daf9b7c000000010100000100009025'},
'clusterTime': Timestamp(1571789692, 1),
'documentKey': {'_id': ObjectId('5daf9502ea258751778163d7')},
'fullDocument': {'_id': ObjectId('5daf9502ea258751778163d7'), 'x': 3},
'ns': {'coll': 'foo', 'db': 'bar'},
'operationType': 'update',
'updateDescription': {'removedFields': [], 'updatedFields': {'x': 3}}}
```

Usando $change StreamSplitLargeEvent para dividir grandes eventos em fragmentos

> **Note**
>
> O StreamSplitLargeEvent estágio $change está disponível no Amazon DocumentDB 8.0 a partir da versão 8.0.2 do mecanismo.

Quando você usa o `$changeStreamSplitLargeEvent` palco, eventos de alteração maiores que 16 MB são divididos em fragmentos que são retornados sequencialmente por meio do cursor do fluxo de alterações. Eventos menores que 16 MB são retornados sem divisão. O palco divide um evento ao longo de seus limites de campo de nível superior; ele não se divide em um único campo nem se transforma em subdocumentos. Por causa disso, se um único campo de nível superior tiver mais de 16 MB, o palco não poderá produzir um fragmento válido para ele e, em vez disso, retornará um erro.

Como melhor prática, recomendamos evitar o uso de, `$changeStreamSplitLargeEvent` quando possível, reduzindo o tamanho do evento.

O `$changeStreamSplitLargeEvent` estágio só pode ser usado em um `$changeStream` pipeline e deve ser usado apenas uma vez como último estágio.

Cada fragmento tem o seu próprio `resumeToken` e o seu. `postBatchResumeToken` A retomada a partir de um token de fragmento não final (usando um token de currículo normal ou uma postagemBatchResumeToken) entrega o fragmento subsequente. No caso em que o token corresponde ao fragmento final de um evento, o cursor do fluxo de alterações retorna o evento subsequente.

Retomar um fluxo de alterações

É possível retomar um fluxo de alterações posteriormente usando um token de retomada, que é igual ao campo `_id` do último documento de evento de alteração 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}}
"""
```

Retomando um fluxo de mudanças com start **`AtOperationTime`**

É possível retomar um fluxo de alterações posteriormente a partir de um carimbo de data/hora específico usando `startAtOperationTime`.

> **Note**
>
> A capacidade de usar `startAtOperationTime` está disponível no Amazon DocumentDB 4.0+. Ao usar `startAtOperationTime`, o cursor do fluxo de alterações retornará apenas as alterações que ocorreram no carimbo de data/hora especificado ou após ele. Os comandos `startAtOperationTime` e `resumeAfter` são mutuamente exclusivos e, portanto, não podem ser usados 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}}
"""
```

Retomando um fluxo de alterações com a postagem **`BatchResumeToken`**

O fluxo de alterações do Amazon DocumentDB agora retorna um campo adicional chamado `postBatchResumeToken`. Esse campo é retornado do comando `$changestream` e do comando `getMore`.

Exemplo do comando `$changestream` em Python:

```
db.command({"aggregate": "sales", "pipeline": [{ "$changeStream": {}}], "cursor":
 {"batchSize": 1}})
```

Saída esperada:

```
cursor" : {
   "firstBatch" : [ ],
   "postBatchResumeToken" : {"_data" : "0167c8cbe60000000004"},
   "id" : NumberLong("9660788144470"),
   "ns" : "test.sales"
}
```

Exemplo do comando `getMore` em Python:

```
db.command({"getMore": NumberLong(<cursor id>), "collection": "sales", "batchSize":
 1 })
```

Saída esperada

```
cursor" : {
   "nextBatch" : [ ],
   "postBatchResumeToken" : {"_data" : "0167c8cbe60000000004"},
   "id" : NumberLong("9660788144470"),
   "ns" : "test.sales"
}
```

O campo `postBatchResumeToken` pode ser usado para abrir novos cursores de fluxo de alterações no campo `resumeAfter`, semelhante à forma como o token de retomada é usado.

Abra um fluxo começando após o `postBatchResumeToken` selecionado:

```
post_batch_resume_token = output['cursor']['postBatchResumeToken']
stream = db.watch(full_document='updateLookup', resume_after=post_batch_resume_token)
```

Ao contrário de um token de retomada normal, que sempre corresponde a uma entrada de log de operações (oplog) que reflete um evento real, `postBatchResumeToken` corresponde a uma entrada de oplog que o fluxo de alterações examinou no servidor, o que não é necessariamente uma alteração correspondente.

A tentativa de continuar com um antigo token de retomada regular forçará o banco de dados a verificar todas as entradas do oplog entre o carimbo de data e hora especificado e a hora atual. Isso pode gerar muitas consultas internamente com a varredura de cada subconsulta por um pequeno período de tempo. Isso causará um pico no uso da CPU e degradará a performance do banco de dados. Continuar com o último `postBatchResumeToken` ignora a verificação de entradas não correspondidas do oplog.

Transações em fluxos de alterações

Os eventos do fluxo de alterações não conterão eventos de transações and/or abortadas não confirmadas. Por exemplo, se você iniciar uma transação com uma operação `INSERT` e uma operação `UPDATE`, e se sua operação `INSERT` tiver êxito, mas a operação `UPDATE` falhar, a transação será revertida. Como esta transação foi revertida, seu fluxo de alterações não conterá nenhum evento para esta transação.

Modificar a duração da retenção do log do fluxo de alterações

Você pode modificar a duração da retenção do log do stream de alterações para ficar entre 1 hora e 7 dias usando o Console de gerenciamento da AWS ou AWS CLI o.

Using the Console de gerenciamento da AWS

Como modificar a duração da retenção do log do fluxo de alterações

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Grupos de parâmetros.

   > **Tip**
   >
   > Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

3. No painel Grupos de parâmetros, escolha o grupo de parâmetros de cluster associado ao cluster. Para identificar o grupo de parâmetros de cluster associado ao cluster, consulte Determinando o grupo de parâmetros de um cluster do Amazon DocumentDB.

4. A página resultante mostra os parâmetros e os detalhes correspondentes para seu grupo de parâmetros do cluster. Selecione o parâmetro `change_stream_log_retention_duration`.
5. No canto superior direito da página, selecione Editar para alterar o valor do parâmetro. O parâmetro `change_stream_log_retention_duration` pode ser modificado para ficar entre 1 hora e 7 dias.
6. Faça a alteração e escolha Modificar parâmetro de cluster para salvar as alterações. Para descartar as alterações, escolha Cancelar.

Using the AWS CLI

Para modificar o parâmetro `change_stream_log_retention_duration` de um grupo de parâmetros de cluster, use a operação `modify-db-cluster-parameter-group` com os parâmetros a seguir:

- **`--db-cluster-parameter-group-name`** — Obrigatório. O nome do grupos de parâmetros de cluster que você está modificando. Para identificar o grupo de parâmetros de cluster associado ao cluster, consulte Determinando o grupo de parâmetros de um cluster do Amazon DocumentDB.
- **`--parameters`** — Obrigatório. O parâmetro que você está modificando. Cada entrada de parâmetro deve incluir o seguinte:
  - **`ParameterName`** — O nome do parâmetro que você está modificando. Neste caso, é `change_stream_log_retention_duration`
  - **`ParameterValue`** — O novo valor para esse parâmetro.
  - **`ApplyMethod`** — como você deseja aplicar as alterações nesse parâmetro. Os valores permitidos são `immediate` e `pending-reboot`.

> **Note**
>
> Os parâmetros com `ApplyType` de `static` devem ter um `ApplyMethod` de `pending-reboot`.

1. Para alterar os valores do parâmetro `change_stream_log_retention_duration`, execute o seguinte comando e substitua `parameter-value` pelo valor para o qual deseja modificar o parâmetro.

Para Linux, macOS ou 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"
```

A saída dessa operação é semelhante ao seguinte (formato JSON).

```
{
    "DBClusterParameterGroupName": "sample-parameter-group"
}
```

2. Aguarde pelo menos 5 minutos.
3. Liste os valores de parâmetro de `sample-parameter-group` para garantir que suas alterações foram feitas.

   Para Linux, macOS ou 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
   ```

   A saída dessa operação é semelhante ao seguinte (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**
>
> A retenção de log de fluxo de alterações não excluirá logs mais antigos que o valor `change_stream_log_retention_duration` configurado até que o tamanho do log seja maior que (>) 51.200 MB.

Usar fluxos de alterações em instâncias secundárias

Para começar a usar o fluxo de alterações em instâncias secundárias, abra o cursor do fluxo de alterações com `readPreference` como secundário.

É possível abrir um cursor de fluxo de alterações para observar os eventos de alteração em uma coleção específica ou em todas as coleções em um cluster ou banco de dados. É possível abrir um

cursor do fluxo de alterações em qualquer instância do Amazon DocumentDB e buscar documentos do fluxo de alterações das instâncias do escritor e do leitor. É possível compartilhar tokens de fluxo de alterações (como `resumeToken` ou `startOperationTime`) em diferentes cursores de fluxo de alterações abertos em uma instância de gravador e leitor.

Exemplo

```
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)
```

Diretrizes e limitações para fluxos de alterações em instâncias secundárias

- Os eventos do fluxo de alterações precisam ser replicados da instância primária para as instâncias secundárias. Você pode monitorar o atraso da `DBInstanceReplicaLag` métrica na Amazon CloudWatch.
- Os carimbos de data e hora em instâncias secundárias nem sempre estão sincronizados com a instância primária. Nesse caso, espere atrasos para que o carimbo de data e hora da instância secundária possa se atualizar. Como prática recomendada, use `startAtOperationTime` ou `resumeToken` para iniciar o relógio na instância secundária.
- Será possível ter um throughput menor em instâncias secundárias em comparação com a instância primária se o tamanho do documento for grande e você estiver fazendo

`fullDocumentLookup`, e se houver um alto workload de gravação simultânea na instância primária. Como melhor prática, monitore a taxa de acertos do cache do buffer no secundário e certifique-se de que a taxa de acertos do cache do buffer seja alta.

Transações no Amazon DocumentDB

O Amazon DocumentDB (com compatibilidade com o MongoDB) oferece suporte a transações no Amazon DocumentDB 4.0 e versões posteriores. É possível realizar transações em vários documentos, extratos, coleções e bancos de dados. As transações simplificam o desenvolvimento de aplicações, permitindo que você execute operações atômicas, consistentes, isoladas e duráveis (ACID) em um ou mais documentos em um cluster do Amazon DocumentDB. Os casos de uso comuns para transações incluem processamento financeiro, atendimento e gerenciamento de pedidos e criação de jogos para vários jogadores.

Não há custo adicional para transações. Você paga apenas pelo iOS de leitura e gravação que você consome como parte das transações.

Tópicos

- Requisitos
- Práticas recomendadas
- Limitações
- Monitoramento e diagnóstico
- Nível de isolamento de transação
- Casos de uso
- Comandos compatíveis
- Capacidades sem suporte
- Sessões
- Erros de transação

Requisitos

Para usar o atributo de transações, você precisa atender aos seguintes requisitos:

- Você deve usar o Amazon DocumentDB 4.0 ou posterior.
- Você deve usar um driver compatível com o MongoDB 4.0 ou posterior.

Práticas recomendadas

Aqui estão algumas das melhores práticas para que você possa aproveitar ao máximo as transações com o Amazon DocumentDB.

- Sempre confirme ou aborte a transação depois que ela for concluída. Deixar uma transação em um estado incompleto consome recursos do banco de dados e pode causar conflitos de gravação.
- É recomendável manter as transações com o menor número de comandos necessários. Se você tiver transações com vários extratos que possam ser divididos em várias transações menores, é recomendável fazer isso para reduzir a probabilidade de um tempo limite. Sempre tente criar transações curtas, não leituras de longa duração.

Limitações

- O Amazon DocumentDB não oferece suporte a cursores em uma transação.
- O Amazon DocumentDB não pode criar novas coleções em uma transação e não pode query/update contra coleções inexistentes.
- Document-level os bloqueios de gravação estão sujeitos a um tempo limite de 1 minuto, que não é configurável pelo usuário.
- Nas versões de mecanismo anteriores à 8.0.2, não há suporte para gravações repetidas. A partir da versão 8.0.2 do mecanismo, as gravações que podem ser repetidas são suportadas fora das transações para operações de gravação nas quais cada gravação individual afeta no máximo um documento, e as operações de confirmação e cancelamento da transação podem ser repetidas. Para obter mais informações, consulte Gravações que podem ser repetidas no Amazon DocumentDB.
- Cada instância do Amazon DocumentDB tem um limite superior no número de transações simultâneas que podem ser abertas na instância. Para obter mais informações, consulte Cotas de instâncias.
- Para uma determinada transação, o tamanho do log de transações deve ser menor que 32MB.
- O Amazon DocumentDB oferece suporte a transações `count()` internas, mas nem todos os drivers oferecem suporte a esse recurso. Uma alternativa é usar a `countDocuments()` API, que traduz a consulta de contagem em uma consulta de agregação no lado do cliente.
- As transações têm um limite de execução de um minuto e as sessões têm um tempo limite de 30 minutos. Se uma transação expirar, ela será abortada e quaisquer comandos subsequentes emitidos na sessão para a transação existente produzirão o seguinte erro:

```
WriteCommandError({
"ok" : 0,
"operationTime" : Timestamp(1603491424, 627726),
"code" : 251,
"errmsg" : "Given transaction number 0 does not match any in-progress transactions."
}
```

Monitoramento e diagnóstico

Com o suporte para transações no Amazon DocumentDB 4.0, CloudWatch métricas adicionais foram adicionadas para ajudar você a monitorar suas transações.

Novas CloudWatch métricas

- `DatabaseTransactions`: o número de transações abertas realizadas em um período de um minuto.
- `DatabaseTransactionsAborted`: o número de transações abortadas realizadas em um período de um minuto.
- `DatabaseTransactionsMax`: o número máximo de transações abertas em um período de um minuto.
- `TransactionsAborted`: o número de transações abortadas em uma instância em um período de um minuto.
- `TransactionsCommitted`: o número de transações confirmadas em uma instância em um período de um minuto.
- `TransactionsOpen`: o número de transações abertas em uma instância realizadas em um período de um minuto.
- `TransactionsOpenMax`: o número máximo de transações abertas em uma instância em um período de um minuto.
- `TransactionsStarted`: o número de transações iniciadas em uma instância em um período de um minuto.

> **Note**
>
> Para obter mais CloudWatch métricas para o Amazon DocumentDB, acesseMonitorando o Amazon DocumentDB com CloudWatch.

Além disso, novos campos foram adicionados a `currentOp` `lsid`,`transactionThreadId`, e um novo estado para "`idle transaction`" e `serverStatus` transações: `currentActive`, `currentInactive`, `currentOpen`, `totalAborted`, `totalCommitted`e `totalStarted`.

Nível de isolamento de transação

Ao iniciar uma transação, você pode especificar o `readConcern` e `writeConcern` conforme mostrado no exemplo a seguir:

```
mySession.startTransaction({readConcern: {level: 'snapshot'}, writeConcern:
{w: 'majority'}});
```

Para `readConcern`, o Amazon DocumentDB oferece suporte ao isolamento de snapshots por padrão. Se uma `readConcern` das opções local, disponível ou majoritária for especificada, o Amazon DocumentDB atualizará o nível `readConcern` para snapshot. O Amazon DocumentDB não suporta o linearizável `readConcern` e especificar esse problema de leitura resultará em um erro.

Para `writeConcern`, o Amazon DocumentDB suporta a maioria por padrão e um quorum de gravação é alcançado quando quatro cópias dos dados persistem em três AZs. Se `writeConcern` menor for especificado, o Amazon DocumentDB atualizará o valor `writeConcern` como maioria. Além disso, todas as gravações do Amazon DocumentDB são registradas no diário e o registro no diário não pode ser desativado.

Casos de uso

Nesta seção, abordaremos dois casos de uso para transações: várias declarações e várias cobranças.

Multi-Statement Transações

As transações do Amazon DocumentDB são de várias declarações, o que significa que você pode escrever uma transação que abranja várias declarações com uma confirmação ou reversão explícita. É possível agrupar `insert`, `update`, `delete`, e `findAndModify` ações como uma única operação atômica.

Um caso de uso comum para transações com vários extratos é uma transação de débito e crédito. Por exemplo: você deve dinheiro a um amigo por roupas. Portanto, você precisa debitar (sacar) $500 da sua conta e creditar $500 (depósito) na conta do seu amigo. Para realizar essa operação, você executa as operações de dívida e crédito em uma única transação para garantir a atomicidade. Isso evita cenários em que $500 sejam debitados de sua conta, mas não creditados na conta de seu amigo. Veja como seria esse 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 transações

Nossas transações também são de cobrança múltipla, o que significa que podem ser usadas para realizar várias operações em uma única transação e em várias cobranças. Isso fornece uma visão consistente dos dados e mantém a integridade dos dados. Quando você confirma os comandos de forma única<>, as transações são execuções de tudo ou nada, ou seja, todas elas serão bem-sucedidas ou todas falharão.

Aqui está um exemplo de transações com várias cobranças, usando o mesmo cenário e dados do exemplo para transações com vários extratos.

```
// *** 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
```

Exemplos de API de transação para API de retorno de chamada

A API de retorno de chamada só está disponível para drivers de 4.2 ou mais.

Javascript

O código a seguir demonstra como utilizar a API de transação do Amazon DocumentDB com 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

O código a seguir demonstra como utilizar a API de transação Amazon DocumentDB com 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#

O código a seguir demonstra como utilizar a API de transação do Amazon DocumentDB com 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

O código a seguir demonstra como utilizar a API de transação do Amazon DocumentDB com 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

O código a seguir demonstra como utilizar a API de transação do Amazon DocumentDB com 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

O código a seguir demonstra como utilizar a API de transação do Amazon DocumentDB com 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

O código a seguir demonstra como utilizar a API de transação do Amazon DocumentDB com 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

O código a seguir demonstra como utilizar a API de transação do Amazon DocumentDB com 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
```

Exemplos de API de transação para API principal

Javascript

O código a seguir demonstra como utilizar a API de transação do Amazon DocumentDB com 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#

O código a seguir demonstra como utilizar a API de transação do Amazon DocumentDB com 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

O código a seguir demonstra como utilizar a API de transação do Amazon DocumentDB com 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

O código a seguir demonstra como utilizar a API de transação do Amazon DocumentDB com 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

O código a seguir demonstra como utilizar a API de transação do Amazon DocumentDB com 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

O código a seguir demonstra como utilizar a API de transação do Amazon DocumentDB com 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

O código a seguir demonstra como utilizar a API de transação do Amazon DocumentDB com o 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

O código a seguir demonstra como utilizar a API de transação do Amazon DocumentDB com 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 compatíveis

Command	Compatível
`abortTransaction`	Sim
`commitTransaction`	Sim
`endSessions`	Sim
`killSession`	Sim
`killAllSession`	Sim
`killAllSessionsByPattern`	Não
`refreshSessions`	Não
`startSession`	Sim

Capacidades sem suporte

Métodos	Estágios ou comandos
`db.collection.aggregate()`	`$collStats` `$currentOp`

Métodos	Estágios ou comandos

	`$indexStats`
	
	`$listSessions`
	
	`$out`

`db.collection.count()`
 	
`db.collection.countDocuments()`
	`$where`
	
	`$near`
	
	`$nearSphere`

`db.collection.insert()`	`insert` não é compatível se não for executado em uma coleção existente. Esse método é compatível se for direcionado a uma coleção pré-existente.

Sessões

As sessões do MongoDB são uma estrutura usada para oferecer suporte a gravações repetitivas, consistência causal, transações e gerenciar operações em bancos de dados. Quando uma sessão é criada, um identificador lógico de sessão (lsid) é gerado pelo cliente e usado para marcar todas as operações dentro dessa sessão ao enviar comandos para o servidor.

O Amazon DocumentDB suporta o uso de sessões para permitir transações, mas não oferece suporte à consistência causal. A partir da versão 8.0.2 do mecanismo, o Amazon DocumentDB suporta gravações que podem ser repetidas fora das transações para operações de gravação nas quais cada gravação individual afeta no máximo um documento. Nas versões anteriores do mecanismo, não há suporte para gravações repetidas. Para obter mais informações, consulte Gravações que podem ser repetidas no Amazon DocumentDB.

Ao utilizar transações no Amazon DocumentDB, uma transação será iniciada de dentro de uma sessão usando `session.startTransaction()` a API e uma sessão oferece suporte a uma única transação por vez. Da mesma forma, as transações são concluídas usando as APIs commit (`session.commitTransaction()`) ou abort (`session.abortTransaction()`).

Consistência causal

A consistência causal garante que, em uma única sessão do cliente, o cliente observe a consistência de leitura após gravação, monoatômica, e as gravações seguirão as leituras reads/writes, e essas garantias se aplicam a todas as instâncias em um cluster, não apenas à primária. O Amazon DocumentDB não oferece suporte à consistência causal e a declaração a seguir resultará em um erro.

```
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()
```

É possível desativar a consistência causal em uma sessão. Observe que isso permitirá que você utilize a estrutura da sessão, mas não fornecerá garantias de consistência causal para leituras. Ao usar o Amazon DocumentDB, as leituras do primário serão consistentes leitura após gravação e as leituras das instâncias de réplica acabarão sendo consistentes. As transações são o principal caso de uso para a utilização de sessões.

```
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 }
```

Gravações repetíveis

Com gravações que podem ser repetidas, seu driver repete automaticamente uma operação de gravação uma vez se ocorrer um erro de rede ou se o driver não conseguir encontrar a primária. A partir da versão 8.0.2 do mecanismo, o Amazon DocumentDB oferece suporte a gravações que podem ser repetidas. Nas versões anteriores do mecanismo, as gravações que podem ser repetidas devem ser desativadas por meio da configuração `retryWrites=false` na cadeia de conexão. Para obter mais informações, consulte Gravações que podem ser repetidas no Amazon DocumentDB.

Erros de transação

Ao usar transações, há cenários que podem gerar um erro que indica que o número de uma transação não corresponde a nenhuma transação em andamento.

O erro pode ser gerado em pelo menos dois cenários diferentes:

- Após o tempo limite da transação de um minuto.
- Após a reinicialização de uma instância (devido a patches, recuperação de falhas etc.), é possível receber esse erro mesmo nos casos em que a transação foi confirmada com sucesso. Durante a reinicialização de uma instância, o banco de dados não consegue distinguir entre uma transação concluída com êxito e uma transação que foi abortada. Em outras palavras, o estado de conclusão da transação é ambíguo.

A melhor maneira de lidar com esse erro é tornar as atualizações transacionais idempotentes — por exemplo, usando o `$set` mutador em vez de uma operação. increment/decrement Consulte abaixo:

```
{ "ok" : 0,
"operationTime" : Timestamp(1603938167, 1),
"code" : 251,
"errmsg" : "Given transaction number 1 does not match any in-progress transactions."
}
```

Gravações que podem ser repetidas no Amazon DocumentDB

A partir da versão 8.0.2 do mecanismo, o Amazon DocumentDB oferece suporte a gravações que podem ser repetidas. Quando uma operação de gravação falha devido a um erro transitório de rede ou a uma eleição primária, o driver pode repetir automaticamente a operação exatamente uma vez.

O Amazon DocumentDB desduplica a gravação repetida para que a operação seja aplicada no máximo uma vez, preservando a idempotência.

As gravações que podem ser repetidas exigem um MongoDB-compatible driver que ofereça suporte ao protocolo de gravações que podem ser repetidas. A maioria dos MongoDB-compatible drivers atuais permite gravações repetidas por padrão (`retryWrites=true`na cadeia de conexão). Se você estiver atualizando para o mecanismo 8.0.2 de uma versão anterior, poderá remover `retryWrites=false` da cadeia de conexão para ativar esse recurso.

Requisitos

Para usar gravações que podem ser repetidas, você deve atender aos seguintes requisitos:

- Mecanismo Amazon DocumentDB versão 8.0.2 ou posterior.
- Um MongoDB-compatible driver que suporta gravações que podem ser repetidas. Para a versão mínima, consulte a documentação do seu driver.
- A cadeia de conexão deve incluir`retryWrites=true`, ou o driver deve usar como padrão gravações que podem ser repetidas (a maioria dos drivers atuais faz isso).

Operações compatíveis

As gravações que podem ser repetidas se aplicam às operações de gravação nas quais cada gravação individual afeta no máximo um documento. As operações em lote são qualificadas, pois cada gravação dentro do lote é desduplicada separadamente.

As seguintes operações de gravação podem ser repetidas:

- `insertOne`
- `insertMany`
- `updateOne`
- `deleteOne`
- `findOneAndUpdate`
- `findOneAndDelete`
- `findOneAndReplace`
- `bulkWrite`(quando composto por `insertOneupdateOne`,`deleteOne`, ou `replaceOne` operações)

Para `insertMany` e`bulkWrite`, cada documento no lote é desduplicado individualmente. Se uma nova tentativa for necessária, somente os documentos que ainda não foram aplicados serão inseridos.

> **Note**
>
> `commitTransaction`e também `abortTransaction` podem ser repetidos, como comandos de controle de transações e não como gravações. As gravações individuais dentro de uma transação não podem ser repetidas. Para obter mais informações, consulte Limitações.

Limitações

As seguintes limitações se aplicam às gravações que podem ser repetidas no Amazon DocumentDB:

- `updateMany`e não `deleteMany` podem ser repetidas.
- As gravações em transações com várias declarações não podem ser repetidas. As operações de confirmação e cancelamento da transação podem ser repetidas separadamente.
- Os documentos devem incluir um `_id` campo para inserções que podem ser repetidas.
- As gravações que podem ser repetidas estão disponíveis somente na versão 8.0.2 e posterior do mecanismo. Nas versões anteriores do mecanismo, `retryWrites=false` defina sua string de conexão para evitar erros.
- As operações de gravação não podem ser repetidas em clusters que usam o planejador de consultas versão 1.0.
- Para `findOneAndReplace` operações de `findOneAndUpdate`, `findOneAndDelete`, e que retornam documentos grandes, as gravações que podem ser repetidas podem aumentar a latência de gravação, pois todo o documento resultante é armazenado em cache para desduplicação. Se o desempenho de gravação for essencial e os documentos devolvidos forem grandes, considere a configuração `retryWrites=false` para essas cargas de trabalho.

Habilitando gravações que podem ser repetidas

Para habilitar gravações que podem ser repetidas, inclua `retryWrites=true` em sua string de conexão:

```
mongodb://<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=true
```

Se o padrão do seu driver for`retryWrites=true`, você poderá remover qualquer conteúdo explícito `retryWrites=false` da sua cadeia de conexão.

Para ver o passo a passo completo da cadeia de conexão, consulte. Conectar-se de forma programática ao Amazon DocumentDB

Como funcionam as gravações que podem ser repetidas

Quando seu aplicativo envia uma gravação com`retryWrites=true`, o driver anexa um ID de sessão lógico (`lsid`) e um número de transação (`txnNumber`) ao comando de gravação. O Amazon DocumentDB usa esses identificadores para desduplicar gravações:

- Na primeira tentativa, o Amazon DocumentDB executa a gravação e armazena o resultado em cache.
- Se o driver tentar novamente a mesma gravação (a mesma `lsid` e`txnNumber`), o Amazon DocumentDB retornará o resultado armazenado em cache sem executar novamente a gravação.
- O Amazon DocumentDB honra novas tentativas por 60 minutos após a gravação original. Os motoristas tentam novamente imediatamente, então essa janela é muito maior do que o necessário para o motorista.

Se uma nova tentativa chegar mais de 60 minutos após a gravação original, o Amazon DocumentDB não terá mais um resultado armazenado em cache e executará a gravação como uma nova operação.

Tratamento de erros

Os seguintes erros são específicos para gravações que podem ser repetidas no Amazon DocumentDB:

Código de erro	Name (Nome)	Description
225	TransactionTooOld	Um novo número de transação para a mesma sessão e declaração já foi confirmado, então essa nova tentativa é muito antiga para ser aplicada.
301	Gravações que podem ser repetidas não são suportadas	A gravação incluiu campos de gravação que podem ser repetidos, mas a versão do mecanismo não suporta gravações que podem ser repetidas ou a operação não pode ser repetida. `retryWrites=false` Definido ao conectar a versões do motor anteriores à 8.0.2.

Migrando de RetryWrites=false

Se você estiver atualizando para a versão 8.0.2 do mecanismo e atualmente usa `retryWrites=false` em suas cadeias de conexão:

1. Atualize seu cluster para a versão 8.0.2 ou posterior do mecanismo.
2. Remova as `retryWrites=false` cadeias de conexão do seu aplicativo ou altere-as para`retryWrites=true`.
3. Nenhuma alteração no código do aplicativo é necessária. O motorista processa as novas tentativas automaticamente.

Usando agrupamento no Amazon DocumentDB 8.0

O Amazon DocumentDB 8.0 agora oferece suporte ao agrupamento. O agrupamento permite que você configure regras específicas do idioma para comparação de strings. Com o agrupamento, você pode especificar regras para comparações com distinção entre maiúsculas e minúsculas ou especificar uma localidade de idioma. O agrupamento pode ser configurado no nível da coleção ou do índice no DocumentDB 8.0. Quando você usa o agrupamento no DocumentDB, internamente, um documento de agrupamento é criado com os seguintes parâmetros.

```
{
locale: string,
caseLevel: boolean,
caseFirst: string,
strength: int,
numericOrdering: boolean,
alternate: string,
maxVariable: string,
backwards: boolean,
normalization: boolean
}
```

Limitações

O agrupamento tem as seguintes limitações no Amazon DocumentDB:

- O agrupamento é compatível com o planner v3 disponível no Amazon DocumentDB 8.0. Mudar para plannerv2 ou plannerv1 pode causar comportamentos inconsistentes, incluindo erros de “Índice não encontrado”.
- Devido às diferenças inerentes à biblioteca, se uma coleção mongodb com agrupamento for exportada, você precisará atualizar os arquivos metadata.bson e alterar sua versão de 57.1 para 60.2 antes da migração.
- Em casos raros, suas configurações de agrupamento podem violar o limite interno na contagem de caracteres, causando o seguinte erro. “Erro: o documento de agrupamento tem atributos não padrão mais do que os suportados. Por favor, reduza o número de opções.” Nesse caso, tente reduzir as opções fornecidas no documento de agrupamento ou, alternativamente, tente usar valores padrão sempre que possível.

Usando visualizações no Amazon DocumentDB 8.0

- O Amazon DocumentDB 8.0 agora oferece suporte a visualizações. As visualizações funcionam como coleções virtuais que apresentam dados com base em operações de agregação especificadas. Ao criar uma exibição, você define uma consulta que transforma dados de uma ou mais coleções de origem. O Amazon DocumentDB 8.0 executa essa consulta sempre que a visualização é acessada, sem consumir recursos adicionais de armazenamento. Diferentemente das coleções padrão, as visualizações no Amazon DocumentDB 8.0 não armazenam documentos em disco, o que as torna uma solução eficiente para apresentar dados transformados ou filtrados

aos aplicativos. Para criar uma visualização no Amazon DocumentDB, você pode usar o comando CreateView ou o método auxiliar db.CreateView ():

```
db.createView("viewName","sourceCollection",
[
    { $match: { status: "active" } },
    { $project: { _id: 1, name: 1, email: 1 } }
]
)
```

Isso cria uma visualização chamada “ViewName” com base em “SourceCollection” que inclui apenas documentos e projetos ativos, somente os campos _id, nome e e-mail. As visualizações no Amazon DocumentDB são somente para leitura. As operações de gravação nas visualizações retornarão um erro. Para um desempenho ideal com grandes conjuntos de dados, você pode estruturar seus pipelines de visualização para maximizar a eficiência. Para pipelines de agregação complexos, é recomendável usar o estágio $match como o primeiro estágio ou no início do pipeline para reduzir o número de documentos que os estágios subsequentes precisam processar, melhorando assim o desempenho da consulta.

Práticas recomendadas

Algumas das melhores práticas a serem seguidas com as visualizações estão listadas abaixo.

- Filtrar antecipadamente: use os estágios $match no início dos pipelines de visualização para reduzir a quantidade de dados processados.
- Evite agregações complexas: para visualizações acessadas com frequência com agregações complexas, considere criar uma coleção separada com resultados pré-computados que são atualizados periodicamente.
- Planejamento de indexação: garanta que os campos usados nos pipelines de visualização, especialmente nas operações $match e $sort, estejam devidamente indexados na coleção de origem.
- Otimização de consultas: use o comando explain para entender como suas consultas de visualização são executadas e otimizá-las adequadamente.
- Alternativas para visualizações: dadas as diferenças funcionais entre as visualizações do Amazon DocumentDB e do MongoDB, considere usar coleções regulares com atualizações programadas como alternativa às visualizações quando encontrar limitações.

Compatibilidade do operador agregador

O Amazon DocumentDB oferece suporte a muitos operadores de agregação em definições de visualização, ao mesmo tempo em que continua expandindo a compatibilidade. Ao usar visualizações, concentre-se nesses operadores compatíveis:

- $match para filtrar documentos
- $project para seleção e transformação de campo
- $addFields para adicionar campos computados
- $sort para resultados de pedidos
- $limit e $skip para paginação

Alguns operadores especializados, como $currentOp, $replaceRoot e $geoNear, atualmente trabalham em consultas de agregação direta em vez de definições de visualização.

Aproveitando índices e visualizações

As visualizações no Amazon DocumentDB 8.0 usam os índices da coleção subjacente. Como resultado, você não pode criar, eliminar ou reconstruir índices diretamente em uma exibição. No entanto, índices bem projetados na coleção de origem podem melhorar significativamente o desempenho da consulta de visualização. Abaixo estão algumas etapas para otimizar o desempenho da consulta nas visualizações:

- Certifique-se de que existam índices apropriados nos campos da coleção de origem usados no pipeline da visualização, especialmente nas operações $match e $sort
- Use o método explain () para analisar os planos de execução de consultas e verificar o uso do índice. E.g., `db.viewName.find({...}).explain()`

Inteligência artificial generativa do Amazon DocumentDB

O Amazon DocumentDB oferece recursos para permitir que modelos de machine learning (ML) e inteligência artificial (IA) generativa funcionem com dados armazenados no Amazon DocumentDB em tempo real. Os clientes não precisam mais perder tempo gerenciando uma infraestrutura separada, escrevendo código para se conectar a outro serviço e duplicando dados do banco de dados primário.

Para obter mais informações sobre inteligência artificial e como AWS atender às suas necessidades de IA, consulte este artigo What-is "".

Tópicos

- No-code aprendizado de máquina com o Amazon SageMaker AI Canvas
- Pesquisa vetorial para Amazon DocumentDB

No-code aprendizado de máquina com o Amazon SageMaker AI Canvas

O Amazon SageMaker AI Canvas permite que você crie seus próprios AI/ML modelos sem precisar escrever uma única linha de código. É possível criar modelos de ML para casos de uso comuns, como regressão e previsão, e pode acessar e avaliar modelos de base (FMs) do Amazon Bedrock. Você também pode acessar FMs públicas da Amazon SageMaker AI JumpStart para geração de conteúdo, extração de texto e resumo de texto para oferecer suporte a soluções generativas de IA.

Como criar modelos de ML sem código com o SageMaker AI Canvas

O Amazon DocumentDB agora se integra ao Amazon SageMaker AI Canvas para permitir o aprendizado de máquina (ML) sem código com dados armazenados no Amazon DocumentDB. Agora você pode criar modelos de ML para necessidades de regressão e previsão e usar modelos de base para resumir e gerar conteúdo usando dados armazenados no Amazon DocumentDB sem escrever uma única linha de código.

SageMaker O AI Canvas fornece uma interface visual que permite aos clientes do Amazon DocumentDB gerar previsões sem precisar de nenhum AI/ML conhecimento ou escrever uma única linha de código. Agora, os clientes podem iniciar o espaço de trabalho do SageMaker AI Canvas a partir do Console de gerenciamento da AWS, importar e unir dados do Amazon DocumentDB para preparação de dados e treinamento de modelos. Os dados no Amazon DocumentDB agora podem ser usados no SageMaker AI Canvas para criar e ampliar modelos para prever a rotatividade de clientes, detectar fraudes, prever falhas de manutenção, prever métricas de negócios e gerar conteúdo. Agora, os clientes podem publicar e compartilhar ML-driven ideias entre as equipes usando a integração nativa do SageMaker AI Canvas com o Quick. Os pipelines de ingestão de dados no SageMaker AI Canvas são executados em instâncias secundárias do Amazon DocumentDB por padrão, garantindo que o desempenho dos aplicativos e das cargas de trabalho de ingestão do SageMaker AI Canvas não seja prejudicado.

Os clientes do Amazon DocumentDB podem começar a usar o SageMaker AI Canvas navegando até a nova página do Amazon DocumentDB No-Code ML Console e conectando-se a espaços de trabalho do SageMaker AI Canvas novos ou disponíveis.

Configurando o domínio e o SageMaker perfil de usuário da IA

Você pode se conectar aos clusters do Amazon DocumentDB a partir de domínios de SageMaker IA que estão sendo executados no modo somente VPC. Ao lançar um domínio de SageMaker IA na sua VPC, você pode controlar o fluxo de dados dos seus ambientes SageMaker AI Studio e Canvas. Isso permite que você restrinja o acesso à Internet, monitore e inspecione o tráfego usando recursos padrão AWS de rede e segurança e se conecte a outros AWS recursos por meio de endpoints VPC. Consulte os conceitos básicos do Amazon SageMaker AI Canvas e configure o Amazon SageMaker AI Canvas em uma VPC sem acesso à Internet localizada no Guia do desenvolvedor de SageMaker IA da Amazon para criar seu domínio de SageMaker IA e conectar-se ao seu cluster Amazon DocumentDB.

Configuração de permissões de acesso do IAM para Amazon DocumentDB e AI Canvas SageMaker

Um usuário do Amazon DocumentDB com `AmazonDocDBConsoleFullAccess` vinculado à sua função e identidade associada pode acessar o Console de gerenciamento da AWS. Adicione as seguintes ações à função ou identidade mencionada anteriormente para fornecer acesso ao aprendizado de máquina sem código com o Amazon SageMaker AI Canvas.

```
"sagemaker:CreatePresignedDomainUrl",
"sagemaker:DescribeDomain",
"sagemaker:ListDomains",
"sagemaker:ListUserProfiles"
```

Criação de usuários e funções de banco de dados para o SageMaker AI Canvas

É possível restringir o acesso às ações que os usuários podem executar em bancos de dados usando o controle de acesso baseado em função (RBAC) no Amazon DocumentDB. O RBAC funciona concedendo uma ou mais funções a um usuário. Estas funções determinam as operações que um usuário pode realizar nos recursos do banco de dados.

Como usuário do Canvas, você se conecta a um banco de dados do Amazon DocumentDB com credenciais de nome de usuário e senha. Você pode criar um banco de dados user/role para

um usuário do Canvas que tenha acesso de leitura aos bancos de dados específicos usando a funcionalidade RBAC do Amazon DocumentDB.

Por exemplo, use a operação `createUser`:

```
db.createUser({
user: "canvas_user",
pwd: "<insert-password>",
roles: [{role: "read", db: "sample-database-1"}]
})
```

Isso cria um `canvas_user` que tem permissões de leitura no banco de dados `sample-database-1`. Seus analistas do Canvas podem usar essa credencial para acessar dados no cluster do Amazon DocumentDB. Consulte Acesso ao banco de dados usando o controle de Role-Based acesso para saber mais.

Regiões disponíveis

A integração sem código está disponível nas regiões em que o Amazon DocumentDB e o Amazon SageMaker AI Canvas são compatíveis. As regiões incluem:

- us-east-1 (N. Virginia)
- us-east-2 (Ohio)
- us-west-2 (Oregon)
- ap-northeast-1 (Tóquio)
- ap-northeast-2 (Seul)
- ap-south-1 (Mumbai)
- ap-southeast-1 (Singapura)
- ap-southeast-2 (Sydney)
- eu-central-1 (Frankfurt)
- eu-west-1 (Irlanda)

Consulte o Amazon SageMaker AI Canvas no Guia do desenvolvedor de SageMaker IA da Amazon para ver a disponibilidade mais recente na região.

Pesquisa vetorial para Amazon DocumentDB

A pesquisa vetorial é um método usado em machine learning para encontrar pontos de dados semelhantes a um determinado ponto de dados comparando suas representações vetoriais usando métricas de distância ou similaridade. Quanto mais próximos os dois vetores estiverem no espaço vetorial, mais semelhantes serão considerados os itens subjacentes. Essa técnica ajuda a capturar o significado semântico dos dados. Essa abordagem é útil em várias aplicações, como sistemas de recomendação, processamento de linguagem natural e reconhecimento de imagem.

A pesquisa vetorial do Amazon DocumentDB combina a flexibilidade e a rica capacidade de consulta de um banco de dados de JSON-based documentos com o poder da pesquisa vetorial. Se você quiser usar seus dados existentes do Amazon DocumentDB ou uma estrutura flexível de dados de documentos para criar casos de uso de machine learning e IA generativa, como experiência de pesquisa semântica, recomendação de produtos, personalização, bots de chat, detecção de fraudes e detecção de anomalias, a pesquisa vetorial do Amazon DocumentDB é a escolha ideal para você. A pesquisa vetorial está disponível nos clusters baseados em instâncias do Amazon DocumentDB 5.0.

Tópicos

- Inserir vetores
- Criar um índice vetorial
- Obter uma definição de índice
- Consultar vetores
- Atributos e limitações
- Práticas recomendadas

Inserir vetores

Para inserir vetores em seu banco de dados do Amazon DocumentDB, você pode usar os métodos de inserção existentes:

Exemplo

No exemplo a seguir, uma coleção de cinco documentos em um banco de dados de teste é criada. Cada documento inclui dois campos: o nome do produto e a incorporação vetorial correspondente.

```
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]}
]);
```

Criar um índice vetorial

O Amazon DocumentDB é compatível com os métodos de indexação Hierarchical Navigable Small World (HNSW) e Inverted File with Flat Compression (IVFFlat). Um índice IVFflat segmenta vetores em listas e, posteriormente, pesquisa um subconjunto selecionado dessas listas que estão mais próximas do vetor de consulta. Por outro lado, um índice HNSW organiza os dados vetoriais em um gráfico de várias camadas. Embora o HNSW tenha tempos de construção mais lentos em comparação com o IVFFlat, ele oferece melhor desempenho e recuperação de consultas. Ao contrário do IVFFlat, o HNSW não envolve nenhuma etapa de treinamento, permitindo que o índice seja gerado sem qualquer carga inicial de dados. Para a maioria dos casos de uso, use o tipo de índice HNSW para pesquisa vetorial.

Se você não criar um índice vetorial, o Amazon DocumentDB executará uma busca exata do vizinho mais próximo, garantindo uma recuperação perfeita. No entanto, em cenários de produção, a velocidade é crucial. Use índices vetoriais, que podem trocar algum recall por maior velocidade. É importante observar que adicionar um índice vetorial pode levar a resultados de consultas diferentes.

Modelos

É possível usar os seguintes modelos `createIndex` ou `runCommand` para criar um índice vetorial em um campo vetorial:

Using createIndex

Em certos drivers, como mongosh e Java, o uso dos `vectorOptions` parâmetros in `createIndex` pode resultar em um erro. Nesses casos, use`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

Em certos drivers, como mongosh e Java, o uso dos `vectorOptions` parâmetros in `createIndex` pode resultar em um erro. Nesses casos, use`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"
      }]
  }
);
```

Parâmetro	Requisito	Tipo de dados	Description	Valor(es)
`name`	optional	string	Especifica o nome do índice.	Alfanumérico
`type`	optional		Especifica o tipo de índice.	Com suporte: hnsw ou ivfflat Padrão: HNSW (patch do

Parâmetro	Requisito	Tipo de dados	Description	Valor(es)
				mecanismo 3.0.4574 em diante)
`dimensions`	obrigatório	integer	Especifica o número de dimensões nos dados vetoriais.	Máximo de 2.000 dimensões.
`similarity`	obrigatório	string	Especifica a métrica de distância usada para o cálculo da similaridade.	• **`euclidean`** • **`cosine`** • **`dotProduct`**
`lists`	obrigatório para IVFFlat	integer	Especifica o número de clusters que o índice IVFFlat usa para agrupar os dados vetoriais. A configuração recomendada é o número de documents /1000 para até 1 milhão de documentos e `sqrt(# of documents )` para mais de 1 milhão de documentos.	Minimum (Mínimo): 1 Máximo: consulte a tabela de listas por tipo de instância no Atributos e limitações abaixo.

Parâmetro	Requisito	Tipo de dados	Description	Valor(es)
`m`	optional	integer	Especifica o número máximo de conexões para um índice HNSW	Padrão: 16 Intervalo: [2, 100]
`efConstruction`	optional	integer	Especifica o tamanho da lista dinâmica de candidatos para construir o gráfico para o índice HNSW. `efConstruction` deve ser maior que ou igual a (2 * m)	Padrão: 64 Intervalo [4, 1000]

É importante que você defina adequadamente o valor dos subparâmetros, como `lists` para IVFFlat `m` e `efConstruction` HNSW, pois isso afetará o tempo de construção e o accuracy/recall desempenho de sua pesquisa. Um valor de lista maior aumenta a velocidade da consulta, pois reduz o número de vetores em cada lista, resultando em regiões menores. No entanto, um tamanho de região menor pode levar a mais erros de recall, ocasionando menor precisão. Para o HNSW, aumentar o valor de `m` e `efConstruction` aumenta a precisão, mas também aumenta o tempo e o tamanho da construção do índice. Veja os exemplos a seguir:

Exemplos

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
    }
  }
)
```

Obter uma definição de índice

É possível visualizar os detalhes dos seus índices, incluindo índices vetoriais, usando o comando `getIndexes`:

Exemplo

```
db.collection.getIndexes()
```

Exemplo de saída

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

Consultar vetores

O Amazon DocumentDB oferece suporte a dois operadores de pesquisa vetorial para consultar vetores:

Operador de busca vetorial clássico

Use o modelo a seguir para consultar um vetor:

```
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]
      }
    }
  }
]);
```

Parâmetro	Requisito	Tipo	Description	Valor(es)
vectorSearch	obrigatório	operador	Usado dentro do comando $search para consultar os vetores.	
vector	obrigatório	array	Indica o vetor de consulta que será usado para encontrar vetores semelhantes.	
path	obrigatório	string	Define o nome do campo vetorial.	
k	obrigatório	integer	Especifica o número máximo de resultados que devem ser retornados da pesquisa.	
similarity	obrigatório	string	Especifica a métrica de distância usada para o cálculo da similaridade.	• **euclidean** • **cosine** • **dotProduct**
probes	optional	integer	O número de clusters que você deseja que a pesquisa vetorial inspecione. Um	Padrão: 1

Parâmetro	Requisito	Tipo	Description	Valor(es)
			valor mais alto proporciona um melhor recall em detrimento da velocidade. Ele pode ser definido como o número de listas para a pesquisa exata do vizinho mais próximo (nesse momento, o planejador não usará o índice). A configuração recomendada para iniciar o ajuste é `sqrt(# of lists)`.	
`efSearch`	optional	integer	Especifica o tamanho da lista dinâmica de candidatos que o índice HNSW usa durante a pesquisa. Um valor mais alto de `efSearch` fornece melhor recuperação em detrimento da velocidade.	Padrão: 60 Intervalo: [1, 1000].

É importante ajustar o valor de `efSearch` (HNSW) ou `probes` (IVFFlat) para obter a performance e a precisão desejadas. Veja os exemplos de operações a seguir:

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
      }
    }
  }
]);
```

Exemplo de saída

A saída dessa operação é semelhante à seguinte:

```
{ "_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 (disponível no Amazon DocumentDB 8.0 em diante)

Use o modelo a seguir para consultar um vetor:

```
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],
  }
}])
```

Atributos e limitações

Compatibilidade da versão

- A pesquisa vetorial do Amazon DocumentDB só está disponível em clusters baseados em instâncias do Amazon DocumentDB 5.0+.

Vetores

- O Amazon DocumentDB pode indexar vetores de até 2.000 dimensões. No entanto, até 16.000 dimensões podem ser armazenadas sem um índice.

Índices

- Para a criação do índice IVFFlat, a configuração recomendada para o parâmetro de listas é o número de documents/1000 até 1 milhão de documentos e `sqrt(# of documents)` para mais de 1 milhão de documentos. Devido ao limite de memória de trabalho, o Amazon DocumentDB suporta um determinado valor máximo do parâmetro de listas, dependendo do número de dimensões. Para sua referência, a tabela a seguir fornece os valores máximos do parâmetro de listas para vetores de 500, 1000 e 2.000 dimensões:

Tipo de instância	Listas com 500 dimensões	Listas com 1000 dimensões	Listas com 2000 dimensões
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
r5,24xl	19.585	19.335	18.845

- Nenhuma outra opção de índice, como `compound`, `sparse` ou `partial`, é aceita para índices vetoriais.
- A criação de índice paralelo não é compatível com o índice HNSW no Amazon DocumentDB 5.0.

Consulta vetorial

- Para consultas de pesquisa vetorial, é importante ajustar os parâmetros, como `probes` ou `efSearch`, para obter os melhores resultados. Quanto maior o valor do parâmetro `probes` ou `efSearch`, maior o recall e menor a velocidade. A configuração recomendada para iniciar o ajuste fino do parâmetro das sondas é `sqrt(# of lists)`.

Práticas recomendadas

Conheça as práticas recomendadas para trabalhar com clusters elásticos do Amazon DocumentDB. Essa seção é continuamente atualizada conforme novas melhores práticas são identificadas.

- A criação do índice IVFFlat (Inverted File with Flat Compression) envolve agrupar e organizar os pontos de dados com base nas semelhanças. Para que um índice seja mais eficaz, carregue alguns dados antes de criar o índice.
- Para consultas de pesquisa vetorial, é importante ajustar os parâmetros, como `probes` ou `efSearch`, para obter os melhores resultados. Quanto maior o valor do parâmetro `probes` ou `efSearch`, maior é o recall e menor é a velocidade. A configuração recomendada para iniciar o ajuste fino do parâmetro `probes` é `sqrt(lists)`.

Recursos

- A pesquisa vetorial do Amazon DocumentDB agora está disponível ao público em geral no blog AWS
- Exemplo de código de pesquisa semântica
- Exemplos de código de pesquisa vetorial do Amazon DocumentDB

Utilizar AWS Lambda com fluxos de mudança

O Amazon DocumentDB é integrado AWS Lambda para que você possa usar as funções do Lambda para processar registros em um fluxo de alterações. O mapeamento da fonte de eventos do Lambda é um recurso que pode ser usado para invocar funções do Lambda para processar eventos do Amazon DocumentDB que não invocam diretamente o Lambda. Com o fluxo de alterações do Amazon DocumentDB como origem do evento, é possível criar aplicações orientadas por eventos que respondam às mudanças em seus dados. Por exemplo, é possível usar as funções do Lambda para processar novos documentos, rastrear atualizações de documentos existentes ou registrar documentos excluídos.

É possível configurar um mapeamento da origem do evento para enviar registros de seu fluxo de alterações no Amazon DocumentDB para uma função do Lambda. Os eventos podem ser enviados um por vez ou agrupados para melhorar a eficiência, e serão processados em ordem. É possível configurar o comportamento de lote do mapeamento da origem do evento com base na duração de um intervalo específico (0 a 300 segundos) ou na contagem de registros em lote (limite máximo de 10.000 registros). É possível criar vários mapeamentos de origem de evento para processar os mesmos dados com várias funções do Lambda ou processar itens de vários fluxos com uma única função.

Se a sua função retornar um erro, o Lambda tentará executar novamente o lote até que o processamento seja bem-sucedido ou os dados expirem. Caso os eventos no fluxo de alterações tenham expirado, o Lambda desabilitará o mapeamento da origem do evento. Nesse caso, é possível criar um novo mapeamento da origem do evento e configurá-lo com uma posição inicial de sua escolha. Mapeamentos de fontes de eventos do Lambda processam eventos pelo menos uma vez devido à natureza distribuída de seus agentes de sondagem. Como resultado, sua função do Lambda pode receber eventos duplicados em situações raras. Siga as melhores práticas para trabalhar com AWS Lambda funções e criar funções idempotentes para evitar problemas relacionados a eventos duplicados. Para obter mais informações, consulte Usando AWS Lambda console com o Amazon DocumentDB no Guia do AWS Lambda desenvolvedor.

Como práticas recomendadas de performance, a função do Lambda precisa ser de curta duração. Para evitar a introdução de atrasos de processamento desnecessários, ela também não deve executar uma lógica complexa. Para um fluxo de alta velocidade em particular, é melhor acionar fluxos de trabalho assíncronos de função de etapa de pós-processamento do que Lambdas síncronos de longa execução. Para obter mais informações sobre AWS Lambda, consulte o Guia do AWS Lambda desenvolvedor.

Limitações

A seguir estão as limitações a serem consideradas ao trabalhar com o Amazon DocumentDB e AWS Lambda:

- AWS Lambda atualmente é suportado somente no Amazon DocumentDB 4.0 e 5.0.
- AWS Lambda atualmente não é compatível com clusters elásticos ou clusters globais.
- AWS Lambda os tamanhos da carga útil não podem exceder 6 MB. Para obter mais informações sobre tamanhos de lote do Lambda, consulte “Comportamento de lotes” na seção Mapeamentos da origem do evento do Lambda no Guia do desenvolvedor do AWS Lambda .

Compatibilidade do Amazon DocumentDB com MongoDB

O Amazon DocumentDB oferece suporte à compatibilidade com o MongoDB, incluindo MongoDB 4.0, MongoDB 5.0 e MongoDB 8.0. A compatibilidade com o MongoDB significa que a grande maioria das aplicações, drivers e ferramentas que você já usa atualmente com seus bancos de dados MongoDB pode ser usada com o Amazon DocumentDB com pouca ou nenhuma alteração. Esta seção descreve tudo o que você precisa saber sobre a compatibilidade do Amazon DocumentDB com o MongoDB, incluindo novos recursos e recursos, conceitos básicos, caminhos de migração e diferenças funcionais.

Tópicos

- Compatibilidade com MongoDB 8.0
- Compatibilidade com MongoDB 5.0
- Compatibilidade do MongoDB 4.0
- Diferenças funcionais: Amazon DocumentDB e MongoDB
- APIs, operações e tipos de dados do MongoDB compatíveis no Amazon DocumentDB

Compatibilidade com MongoDB 8.0

Tópicos

- O que há de novo no Amazon DocumentDB 8.0
- Comece a usar o Amazon DocumentDB 8.0
- Atualize ou migre para o Amazon DocumentDB 5.0 ou 8.0
- Diferenças funcionais

O que há de novo no Amazon DocumentDB 8.0

O Amazon DocumentDB 8.0 melhora o desempenho das consultas em até 7 vezes e a taxa de compressão em até 5 vezes, permitindo que você crie aplicativos de alto desempenho a um custo menor. O Amazon DocumentDB 8.0 tem total compatibilidade de protocolo de conexão com o MongoDB 8.0. O resumo abaixo apresenta alguns dos principais recursos que foram introduzidos no Amazon DocumentDB 8.0. Para ver uma lista completa dos novos recursos, consulte Notas da versão.

- Fornece compatibilidade com o MongoDB 8.0 adicionando suporte aos drivers da API MongoDB 8.0. Também oferece suporte a aplicativos criados usando as versões 6.0 e 7.0 da API MongoDB.
- O novo planejador de consultas no Amazon DocumentDB 8.0 estende as melhorias de desempenho aos operadores do estágio de agregação, além de oferecer suporte a otimizações de pipeline de agregação e comandos distintos.
- O Amazon DocumentDB 8.0 suporta agrupamento e visualizações.
- Oferece 6 novos estágios de agregação: $replaceWith, $vectorSearch, $merge, $set, $unset, $bucket e 3 novos operadores de agregação $pow, $rand, $dateTrunc.
- Uma nova versão do índice de texto: o índice de texto v2 no Amazon DocumentDB 8.0 introduz tokens adicionais, aprimorando os recursos de pesquisa de texto.
- Por meio da criação paralela de índices vetoriais, o Amazon DocumentDB 8.0 reduz o tempo de criação do índice em até 30 vezes.
- Limite de armazenamento aumentado para 256 TiB.

Comece a usar o Amazon DocumentDB 8.0

Para começar a usar o Amazon DocumentDB 8.0, consulte o. Conceitos básicos do Amazon DocumentDB Você pode criar um novo cluster Amazon DocumentDB 8.0 usando o Console de gerenciamento da AWS ou o AWS SDK AWS CLI, ou. CloudFormation Ao se conectar ao Amazon DocumentDB, é necessário que você use um driver ou utilitário do MongoDB compatível com o MongoDB 5.0 ou superior.

> **Note**
>
> Ao usar o AWS SDK,, ou AWS CLI CloudFormation, a versão do mecanismo será predefinida como 5.0.0. Você deve especificar explicitamente o parâmetro `engineVersion = 8.0.0` para criar um novo cluster Amazon DocumentDB 8.0 ou `engineVersion = 4.0.0` criar um novo cluster Amazon DocumentDB 4.0 ou `engineVersion = 3.6.0` criar um novo cluster Amazon DocumentDB 3.6. Para um determinado cluster do Amazon DocumentDB, você pode determinar a versão do cluster usando o console de gerenciamento do Amazon DocumentDB AWS CLI para chamar `describe-db-clusters` ou usar o console de gerenciamento do Amazon DocumentDB para visualizar o número da versão do mecanismo de um cluster específico.

O Amazon DocumentDB 5.0 e 8.0 oferece suporte aos processadores `r8g` Graviton4 e Graviton2 do Amazon EC2, como tipos de `t4.medium` instância para seus clusters, `r6g` e está disponível em todas as versões suportadas (consulte). Regiões da AWS Instâncias Para obter mais informações sobre preços, consulte Preços do Amazon DocumentDB (compatível com MongoDB).

Atualize ou migre para o Amazon DocumentDB 5.0 ou 8.0

Você pode migrar do MongoDB 3.6 ou MongoDB 4.0 para o Amazon DocumentDB 5.0 ou Amazon DocumentDB 8.0 usando os utilitários AWS DMS or, como mongodump, mongorestore, mongoimport e mongoexport. Para obter instruções sobre migração, consulteAtualizando seu cluster Amazon DocumentDB usando AWS Database Migration Service.

Diferenças funcionais

Diferenças funcionais entre o Amazon DocumentDB 5.0 e 8.0

Com o lançamento do Amazon DocumentDB 8.0, existem diferenças funcionais entre o Amazon DocumentDB 5.0 e o Amazon DocumentDB 8.0:

- O Planner v1 é o planejador de consultas padrão no Amazon DocumentDB 5.0, enquanto o Planner v3, de maior desempenho, é o padrão no Amazon DocumentDB 8.0.
- Os novos recursos do Amazon DocumentDB 8.0, incluindo visualizações, agrupamento e operadores como $merge, são compatíveis somente com o Planner v3.
- A compactação é ativada por padrão no Amazon DocumentDB 8.0 e configurada para usar o algoritmo Zstandard. Além disso, no Amazon DocumentDB 8.0, 'enabled' não é mais uma opção válida; você pode selecionar entre Zstd, LZ4 e nenhum.
- O Amazon DocumentDB 8.0 só oferece suporte ao TLS 1.2 e superior para conexões de banco de dados.

> **Important**
>
> O Amazon DocumentDB 8.0 exige TLS 1.2 ou superior. O TLS 1.0 e o TLS 1.1 não são mais suportados.

Compatibilidade com MongoDB 5.0

Tópicos

- Novidades do Amazon DocumentDB 5.0
- O que há de novo no Amazon DocumentDB 5.0.1
- Conceitos básicos do Amazon DocumentDB 5.0
- Atualize ou migre para o Amazon DocumentDB 5.0
- Diferenças funcionais

Novidades do Amazon DocumentDB 5.0

O Amazon DocumentDB 5.0 introduz novos recursos e capacidades que incluem limites de armazenamento e criptografia em nível de campo do lado do cliente. O resumo abaixo apresenta alguns dos principais recursos que foram introduzidos no Amazon DocumentDB 5.0. Para ver uma lista completa dos novos recursos, consulte Notas da versão.

- Aumento do limite de armazenamento para 128 TiB para todos os clusters do Amazon DocumentDB baseados em instâncias e clusters elásticos baseados em fragmentos.
- Introdução do Amazon DocumentDB 5.0 Engine (versão 3.0.775)
  - Suporte para drivers de API do MongoDB 5.0
  - Suporte para criptografia em nível de Client-side campo (FLE). Agora você pode criptografar campos no lado do cliente antes de gravar os dados no cluster do Amazon DocumentDB. Para obter mais informações, consulte Client-side criptografia em nível de campo.
  - Novos operadores de agregação: `$dateAdd`, `$dateSubtract`
  - Suportes para índices com operador `$elemMatch`. Como resultado, as consultas `$elemMatch` resultarão em varreduras de índice.

O Amazon DocumentDB não oferece suporte a todos os atributos do MongoDB 5.0. Quando criamos o Amazon DocumentDB 5.0, trabalhamos de trás para frente com base nos atributos e nas capacidades que nossos clientes mais pediram que criássemos. Continuaremos adicionando recursos adicionais do MongoDB 5.0 com base no feedback dos clientes. Para ver a lista mais recente de APIs compatíveis, consulteAPIs, operações e tipos de dados do MongoDB compatíveis no Amazon DocumentDB.

O que há de novo no Amazon DocumentDB 5.0.1

O Amazon DocumentDB 5.0.1 é a primeira versão secundária da versão principal do Amazon DocumentDB 5.0. As versões secundárias incluem melhorias, como novos recursos, correções de segurança e correções de erros. Você pode atualizar para 5.0.1 de 5.0.0 usando uma atualização de versão secundária. Para obter mais informações, consulte Atualização da versão secundária do Amazon DocumentDB.

- Recursos aprimorados de agregação com novos operadores (`$rand`,`$pow`,`$dateToParts`,`$dateFromParts`).
- Métrica de conexões ativas para monitorar instâncias. Para obter mais informações, consulte Cotas de instâncias.
- Métricas granulares de desempenho em nível de comando CloudWatch para operações CRUD (`findinsert`,`findAndModify`,`update`,`delete`,`aggregate`,`count`,`distinct`,`commitTransaction`) `getMoreabortTransaction`, exposição `AvgDuration` e `MaxConcurrent` dimensões para melhorar o `P100Duration` monitoramento operacional.
- O Amazon DocumentDB 5.0.1 só oferece suporte ao TLS 1.2 e superior para conexões de banco de dados.
- Foram `gcRuntimeStats` `documentFragmentStats` adicionadas e divididas as métricas do nível de coleta `storageSegmentExtended` em `storageSegmentBase` e `collstats` para mostrar mais informações sobre a integridade geral do processo de coleta de lixo.

Conceitos básicos do Amazon DocumentDB 5.0

Para começar a usar o Amazon DocumentDB 5.0, consulte o. Conceitos básicos do Amazon DocumentDB Você pode criar um novo cluster Amazon DocumentDB 5.0 usando o Console de gerenciamento da AWS ou o AWS SDK AWS CLI, ou. CloudFormation Ao se conectar ao Amazon DocumentDB, é necessário que você use um driver ou utilitário do MongoDB compatível com o MongoDB 5.0 ou superior.

> **Note**
>
> Ao usar o AWS SDK,, ou AWS CLI CloudFormation, a versão do mecanismo será predefinida como 5.0.0. É necessário especificar explicitamente o parâmetro `engineVersion = 4.0.0` para criar um novo cluster do Amazon DocumentDB 4.0 ou `engineVersion = 3.6.0` para criar um novo cluster do Amazon DocumentDB 3.6. Para um determinado cluster

do Amazon DocumentDB, você pode determinar a versão do cluster usando o console de gerenciamento do Amazon DocumentDB AWS CLI para chamar `describe-db-clusters` ou usar o console de gerenciamento do Amazon DocumentDB para visualizar o número da versão do mecanismo de um cluster específico.

O Amazon DocumentDB 5.0 e 8.0 oferece suporte aos processadores `r8g` Graviton4 e Graviton2 do Amazon EC2, como tipos de `t4.medium` instância para seus clusters, `r6g` e está disponível em todas as versões suportadas (consulte). Regiões da AWS Instâncias Para obter mais informações sobre preços, consulte Preços do Amazon DocumentDB (compatível com MongoDB).

Atualize ou migre para o Amazon DocumentDB 5.0

Você pode migrar do MongoDB 3.6 ou MongoDB 4.0 para o Amazon DocumentDB 5.0 usando os utilitários AWS DMS or, como mongodump, mongorestore, mongoimport e mongoexport. Para obter instruções sobre migração, consulteAtualizando seu cluster Amazon DocumentDB usando AWS Database Migration Service.

Diferenças funcionais

Diferenças funcionais entre Amazon DocumentDB 4.0 e 5.0

Com o lançamento do Amazon DocumentDB 5.0, há diferenças funcionais entre o Amazon DocumentDB 4.0 e o Amazon DocumentDB 5.0:

- A função integrada de backup agora é compatível com `serverStatus`. Ação: desenvolvedores e aplicações com função de backup podem coletar estatísticas sobre o estado do cluster Amazon DocumentDB.
- O campo `SecondaryDelaySecs` substitui `slaveDelay` na saída `replSetGetConfig`.
- O comando hello substitui `isMaster` - hello retorna um documento que descreve a função do cluster Amazon DocumentDB.
- O Amazon DocumentDB 5.0 agora oferece suporte a escaneamentos de índice com o operador `$elemMatch` no primeiro nível de agrupamento. As varreduras de índice são aceitas quando o filtro de consulta tem apenas um nível do filtro `$elemMatch`, mas não são aceitas se uma consulta aninhada `$elemMatch` for incluída.

  Por exemplo, no Amazon DocumentDB 5.0, se você incluir o operador `$elemMatch` no nível aninhado, ele não retornará um valor como no 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 ] } ] ] }
```

- A projeção “$” no Amazon DocumentDB 4.0 retorna todos os documentos com todos os campos. Com o Amazon DocumentDB 5.0, o comando find com uma projeção “$” retorna documentos que correspondem ao parâmetro de consulta contendo somente o campo que corresponde à projeção “$”.
- No Amazon DocumentDB 5.0, os comandos find com parâmetros de consulta `$regex` e `$options` retornam um erro: “Não é possível definir opções em `$regex` e `$options`”.
- Com o Amazon DocumentDB 5.0, `$indexOfCP` agora retorna “-1" quando:
  - a substring não foi encontrada na expressão da string, ou
  - inicia-se com um número maior que o final, ou
  - inicia-se com um número maior que o comprimento do byte da string.
- No Amazon DocumentDB 4.0, `$indexOfCP` retorna “0" quando a posição inicial é um número maior que o final ou o comprimento do byte da string.
- Com o Amazon DocumentDB 5.0, as operações de projeção em `_id fields`, como por exemplo `{"_id.nestedField" : 1}`, retornam documentos que incluem apenas o campo projetado. Já

no Amazon DocumentDB 4.0, os comandos de projeção de campo aninhados não filtram nenhum documento.

Compatibilidade do MongoDB 4.0

Tópicos

- Atributos do Amazon DocumentDB 4.0
- Conceitos básicos do Amazon DocumentDB 4.0
- Atualizar ou migrar para o Amazon DocumentDB 4.0
- Diferenças funcionais

Atributos do Amazon DocumentDB 4.0

O Amazon DocumentDB 4.0 introduziu muitos novos atributos e capacidades, que incluíram transações ACID e melhorias para alterar fluxos. O resumo abaixo apresenta alguns dos principais atributos que foram introduzidos no Amazon DocumentDB 4.0. Para ver uma lista completa dos recursos, consulte Notas da versão.

- Transações ACID: o Amazon DocumentDB agora oferece suporte à capacidade de realizar transações em vários documentos, declarações, coleções e bancos de dados. As transações simplificam o desenvolvimento de aplicações, permitindo que você execute operações atômicas, consistentes, isoladas e duráveis (ACID) em um ou mais documentos em um cluster do Amazon DocumentDB. Para obter mais informações, consulte Transações no Amazon DocumentDB.
- Fluxos de alteração: agora você tem a capacidade de abrir um fluxo de alterações no nível do cluster (`client.watch()` ou `mongo.watch()`) e do banco de dados (`db.watch()`), pode especificar um `startAtOperationTime` para abrir um cursor do fluxo de alterações e, por fim, agora pode estender o período de retenção do fluxo de alterações para 7 dias (anteriormente 24 horas). Para obter mais informações, consulte Usar fluxos de alterações com o Amazon DocumentDB.
- AWS Database Migration Service(AWS DMS): Agora você pode usar AWS DMS para migrar suas cargas de trabalho do MongoDB 4.0 para o Amazon DocumentDB. AWS DMS agora suporta uma fonte MongoDB 4.0, um destino Amazon DocumentDB 4.0 e uma fonte Amazon DocumentDB 3.6 para realizar atualizações entre o Amazon DocumentDB 3.6 e 4.0. Para obter mais informações, consulte a Documentação do AWS DMS.

- Desempenho e indexação: agora é possível utilizar um índice com `$lookup`, encontrar consultas com uma projeção que contenha um campo ou um campo, e o campo `_id` pode ser servido diretamente do índice e sem a necessidade de ler a coleção (consulta coberta), a capacidade de `hint()` com `findAndModify`, otimizações de desempenho para `$addToSet` e melhorias para reduzir os tamanhos gerais dos índices. Para obter mais informações, consulte Notas da versão.
- Operadores: o Amazon DocumentDB 4.0 agora oferece suporte a vários novos operadores de agregação: `$ifNull`, `$replaceRoot`, `$setIsSubset`, `$setIntersection`, `$setUnion`, `$setEquals`. É possível ver todas as APIs, operações e tipos de dados compatíveis do MongoDB no APIs, operações e tipos de dados do MongoDB compatíveis no Amazon DocumentDB.
- Controle de acesso baseado em função (RBAC): com ambos os comandos `ListCollection` e `ListDatabase`, agora é possível opcionalmente usar os parâmetros `authorizedCollections` e `authorizedDatabases` para permitir que os usuários listem as coleções e bancos de dados que eles têm permissão para acessar sem exigir as funções `listCollections` e `listDatabase`, respectivamente. Você também tem a habilidade de matar seus próprios cursores sem precisar da função `KillCursor`.

O Amazon DocumentDB não oferece suporte a todos os atributos do MongoDB 4.0. Quando criamos o Amazon DocumentDB 4.0, trabalhamos de trás para frente com base nos atributos e nas capacidades que nossos clientes mais pediram que criássemos. Continuaremos adicionando recursos adicionais do MongoDB 4.0 com base no feedback dos clientes. Por exemplo, o Amazon DocumentDB 4.0 atualmente não suporta os operadores de conversão de tipo ou os operadores de string que foram introduzidos no MongoDB 4.0. Para ver a lista mais recente de APIs compatíveis, consulteAPIs, operações e tipos de dados do MongoDB compatíveis no Amazon DocumentDB.

Conceitos básicos do Amazon DocumentDB 4.0

Para começar a usar o Amazon DocumentDB 4.0, consulte o. Conceitos básicos do Amazon DocumentDB Você pode criar um novo cluster Amazon DocumentDB 4.0 usando o Console de gerenciamento da AWS ou o AWS SDK AWS CLI, ou. CloudFormation Ao se conectar ao Amazon DocumentDB, é necessário que você use um driver ou utilitário do MongoDB compatível com o MongoDB 4.0 ou superior.

> **Note**
>
> Ao usar o AWS SDK,, ou AWS CLI CloudFormation, a versão do mecanismo será predefinida como 5.0.0. É necessário especificar explicitamente o parâmetro `engineVersion = 4.0.0` para criar um novo cluster do Amazon DocumentDB 4.0 ou `engineVersion =`

`3.6.0` para criar um novo cluster do Amazon DocumentDB 3.6. Para um determinado cluster do Amazon DocumentDB, você pode determinar a versão do cluster usando o console de gerenciamento do Amazon DocumentDB AWS CLI para chamar `describe-db-clusters` ou usar o console de gerenciamento do Amazon DocumentDB para visualizar o número da versão do mecanismo de um cluster específico.

O Amazon DocumentDB 4.0 oferece suporte a `r5r6g`,`t3.medium`, e tipos de `t4g.medium` instância para seus clusters e está disponível em todos os formatos compatíveis Regiões da AWS. Não existem custos adicionais para usar o Amazon DocumentDB 4.0. Para obter mais informações sobre preços, consulte Preços do Amazon DocumentDB (compatível com MongoDB).

Atualizar ou migrar para o Amazon DocumentDB 4.0

Você pode migrar do MongoDB 3.6 ou MongoDB 4.0 para o Amazon DocumentDB 4.0 usando os utilitários AWS DMS or, como mongodump, mongorestore, mongoimport e mongoexport. Da mesma forma, é possível usar as mesmas ferramentas para atualizar do Amazon DocumentDB 3.6 para o Amazon DocumentDB 4.0. Para obter instruções sobre migração, consulteAtualizando seu cluster Amazon DocumentDB usando AWS Database Migration Service.

Diferenças funcionais

Diferenças funcionais entre Amazon DocumentDB 3.6 e 4.0

Com o lançamento do Amazon DocumentDB 4.0, há diferenças funcionais entre o Amazon DocumentDB 3.6 e o Amazon DocumentDB 4.0:

- Projeção para documentos aninhados: o Amazon DocumentDB 3.6 considera o primeiro campo em um documento aninhado ao aplicar uma projeção. No entanto, o Amazon DocumentDB 4.0 analisará subdocumentos e também aplicará a projeção a cada subdocumento. Por exemplo: se a projeção for `"a.b.c": 1`, o comportamento nas duas versões será idêntico. No entanto, se a projeção for `{a:{b:{c:1}}}`, o Amazon DocumentDB 3.6 aplicará a projeção somente a 'a' e não a 'b' ou 'c'.
- Comportamento para **`minKey`**, **`maxKey`**: no Amazon DocumentDB 4.0, o comportamento para `{x: {$gt:MaxKey}}` retorna nada, e para `{x:{$lt:MaxKey}}` retorna tudo.
- Diferenças na comparação de documentos: a comparação de valores numéricos de diferentes tipos (double, int, long) em subdocumentos (por exemplo, b em `{"_id" :1, "a" :{"b":1}}`)

agora fornece uma saída consistente em todos os tipos de dados numéricos e para cada nível de um documento.

Diferenças funcionais entre Amazon DocumentDB 4.0 e MongoDB 4.0

Abaixo estão as diferenças funcionais entre o Amazon DocumentDB 4.0 e o MongoDB 4.0.

- Pesquisa com chave vazia no caminho: quando uma coleção contém um documento com chave vazia dentro da matriz (por exemplo, `{"x" : [ { "" : 10 }, { "b" : 20 } ]}`) e quando a chave usada na consulta termina em uma string vazia (por exemplo, `x.`), o Amazon DocumentDB retornará esse documento, pois percorre todos os documentos na matriz, enquanto o MongoDB não retornará esse documento.
- **`$setOnInsert`** junto com **`$`** no caminho: o operador de campo `$setOnInsert` não funcionará em combinação com $ no caminho no Amazon DocumentDB, que também é consistente com o MongoDB 4.0.

Diferenças funcionais: Amazon DocumentDB e MongoDB

Abaixo estão as diferenças funcionais entre o Amazon DocumentDB (compatível com MongoDB) e o MongoDB.

Tópicos

- Benefícios funcionais do Amazon DocumentDB
- Diferenças funcionais atualizadas
- Diferenças funcionais com o MongoDB

Benefícios funcionais do Amazon DocumentDB

Transações implícitas

No Amazon DocumentDB, todas as instruções CRUD (`findAndModify`, `update`, `insert`, `delete`) garantem atomicidade e consistência, até mesmo para operações que modificam vários documentos. Agora, com o lançamento do Amazon DocumentDB 4.0, há suporte para transações explícitas que fornecem propriedades ACID para operações com várias instruções e várias coleções. Para saber mais sobre o uso de transações no Amazon DocumentDB, consulteTransações no Amazon DocumentDB.

Veja a seguir exemplos de operações no Amazon DocumentDB que modificam vários documentos que satisfazem ambos os comportamentos atômico e consistente.

```
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 }]
})
```

As operações individuais que compõem operações em massa, como `updateMany` e `deleteMany`, são atômicas, mas a própria operação em massa não é atômica. Por exemplo, a totalidade da operação `insertMany` será atômica se as operações de inserção individuais forem executadas com êxito sem erros. Se um erro for encontrado em uma operação `insertMany`, cada instrução de inserção individual na operação `insertMany` será executada como uma operação atômica. Se você precisar de propriedades ACID para as operações `insertMany`, `updateMany` e `deleteMany`, é recomendável usar uma transação.

Diferenças funcionais atualizadas

O Amazon DocumentDB continua a melhorar a compatibilidade com o MongoDB, trabalhando retroativamente com as capacidades que nossos clientes nos pedem para criar. Esta seção contém as diferenças funcionais removidas do Amazon DocumentDB para facilitar as migrações e a criação de aplicações para nossos clientes.

Tópicos

- Indexação de matriz
- Multi-key índices
- Caracteres nulos em strings
- Role-based controle de acesso
- indexação $regex
- Projeção para documentos aninhados

Indexação de matriz

A partir de 23 de abril de 2020, o Amazon DocumentDB passa a oferecer suporte à capacidade de indexar matrizes maiores que 2.048 bytes. O limite para um item individual em uma matriz ainda permanece como 2.048 bytes, o que é consistente com MongoDB.

Se você estiver criando um novo índice, nenhuma ação será necessária para aproveitar a funcionalidade aprimorada. Se tiver um índice existente, você pode aproveitar a funcionalidade aprimorada, eliminando o índice e recriando-o. A versão atual do índice com os recursos aprimorados é `"v" : 3`.

> **Note**
>
> Para clusters de produção, a eliminação do índice pode ter um impacto no desempenho do seu aplicativo. Teste primeiro e continue com cuidado ao fazer alterações em um sistema de produção. Além disso, o tempo necessário para recriar o índice depende do tamanho geral dos dados da coleção.

É possível consultar a versão dos índices usando o seguinte comando.

```
db.collection.getIndexes()
```

A saída dessa operação é semelhante à seguinte. Nesta saída, a versão do índice é `"v" : 3`, que é a versão mais atual.

```
[
    {
        "v" : 3,
        "key" : {
        "_id" : 1
        },
        "name" : "_id_",
        "ns" : "test.test"
    }
]
```

Multi-key índices

A partir de 23 de abril de 2020, o Amazon DocumentDB passa a oferecer suporte à capacidade de criar um índice composto com várias chaves na mesma matriz.

Se você estiver criando um novo índice, nenhuma ação será necessária para aproveitar a funcionalidade aprimorada. Se tiver um índice existente, você pode aproveitar a funcionalidade aprimorada, eliminando o índice e recriando-o. A versão atual do índice com os recursos aprimorados é `"v" : 3`.

> **Note**
>
> Para clusters de produção, a eliminação do índice pode ter um impacto no desempenho do seu aplicativo. Teste primeiro e continue com cuidado ao fazer alterações em um sistema de

produção. Além disso, o tempo necessário para recriar o índice depende do tamanho geral dos dados da coleção.

É possível consultar a versão dos índices usando o seguinte comando.

```
db.collection.getIndexes()
```

A saída dessa operação é semelhante à seguinte. Nesta saída, a versão do índice é `"v" : 3`, que é a versão mais atual.

```
[
    {
        "v" : 3,
        "key" : {
            "_id" : 1
        },
        "name" : "_id_",
        "ns" : "test.test"
    }
]
```

Caracteres nulos em strings

Desde 22 de junho de 2020, o Amazon DocumentDB passou a oferecer suporte a caracteres nulos (`'\0'`) em strings.

Role-based controle de acesso

Desde 26 de março de 2020, o Amazon DocumentDB oferece suporte ao controle de acesso baseado em função (RBAC) para funções internas. Para saber mais, consulte Role-Based Controle de acesso.

`indexação $regex`

Desde 22 de junho de 2020, o Amazon DocumentDB passou a oferecer suporte à capacidade de operadores `$regex` utilizarem um índice.

Para utilizar um índice com o operador `$regex`, é necessário usar o comando `hint()`. Ao usar `hint()`, é necessário especificar o nome do campo no qual você está aplicando o `$regex`.

Por exemplo, se você tiver um índice no campo `product` com o nome do índice como p_1, `db.foo.find({product: /^x.*/}).hint({product:1})` usará o índice p_1, mas não usará o índice `db.foo.find({product: /^x.*/}).hint("p_1")`. É possível verificar se um índice é escolhido utilizando o comando `explain()` ou usando o profiler para registrar em log consultas lentas. Por exemplo, `.db.foo.find({product: /^x.*/}).hint("p_1").explain()`

> **Note**
>
> O método `hint()` só pode ser usado com um índice de cada vez.

O uso de um índice para uma consulta `$regex` é otimizado para consultas de regex que usam um prefixo e não especificam as opções de regex `i`, `m` ou `o`.

Ao usar um índice com `$regex`, é recomendável criar um índice em campos altamente seletivos onde o número de valores duplicados é inferior a 1% do número total de documentos na coleção. Por exemplo, se sua coleção contiver 100.000 documentos, crie índices somente em campos em que o mesmo valor ocorrer 1000 vezes ou menos.

Projeção para documentos aninhados

Há uma diferença funcional entre o Amazon DocumentDB e o MongoDB com o operador `$project` na versão 3.6 que foi resolvida no Amazon DocumentDB 4.0, mas que permanecerá sem suporte no Amazon DocumentDB 3.6.

O Amazon DocumentDB 3.6 só considera o primeiro campo em um documento aninhado ao aplicar uma projeção e o MongoDB 3.6 também analisará subdocumentos e aplicará a projeção a cada subdocumento.

Por exemplo: se a projeção for `"a.b.c": 1`, o comportamento funcionará conforme o esperado no Amazon DocumentDB e no MongoDB. No entanto, se a projeção for `{a:{b:{c:1}}}`, o Amazon DocumentDB 3.6 aplicará a projeção somente a `a` e não a `b` ou `c`. No Amazon DocumentDB 4.0, a projeção `{a:{b:{c:1}}}` será aplicada a `a`, `b` e `c`.

Diferenças funcionais com o MongoDB

Tópicos

- operador $vectorSearch

- OpCountersCommand
- Bancos de dados e coleções de administradores
- Tempo máximo do cursor MS
- explain()
- Compilações de índice
- Pesquisa com chave vazia no caminho
- APIs, operações e tipos de dados do MongoDB
- utilitários mongodump e mongorestore
- Ordenação de resultados
- Gravações repetíveis
- Índice esparso
- Usando $elemMatch em uma expressão $all
- Dólar ($) e ponto (.) nos nomes dos campos
- $lookup
- $ natural e classificação inversa

operador $vectorSearch

O Amazon DocumentDB não oferece suporte a `$vectorSearch` como um operador independente. Em vez disso, temos suporte para `vectorSearch` dentro do operador `$search`. Para obter mais informações, consulte Pesquisa vetorial para Amazon DocumentDB.

OpCountersCommand

O comportamento `OpCountersCommand` do Amazon DocumentDB se desvia do `opcounters.command` do MongoDB da seguinte forma:

- O `opcounters.command` do MongoDB conta todos os comandos, exceto inserir, atualizar e excluir, enquanto que o `OpCountersCommand` do Amazon DocumentDB também exclui o comando `find`.
- O Amazon DocumentDB considera alguns comandos internos com relação a `OpCountersCommand`.

Bancos de dados e coleções de administradores

O Amazon DocumentDB não oferece suporte ao banco de dados de administrador ou local nem às coleções do MongoDB `system.*` ou `startup_log` respectivamente.

Tempo máximo do cursor MS

No Amazon DocumentDB, `cursor.maxTimeMS` redefine o contador de cada solicitação `getMore`. Portanto, se for especificado 3.000 ms `maxTimeMS`, a consulta irá demorar 2.800 ms e cada solicitação `getMore` subsequente irá demorar 300 ms, depois, o cursor não atingirá o tempo limite. O cursor só atingirá o tempo limite quando uma única operação, seja a consulta ou uma determinada solicitação `getMore`, demorar mais do que o `maxTimeMS` especificado. Além disso, o varredor que verifica o tempo de execução do cursor é executado em uma granularidade de cinco (5) minutos.

explain()

O Amazon DocumentDB emula as APIs MongoDB 3.6, 4.0, 5.0 e 8.0 em um mecanismo de banco de dados criado especificamente que utiliza um sistema de armazenamento distribuído, tolerante a falhas e com recuperação automática. Como resultado, os planos de consulta e a saída de `explain()` podem diferir entre o Amazon DocumentDB e o MongoDB. Os clientes que desejam ter controle sobre seu plano de consulta podem usar o operador `$hint` para impor a seleção de um índice preferencial.

Compilações de índice

O Amazon DocumentDB permite que apenas uma compilação de índice ocorra em uma coleção a qualquer momento. Seja em primeiro plano ou em segundo plano. Se operações como `createIndex()` ou `dropIndex()` ocorrerem na mesma coleção quando uma compilação de índice estiver em andamento no momento, ocorrerá uma falha na operação que você tentou executar recentemente.

Por padrão, as construções de índices no Amazon DocumentDB e na versão 4.0 do MongoDB ocorrem em segundo plano. As versões 4.2 e posteriores do MongoDB ignoram a opção de construção do índice em segundo plano se especificada para createIndexes ou seus auxiliares de shell `createIndex()` e `createIndexes()`.

Um índice de Tempo de vida (TTL) começará a extinguir a validade de documentos depois que o índice de compilação for concluído.

Pesquisa com chave vazia no caminho

Quando você pesquisa com uma chave que inclui uma string vazia como parte do caminho (por exemplo, `x.`, `x..b`) e o objeto tem um caminho de chaves de string vazio (por exemplo, `{"x" : [ { "" : 10 }, { "b" : 20 } ]}`) dentro de uma matriz, o Amazon DocumentDB retornará resultados diferentes do que se você executasse a mesma pesquisa no MongoDB.

No MongoDB, a pesquisa do caminho de chaves vazio dentro da matriz funciona conforme o esperado quando a chave de string vazia não está no final da pesquisa do caminho. No entanto, quando a chave de string vazia está no final da pesquisa do caminho, ela não examina a matriz.

No entanto, no Amazon DocumentDB, somente o primeiro elemento dentro da matriz é lido, pois `getArrayIndexFromKeyString` converte uma string vazia em `0`, e, então, a pesquisa por chaves de string é tratada como uma pesquisa de índice de matriz.

APIs, operações e tipos de dados do MongoDB

O Amazon DocumentDB é compatível com as APIs MongoDB 3.6, 4.0, 5.0 e 8.0. Para obter uma lista atualizada das funcionalidades compatíveis, consulte APIs, operações e tipos de dados do MongoDB compatíveis no Amazon DocumentDB.

`utilitários mongodump` e `mongorestore`

O Amazon DocumentDB não oferece suporte a um banco de dados de administrador e, portanto, não despeja nem restaura o banco de dados de administrador ao usar os utilitários `mongodump` ou `mongorestore`. Ao criar um novo banco de dados no Amazon DocumentDB usando `mongorestore`, é necessário recriar as funções de usuário, além da operação de restauração.

> **Requisito de versão do MongoDB Database Tools**
>
> Use as ferramentas de banco de dados do MongoDB até a versão 100.11.0, inclusive, para o Amazon DocumentDB. Para fazer o download, consulte os lançamentos do MongoDB Database Tools no site do MongoDB.

Ordenação de resultados

O Amazon DocumentDB não garante a ordenação de resultados implícita dos conjuntos de resultados. Para garantir a ordenação de um conjunto de resultados, especifique explicitamente uma ordem de classificação usando `sort()`.

O exemplo a seguir classifica os itens na coleção de inventário em ordem decrescente com base no campo de estoque.

```
db.inventory.find().sort({ stock: -1 })
```

Ao usar o estágio de agregação `$sort`, a ordem de classificação não é preservada, a menos que o estágio `$sort` seja o último estágio no pipeline de agregação. Ao usar o estágio de agregação `$sort` em combinação com o estágio de agregação `$group`, o estágio de agregação `$sort` só é aplicado aos acumuladores `$first` e `$last`. No Amazon DocumentDB 4.0, foi adicionado suporte para `$push` para respeitar a ordem de classificação do estágio `$sort` anterior.

Gravações repetíveis

A maioria dos MongoDB-compatible drivers atuais permite gravações repetidas por padrão. O Amazon DocumentDB suporta gravações que podem ser repetidas a partir da versão 8.0.2 do mecanismo. Nas versões anteriores do mecanismo, não há suporte para gravações repetidas e você deve desativá-las para evitar erros.

Nas versões de mecanismo anteriores à 8.0.2, habilitar gravações que podem ser repetidas resulta em uma mensagem de erro semelhante à seguinte.

```
{"ok":0,"errmsg":"Retryable writes are not supported","code":301,"name":"MongoError"}
```

As gravações que podem ser repetidas podem ser desativadas por meio da string de conexão (por exemplo,`MongoClient("mongodb://my.mongodb.cluster/db?retryWrites=false")`) ou do argumento de palavra-chave do MongoClient construtor (por exemplo,). `MongoClient("mongodb://my.mongodb.cluster/db", retryWrites=False)`

Para obter mais informações sobre gravações que podem ser repetidas na versão 8.0.2 e posterior do mecanismo, consulte. Gravações que podem ser repetidas no Amazon DocumentDB

Índice esparso

Para usar um índice esparso que você criou em uma consulta, é necessário usar a cláusula `$exists` nos campos que abrangem o índice. Se você omitir `$exists`, o Amazon DocumentDB não usará o índice esparso.

Veja um exemplo do a seguir:

```
db.inventory.count({ "stock": { $exists: true }})
```

Para índices esparsos de várias chaves, o Amazon DocumentDB não oferecerá suporte a uma restrição de chave exclusiva se a pesquisa de um documento resultar em um conjunto de valores e apenas um subconjunto dos campos indexados estiver ausente. Por exemplo, `createIndex({"a.b" : 1 }, { unique : true, sparse :true })` não tem suporte, dada a entrada de `"a" : [ { "b" : 2 }, { "c" : 1 } ]`, já que `"a.c"` é armazenado no índice.

Usando **`$elemMatch`** em uma **`expressão $all`**

No momento, o Amazon DocumentDB não oferece suporte ao uso do operador `$elemMatch` dentro de uma expressão `$all`. Como solução alternativa, é possível usar o operador `$and` com `$elemMatch` da seguinte forma.

Operação original:

```
db.col.find({
  qty: {
    $all: [
      { "$elemMatch": { part: "xyz", qty: { $lt: 11 } } },
      { "$elemMatch": { num: 40, size: "XL" } }
    ]
  }
})
```

Operação atualizada:

```
db.col.find({
  $and: [
    { qty: { "$elemMatch": { part: "xyz", qty: { $lt: 11 } } } },
    { qty: { "$elemMatch": { qty: 40, size: "XL" } } }
  ]
})
```

Dólar ($) e ponto (.) nos nomes dos campos

O Amazon DocumentDB não oferece suporte à consulta de campos prefixados com dólar ($) em $in, $nin e $all em objetos aninhados. Por exemplo, a consulta a seguir não é válida no Amazon DocumentDB:

```
coll.find({"field": {"$all": [{ "$a": 1 }]}})
```

`$lookup`

O Amazon DocumentDB suporta a capacidade de fazer correspondências de igualdade (por exemplo, união externa esquerda) e também oferece suporte a subconsultas não correlacionadas. A partir da versão 8.0.2, o Amazon DocumentDB também oferece suporte a subconsultas correlacionadas, incluindo a sintaxe concisa de subconsultas correlacionadas. Em uma subconsulta correlacionada, somente um subconjunto de estágios de agregação que avaliam expressões pode referenciar as variáveis. `let` Para ver a lista de estágios suportados e restrições relacionadas, consulteEstágios suportados em subconsultas correlacionadas. Nas versões anteriores à 8.0.2, não há suporte para subconsultas correlacionadas.

Utilizando um índice com $lookup

Agora você pode utilizar um índice com o operador de estágio `$lookup`. Com base no seu caso de uso, há vários algoritmos de indexação que você pode usar para otimizar o desempenho. Esta seção explicará os diferentes algoritmos de indexação para `$lookup` e ajudará você a escolher o melhor para sua workload.

Por padrão, o Amazon DocumentDB utilizará o algoritmo de hash quando `allowDiskUse:false` for usado e a mesclagem de classificação quando `allowDiskUse:true` for usado .

> **Note**
>
> No momento, a opção `allowDiskUse` não recebem suporte para o comando `find`. A opção só tem suporte como parte da agregação. Use a estrutura de agregação com `allowDiskUse:true` para lidar com consultas grandes que podem exceder os limites de memória.

Para alguns casos de uso, pode ser melhor forçar o otimizador de consultas a usar um algoritmo diferente. Abaixo, estão os diferentes algoritmos de indexação que o operador de agregação `$lookup` pode utilizar:

- Loop aninhado: um plano de loop aninhado geralmente é benéfico para uma workload se a coleção externa tiver menos de 1 GB e o campo na coleção externa tiver um índice. Se o algoritmo de loop aninhado estiver sendo usado, o plano de explicação mostrará o estágio como `NESTED_LOOP_LOOKUP`.
- Mesclagem de classificação: um plano de mesclagem de classificação geralmente é benéfico para uma workload se a coleção externa não tiver um índice no campo usado na pesquisa e o conjunto de dados de trabalho não couber na memória. Se o algoritmo de mesclagem de classificação estiver sendo usado, o plano de explicação mostrará o estágio como `SORT_LOOKUP`.
- Hash: um plano de hash geralmente é benéfico para uma workload se a coleção externa for < 1 GB e o conjunto de dados de trabalho couber na memória. Se o algoritmo de hash estiver sendo usado, o plano de explicação mostrará o estágio como `HASH_LOOKUP`.

É possível identificar o algoritmo de indexação que está sendo usado pelo operador `$lookup` usando `explain` na consulta. Abaixo está um exemplo:

```
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 ao uso do método `explain()`, é possível usar o criador de perfil para revisar o algoritmo que está sendo utilizado com o uso do operador `$lookup`. Para obter mais informações sobre o criador de perfil, consulte[Definir o perfil das operações do Amazon DocumentDB](#).

Usando um **`PlanHint`**

Se você quiser forçar o otimizador de consultas a usar um algoritmo de indexação diferente com `$lookup`, você pode usar um `planHint`. Para fazer isso, use o comentário nas opções do estágio de agregação para forçar um plano diferente. Abaixo, está um exemplo de sintaxe do comentário:

```
comment : {
```

```
    comment :  "<string>",
    lookupStage : { planHint : "SORT" | "HASH" | "NESTED_LOOP" }
}
```

Abaixo, está um exemplo de uso do `planHint` para forçar o otimizador de consultas a usar o algoritmo de indexação `HASH`:

```
db.foo.aggregate(
   [
      {
         $lookup:
            {
               from: "foo",
               localField: "_id",
               foreignField: "_id",
               as: "joined"
            },
      }
   ]
),
{
   comment : "{ \"lookupStage\" : { \"planHint\": \"HASH\" }}"
```

Para testar qual algoritmo é melhor para sua workload, você pode usar o parâmetro `executionStats` do método `explain` para medir o tempo de execução do estágio `$lookup` enquanto modifica o algoritmo de indexação (ou seja, `HASH`/`SORT`/`NESTED_LOOP`).

O exemplo a seguir mostra como usar `executionStats` para medir o tempo de execução do estágio `$lookup` usando o algoritmo `SORT`.

```
db.foo.explain("executionStats").aggregate(
   [
      {
         $lookup:
            {
               from: "foo",
               localField: "_id",
               foreignField: "_id",
               as: "joined"
            },
```

```
      }
   ]
),
{
   comment : "{ \"lookupStage\" : { \"planHint\": \"SORT\" }}"
```

`$ natural` e classificação inversa

O Amazon DocumentDB oferece suporte somente a `$natural` para varreduras de coleções à frente. As varreduras de coleções reversas (`{$natural: -1}`) levarão a um `MongoServerError`.

APIs, operações e tipos de dados do MongoDB compatíveis no Amazon DocumentDB

O Amazon DocumentDB (compatível com MongoDB) é um serviço de banco de dados de documentos rápido, escalável, totalmente gerenciado e altamente disponível que oferece suporte a workloads. O Amazon DocumentDB é compatível com as APIs MongoDB 3.6, 4.0, 5.0 e 8.0. Esta seção lista as funcionalidades com suporte. Para obter suporte ao uso de APIs e drivers do MongoDB, consulte os fóruns da comunidade do MongoDB. Para obter suporte usando o serviço Amazon DocumentDB, entre em contato com a equipe de AWS suporte apropriada. Para saber as diferenças funcionais entre o Amazon DocumentDB e o MongoDB, consulte. Diferenças funcionais: Amazon DocumentDB e MongoDB

Os comandos e operadores do MongoDB somente para uso interno ou não aplicáveis a um serviço totalmente gerenciado não têm suporte e não são incluídos na lista de funcionalidades com suporte.

Adicionamos mais de 50 recursos desde o lançamento e continuaremos trabalhando para oferecer os recursos de que nossos clientes precisam. Para obter informações sobre os lançamentos mais recentes, consulte Anúncios do Amazon DocumentDB.

Se você quiser que um recurso passe a ter suporte, envie um e-mail informando seu ID de conta, os atributos solicitados e o caso de uso para a equipe de serviço do Amazon DocumentDB.

Tópicos

- Comandos do banco de dados
- Operadores de consulta e projeção
- Operadores de atualização

- Geoespacial
- Métodos de cursor
- Operadores de pipeline de agregação
- Tipos de dados
- Índices

Comandos do banco de dados

Tópicos

- Comandos administrativos
- Agregação
- Autenticação
- Comandos de diagnóstico
- Operações de gravação e de consulta
- Comandos de gerenciamento de perfis
- Comandos de sessão
- Gerenciamento de usuários
- Comandos de fragmentação

Comandos administrativos

Command	3.6	4,0	5,0	8.0	Cluster elástico
Coleções limitadas	Não	Não	Não	Não	Não
clone CollectionAsCapped	Não	Não	Não	Não	Não
collMod	Parcial	Parcial	Parcial	Parcial	Parcial

Command	3.6	4,0	5,0	8.0	Cluster elástico
CollMod: expiraAfterSeconds	Sim	Sim	Sim	Sim	Sim
CollMod: oculto	Não	Não	Sim (5.0.2+)	Sim (8.0.1+)	Não
converterToCapped	Não	Não	Não	Não	Não
copydb	Não	Não	Não	Não	Não
criar	Sim	Sim	Sim	Sim	Sim
createView	Não	Não	Não	Sim	Não
createIndexes	Sim	Sim	Sim	Sim	Sim
currentOp	Sim	Sim	Sim	Sim	Sim
drop	Sim	Sim	Sim	Sim	Sim
dropDatabase	Sim	Sim	Sim	Sim	Sim
dropIndexes	Sim	Sim	Sim	Sim	Sim
filemd5	Não	Não	Não	Não	Não
obterAuditConfig	Não	Sim	Sim	Sim	Não
killCursors	Sim	Sim	Sim	Sim	Sim
killOp	Sim	Sim	Sim	Sim	Sim

Command	3.6	4,0	5,0	8.0	Cluster elástico
listCollections*	Sim	Sim	Sim	Sim	Sim
listDatabases	Sim	Sim	Sim	Sim	Sim
listIndexes	Sim	Sim	Sim	Sim	Sim
reIndex	Não	Não	Sim	Sim	Não
renameCollection	Sim	Sim	Sim	Sim	Não
conjunto AuditConfig	Não	Sim	Sim	Sim	Não

* A chave `type` na opção de filtro não é compatível.

Agregação

Command	3.6	4,0	5,0	8.0	Cluster elástico
aggregate	Sim	Sim	Sim	Sim	Sim
contagem	Sim	Sim	Sim	Sim	Sim
distinct	Sim	Sim	Sim	Sim	Sim
mapReduce	Não	Não	Não	Sim	Não

Autenticação

Command	3.6	4,0	5,0	8.0	Cluster elástico
authenticate	Sim	Sim	Sim	Sim	Sim
logout	Sim	Sim	Sim	Sim	Sim

Comandos de diagnóstico

Command	3.6	4,0	5,0	8.0	Cluster elástico
buildInfo	Sim	Sim	Sim	Sim	Sim
collStats	Sim	Sim	Sim	Sim	Sim
enganar PoolStats	Não	Não	Não	Não	Não
connectio nStatus	Sim	Sim	Sim	Sim	Sim
dataSize	Sim	Sim	Sim	Sim	Sim
dbHash	Não	Não	Não	Não	Não
dbStats	Sim	Sim	Sim	Sim	Sim
explain	Sim	Sim	Sim	Sim	Sim
explain: execution Stats	Sim	Sim	Sim	Sim	Sim
recursos	Não	Não	Não	Não	Não
hostInfo	Sim	Sim	Sim	Sim	Sim

Command	3.6	4,0	5,0	8.0	Cluster elástico
listCommands	Sim	Sim	Sim	Sim	Sim
planoCacheClearFilters	Não	Não	Sim	Sim	Não
planoCacheListFilters	Não	Não	Sim	Sim	Não
planoCacheSetFilter	Não	Não	Sim	Sim	Não
profiler	Sim	Sim	Sim	Sim	Não
serverStatus	Sim	Sim	Sim	Sim	Sim
top	Sim	Sim	Sim	Sim	Sim

Operações de gravação e de consulta

Command	3.6	4,0	5,0	8.0	Cluster elástico
Change streams	Sim	Sim	Sim	Sim	Não
excluir	Sim	Sim	Sim	Sim	Sim
find	Sim	Sim	Sim	Sim	Sim
encontrarAndModify	Sim	Sim	Sim	Sim	Sim

Command	3.6	4,0	5,0	8.0	Cluster elástico
obter LastError	Não	Não	Não	Não	Não
getMore	Sim	Sim	Sim	Sim	Sim
obter PrevError	Não	Não	Não	Não	Não
GridFS	Sim	Sim	Sim	Sim	Não
insert	Sim	Sim	Sim	Sim	Sim
paralela CollectionScan	Não	Não	Não	Não	Não
resetError	Não	Não	Não	Não	Não
atualizar	Sim	Sim	Sim	Sim	Sim
ReplaceOne	Sim	Sim	Sim	Sim	Sim

Comandos de gerenciamento de perfis

3.6	4,0	5,0	8.0	Cluster elástico
Sim	Sim	Sim	Sim	Não
Sim	Sim	Sim	Sim	Não

C	3.6	4,0	5,0	8.0	Cluster elástico
c	Sim	Sim	Sim	Sim	Não
c F le	Sim	Sim	Sim	Sim	Não
n F F	Sim	Sim	Sim	Sim	Não
n F s	Sim	Sim	Sim	Sim	Não
n	Sim	Sim	Sim	Sim	Não
u e	Sim	Sim	Sim	Sim	Não

Comandos de sessão

Command	3.6	4,0	5,0	8.0	Cluster elástico
abortTransaction	Não	Sim	Sim	Sim	Não
commitTransaction	Não	Sim	Sim	Sim	Não
endSessions	Não	Não	Não	Não	Não
matarAllSessions	Não	Sim	Sim	Sim	Não

Command	3.6	4,0	5,0	8.0	Cluster elástico
matar AllSessio nsByPattern	Não	Não	Não	Não	Não
killSessions	Não	Sim	Sim	Sim	Não
refreshSe ssions	Não	Não	Não	Não	Não
startSession	Não	Sim	Sim	Sim	Não

Gerenciamento de usuários

Command	3.6	4,0	5,0	8.0	Cluster elástico
createUser	Sim	Sim	Sim	Sim	Sim
derrubar AllUsersF romDatabase	Sim	Sim	Sim	Sim	Sim
dropUser	Sim	Sim	Sim	Sim	Sim
conceder RolesToUser	Sim	Sim	Sim	Sim	Sim
revogar RolesFrom User	Sim	Sim	Sim	Sim	Sim
updateUser	Sim	Sim	Sim	Sim	Sim
usersInfo	Sim	Sim	Sim	Sim	Sim

Comandos de fragmentação

Command	Cluster elástico
abortar ReshardCollection	Não
addShard	Não
adicionar ShardToZone	Não
balanceador CollectionStatus	Não
balancerStart	Não
balancerStatus	Não
balancerStop	Não
verificar ShardingIndex	Não
claro JumboFlag	Não
cleanupOrphaned	Não
limpeza ReshardCollection	Não
comprometer ReshardCollection	Não
enableSharding	Sim
rubor RouterConfig	Não
obter ShardMap	Não
obter ShardVersion	Não
isdbgrid	Não
listShards	Não
medianKey	Não

Command	Cluster elástico
moveChunk	Não
movePrimary	Não
mergeChunks	Não
refinar CollectionShardKey	Não
removeShard	Não
remover ShardFromZone	Não
reshardCollection	Não
conjunto AllowMigrations	Não
conjunto ShardVersion	Não
shardCollection	Sim
shardingState	Não
dividir	Não
splitVector	Não
unsetSharding	Não
atualizar ZoneKeyRange	Não

Operadores de consulta e projeção

Tópicos

- Operadores de matriz
- Operadores bitwise
- Operador de comentários
- Operadores de comparação

- Operadores de elemento
- Operadores de consulta de avaliação
- Operadores lógicos
- Operadores de projeção

Operadores de matriz

Command	3.6	4,0	5,0	8.0	Cluster elástico
$all	Sim	Sim	Sim	Sim	Sim
$elemMatch	Sim	Sim	Sim	Sim	Sim
$size	Sim	Sim	Sim	Sim	Sim

Operadores bitwise

Command	3.6	4,0	5,0	8.0	Cluster elástico
$ bits AllSet	Sim	Sim	Sim	Sim	Sim
$ bits AnySet	Sim	Sim	Sim	Sim	Sim
$ bits AllClear	Sim	Sim	Sim	Sim	Sim
$ bits AnyClear	Sim	Sim	Sim	Sim	Sim

Operador de comentários

Command	3.6	4,0	5,0	8.0	Cluster elástico
$comment	Sim	Sim	Sim	Sim	Sim

Operadores de comparação

Command	3.6	4,0	5,0	8.0	Cluster elástico
$eq	Sim	Sim	Sim	Sim	Sim
$gt	Sim	Sim	Sim	Sim	Sim
$gte	Sim	Sim	Sim	Sim	Sim
$in	Sim	Sim	Sim	Sim	Sim
$lt	Sim	Sim	Sim	Sim	Sim
$lte	Sim	Sim	Sim	Sim	Sim
$ne	Sim	Sim	Sim	Sim	Sim
$nin	Sim	Sim	Sim	Sim	Sim

Operadores de elemento

Command	3.6	4,0	5,0	8.0	Cluster elástico
$exists	Sim	Sim	Sim	Sim	Sim
$type	Sim	Sim	Sim	Sim	Sim

Operadores de consulta de avaliação

Command	3.6	4,0	5,0	8.0	Cluster elástico
$expr	Não	Sim	Sim	Sim	Não
$jsonSchema	Não	Sim	Sim	Sim	Não
$mod	Sim	Sim	Sim	Sim	Sim
$regex	Sim	Sim	Sim	Sim	Sim
$text	Não	Não	Sim	Sim	Não
$where	Não	Não	Não	Não	Não

Operadores lógicos

Command	3.6	4,0	5,0	8.0	Cluster elástico
$and	Sim	Sim	Sim	Sim	Sim
$nor	Sim	Sim	Sim	Sim	Sim
$not	Sim	Sim	Sim	Sim	Sim
$or	Sim	Sim	Sim	Sim	Sim

Operadores de projeção

Command	3.6	4,0	5,0	8.0	Cluster elástico
$	Sim	Sim	Sim	Sim	Sim

Command	3.6	4,0	5,0	8.0	Cluster elástico
$elemMatch	Sim	Sim	Sim	Sim	Sim
$meta	Não	Não	Sim	Sim	Não
$slice	Sim	Sim	Sim	Sim	Sim

Operadores de atualização

Tópicos

- Operadores de matriz
- Operadores bitwise
- Operadores de campo
- Modificadores de atualização

Operadores de matriz

Command	3.6	4,0	5,0	8.0	Cluster elástico
$	Sim	Sim	Sim	Sim	Sim
$[]	Sim	Sim	Sim	Sim	Sim
$ [<identificador>]	Sim	Sim	Sim	Sim	Sim
$addToSet	Sim	Sim	Sim	Sim	Sim
$pop	Sim	Sim	Sim	Sim	Sim
$pullAll	Sim	Sim	Sim	Sim	Sim
$pull	Sim	Sim	Sim	Sim	Sim

Command	3.6	4,0	5,0	8.0	Cluster elástico
$push	Sim	Sim	Sim	Sim	Sim

Operadores bitwise

Command	3.6	4,0	5,0	8.0	Cluster elástico
$bit	Sim	Sim	Sim	Sim	Sim

Operadores de campo

Operador	3.6	4,0	5,0	8.0	Cluster elástico
$currentDate	Sim	Sim	Sim	Sim	Sim
$inc	Sim	Sim	Sim	Sim	Sim
$max	Sim	Sim	Sim	Sim	Sim
$min	Sim	Sim	Sim	Sim	Sim
$mul	Sim	Sim	Sim	Sim	Sim
$rename	Sim	Sim	Sim	Sim	Sim
$set	Sim	Sim	Sim	Sim	Sim
$setOnInsert	Sim	Sim	Sim	Sim	Sim
$unset	Sim	Sim	Sim	Sim	Sim

Modificadores de atualização

Operador	3.6	4,0	5,0	8.0	Cluster elástico
$each	Sim	Sim	Sim	Sim	Sim
$position	Sim	Sim	Sim	Sim	Sim
$slice	Sim	Sim	Sim	Sim	Sim
$sort	Sim	Sim	Sim	Sim	Sim

Geoespacial

Especificadores de geometria

Seletores de consulta	3.6	4,0	5,0	8.0	Cluster elástico
$box	Não	Não	Não	Não	Não
$center	Não	Não	Não	Não	Não
$centerSphere	Não	Não	Não	Não	Não
$geometry	Sim	Sim	Sim	Sim	Sim
$maxDistance	Sim	Sim	Sim	Sim	Sim
$minDistance	Sim	Sim	Sim	Sim	Sim
$nearSphere	Sim	Sim	Sim	Sim	Sim
$polygon	Não	Não	Não	Não	Não
$uniqueDocs	Não	Não	Não	Não	Não

Seletores de consulta

Command	3.6	4,0	5,0	8.0	Cluster elástico
$geoIntersects	Sim	Sim	Sim	Sim	Sim
$geoWithin	Sim	Sim	Sim	Sim	Sim
$near	Sim	Sim	Sim	Sim	Sim
$nearSphere	Sim	Sim	Sim	Sim	Sim
$polygon	Não	Não	Não	Não	Não
$uniqueDocs	Não	Não	Não	Não	Não

Métodos de cursor

Command	3.6	4,0	5,0	8.0	Cluster elástico
cursor.batchSize()	Sim	Sim	Sim	Sim	Sim
cursor.close()	Sim	Sim	Sim	Sim	Sim
cursor.collation()	Não	Não	Não	Sim	Não
cursor.comment()	Sim	Sim	Sim	Sim	Sim
cursor.count()	Sim	Sim	Sim	Sim	Sim
cursor.explain()	Sim	Sim	Sim	Sim	Não

Command	3.6	4,0	5,0	8.0	Cluster elástico
cursor.forEach()	Sim	Sim	Sim	Sim	Sim
cursor.hasNext()	Sim	Sim	Sim	Sim	Sim
cursor.hint()	Sim	Sim	Sim	Sim	Sim*
cursor.isClosed()	Sim	Sim	Sim	Sim	Sim
cursor.isExhausted()	Sim	Sim	Sim	Sim	Não
cursor.itcount()	Sim	Sim	Sim	Sim	Não
cursor.limit()	Sim	Sim	Sim	Sim	Não
cursor.map()	Sim	Sim	Sim	Sim	Não
cursor.max()	Não	Não	Sim (5.0.2+)	Sim (8.0.1+)	Não
cursor.maxScan()	Sim	Sim	Sim	Sim	Não
cursor.maxTimeMS()	Sim	Sim	Sim	Sim	Não
cursor.min()	Não	Não	Sim (5.0.2+)	Sim (8.0.1+)	Não
cursor.next()	Sim	Sim	Sim	Sim	Sim
CursorTimeoutcursor.no()	Não	Não	Não	Não	Não

Command	3.6	4,0	5,0	8.0	Cluster elástico
cursor.objsLeftInBatch ()	Sim	Sim	Sim	Sim	Não
cursor.pretty()	Sim	Sim	Sim	Sim	Não
cursor.readConcern()	Sim	Sim	Sim	Sim	Não
cursor.readPref()	Sim	Sim	Sim	Sim	Não
cursor.returnKey()	Não	Não	Não	Não	Não
cursor.showRecordId ()	Não	Não	Não	Não	Não
cursor.size()	Sim	Sim	Sim	Sim	Não
cursor.skip()	Sim	Sim	Sim	Sim	Não
cursor.sort()	Sim	Sim	Sim	Sim	Não
cursor.tailable()	Não	Não	Não	Não	Não
cursor.toArray()	Sim	Sim	Sim	Sim	Não

* O `hint` de índice é compatível com expressões de índice. Por exemplo, `.db.foo.find().hint({x:1})`

Os `cursor.max()` métodos `cursor.min()` e exigem um `hint` que nomeie o índice a ser usado. Ambos os métodos devem especificar os mesmos campos e você não pode combiná-los com uma `$natural` classificação. Eles são suportados apenas pelo `find` comando.

Operadores de pipeline de agregação

Tópicos

- Expressões do acumulador
- Operadores aritméticos
- Operadores de matriz
- Operadores booleanos
- Operadores de comparação
- Operadores de expressão condicional
- Operador de tipo de dados
- Operador de tamanho de dados
- Operadores de data
- Operador literal
- Operador de mesclagem
- Operador natural
- Configurar operadores
- Operadores de estágio
- Operadores de string
- Variáveis do sistema
- Operador de pesquisa de texto
- Operadores de conversão de tipo
- Operadores variáveis
- Operadores de trigonometria
- Operadores bit a bit (agregação)
- Operadores de registro de data e hora
- Operadores diversos

Expressões do acumulador

Expressão	3.6	4,0	5,0	8.0	Cluster elástico
$accumulator	-	-	Não	Não	Não
$adicionarToSet	Sim	Sim	Sim	Sim	Sim
$avg	Sim	Sim	Sim	Sim	Sim
$ fundo	-	-	-	Sim (8.0.1+)	Não
$bottomN	-	-	-	Sim (8.0.1+)	Não
$count	-	-	Não	Sim (8.0.1+)	Não
$covariancePop	Não	Não	Não	Sim (8.0.2+)	Não
$covarianceSamp	Não	Não	Não	Sim (8.0.2+)	Não
$denseRank	Não	Não	Não	Sim (8.0.2+)	Não
$derivative	Não	Não	Não	Não	Não
$documentNumber	Não	Não	Não	Sim (8.0.2+)	Não
$ expMovingAvg	Não	Não	Não	Não	Não
$first	Sim	Sim	Sim	Sim	Sim
$primeiroN	-	-	-	Sim (8.0.1+)	Não
$integral	Não	Não	Não	Não	Não
$last	Sim	Sim	Sim	Sim	Sim

Expressão	3.6	4,0	5,0	8.0	Cluster elástico
$LastN	-	-	-	Sim (8.0.1+)	Não
$ LinearFill	Não	Não	Não	Não	Não
$locf	Não	Não	Não	Não	Não
$max	Sim	Sim	Sim	Sim	Sim
$ máximo	-	-	-	Sim (8.0.1+)	Não
$ mediana	-	-	-	Sim (8.0.1+)	Não
$min	Sim	Sim	Sim	Sim	Sim
$ min	-	-	-	Sim (8.0.1+)	Não
$ percentil	-	-	-	Sim (8.0.1+)	Não
$push	Sim	Sim	Sim	Sim	Sim
$rank	Não	Não	Não	Sim (8.0.2+)	Não
$shift	Não	Não	Não	Não	Não
$ std DevPop	Não	Não	Não	Sim (8.0.1+)	Não
$ std DevSamp	Não	Não	Não	Sim (8.0.1+)	Não
$sum	Sim	Sim	Sim	Sim	Sim
$top	-	-	-	Sim (8.0.1+)	Não
$ para PN	-	-	-	Sim (8.0.1+)	Não

Operadores aritméticos

Command	3.6	4,0	5,0	8.0	Cluster elástico
$abs	Sim	Sim	Sim	Sim	Sim
$add	Sim	Sim	Sim	Sim	Sim
$ceil	Não	Sim	Sim	Sim	Sim
$divide	Sim	Sim	Sim	Sim	Sim
$exp	Não	Sim	Sim	Sim	Sim
$floor	Não	Sim	Sim	Sim	Sim
$ln	Não	Sim	Sim	Sim	Sim
$log	Não	Sim	Sim	Sim	Sim
$log10	Não	Sim	Sim	Sim	Sim
$mod	Sim	Sim	Sim	Sim	Sim
$multiply	Sim	Sim	Sim	Sim	Sim
$pow	Não	Não	Sim (5.0.1+)	Sim	Não
$round	-	-	Não	Sim (8.0.1+)	Não
$sqrt	Não	Sim	Sim	Sim	Sim
$subtract	Sim	Sim	Sim	Sim	Sim
$trunc	Não	Não	Não	Sim (8.0.1+)	Não

Operadores de matriz

Command	3.6	4,0	5,0	8.0	Cluster elástico
$matrizElemAt	Sim	Sim	Sim	Sim	Sim
$matrizToObject	Sim	Sim	Sim	Sim	Sim
$concatArrays	Sim	Sim	Sim	Sim	Sim
$filter	Sim	Sim	Sim	Sim	Sim
$first	-	-	Sim	Sim	Não
$in	Sim	Sim	Sim	Sim	Sim
$índiceOfArray	Sim	Sim	Sim	Sim	Sim
$isArray	Sim	Sim	Sim	Sim	Sim
$last	-	-	Sim	Sim	Não
$objetoToArray	Sim	Sim	Sim	Sim	Sim
$range	Sim	Sim	Sim	Sim	Sim
$reverseArray	Sim	Sim	Sim	Sim	Sim
$SortArray	Não	Não	Não	Sim	Não
$reduce	Sim	Sim	Sim	Sim	Sim
$size	Sim	Sim	Sim	Sim	Sim

Command	3.6	4,0	5,0	8.0	Cluster elástico
$slice	Sim	Sim	Sim	Sim	Sim
$zip	Sim	Sim	Sim	Sim	Sim

Operadores booleanos

Command	3.6	4,0	5,0	8.0	Cluster elástico
$and	Sim	Sim	Sim	Sim	Sim
$not	Sim	Sim	Sim	Sim	Sim
$or	Sim	Sim	Sim	Sim	Sim

Operadores de comparação

Command	3.6	4,0	5,0	8.0	Cluster elástico
$cmp	Sim	Sim	Sim	Sim	Sim
$eq	Sim	Sim	Sim	Sim	Sim
$gt	Sim	Sim	Sim	Sim	Sim
$gte	Sim	Sim	Sim	Sim	Sim
$lt	Sim	Sim	Sim	Sim	Sim
$lte	Sim	Sim	Sim	Sim	Sim
$ne	Sim	Sim	Sim	Sim	Sim

Operadores de expressão condicional

Command	3.6	4,0	5,0	8.0	Cluster elástico
$cond	Sim	Sim	Sim	Sim	Sim
$ifNull	Sim	Sim	Sim	Sim	Sim
$switch	Não	Sim	Sim	Sim	Não

Operador de tipo de dados

Command	3.6	4,0	5,0	8.0	Cluster elástico
$type	Sim	Sim	Sim	Sim	Sim

Operador de tamanho de dados

Command	3.6	4,0	5,0	8.0	Cluster elástico
$binarySize	-	-	Não	Sim (8.0.1+)	Não
$bsonSize	-	-	Não	Sim (8.0.1+)	Não

Operadores de data

Command	3.6	4,0	5,0	8.0	Cluster elástico
$dateAdd	Não	Não	Sim	Sim	Sim
$dateDiff	-	-	Sim	Sim	Não

Command	3.6	4,0	5,0	8.0	Cluster elástico
$dataFromParts	Não	Não	Sim (5.0.1+)	Não	Não
$dataFromString	Sim	Sim	Sim	Sim	Sim
$dateSubtract	Não	Não	Sim	Sim	Sim
$dataToParts	Não	Não	Sim (5.0.1+)	Não	Não
$dataToString	Sim	Sim	Sim	Sim	Sim
$dateTrunc	-	-	Não	Sim	Não
$dia OfMonth	Sim	Sim	Sim	Sim	Sim
$dia OfWeek	Sim	Sim	Sim	Sim	Sim
$dia OfYear	Sim	Sim	Sim	Sim	Sim
$hour	Sim	Sim	Sim	Sim	Sim
$ isoDayOfWeek	Sim	Sim	Sim	Sim	Sim
$isoWeek	Sim	Sim	Sim	Sim	Sim
$ isoWeekYear	Sim	Sim	Sim	Sim	Sim
$millisecond	Sim	Sim	Sim	Sim	Sim
$minute	Sim	Sim	Sim	Sim	Sim
$month	Sim	Sim	Sim	Sim	Sim

Command	3.6	4,0	5,0	8.0	Cluster elástico
$second	Sim	Sim	Sim	Sim	Sim
$week	Sim	Sim	Sim	Sim	Sim
$year	Sim	Sim	Sim	Sim	Sim

Operador literal

Command	3.6	4,0	5,0	8.0	Cluster elástico
$literal	Sim	Sim	Sim	Sim	Sim

Operador de mesclagem

Command	3.6	4,0	5,0	8.0	Cluster elástico
$mergeObjects	Sim	Sim	Sim	Sim	Sim

Operador natural

Command	3.6	4,0	5,0	8.0	Cluster elástico
$natural	Sim	Sim	Sim	Sim	Sim

Configurar operadores

Command	3.6	4,0	5,0	8.0	Cluster elástico
$tudo ElementsTrue	Não	Sim	Sim	Sim	Sim
$ qualquer ElementTrue	Não	Sim	Sim	Sim	Sim
$setDifference	Não	Sim	Sim	Sim	Sim
$setEquals	Sim	Sim	Sim	Sim	Sim
$setIntersection	Sim	Sim	Sim	Sim	Sim
$ set IsSubset	Sim	Sim	Sim	Sim	Sim
$setUnion	Sim	Sim	Sim	Sim	Sim

Operadores de estágio

Command	3.6	4,0	5,0	8.0	Cluster elástico
$addFields	Sim	Sim	Sim	Sim	Sim
$bucket	Não	Não	Não	Sim	Não
$bucketAuto	Não	Não	Não	Sim (8.0.2+)	Não
$changeStream	Sim	Sim	Sim	Sim	Não

Command	3.6	4,0	5,0	8.0	Cluster elástico
$ alterar StreamSplitLargeEvent	Não	Não	Não	Sim (8.0.2+)	Não
$collStats	Não	Sim	Sim	Sim	Não
$count	Sim	Sim	Sim	Sim	Sim
$currentOp	Sim	Sim	Sim	Sim	Sim
$ densidade	Não	Não	Não	Não	Não
$facet	Não	Não	Não	Sim (8.0.2+)	Não
$preencher	Não	Não	Não	Não	Não
$geoNear	Sim	Sim	Sim	Sim	Sim
$graphLookup	Não	Não	Não	Sim	Não
$group	Sim	Sim	Sim	Sim	Sim
$indexStats	Sim	Sim	Sim	Sim	Não
$limit	Sim	Sim	Sim	Sim	Sim
$lista LocalSessions	Não	Não	Não	Não	Não
$listSessions	Não	Não	Não	Não	Não
$lookup	Sim	Sim	Sim	Sim	Sim
$match	Sim	Sim	Sim	Sim	Sim
$merge	-	-	Não	Sim	Não

Command	3.6	4,0	5,0	8.0	Cluster elástico
$out	Sim	Sim	Sim	Sim	Não
$planCacheStats	-	-	Não	Não	Não
$project	Sim	Sim	Sim	Sim	Sim
$redact	Sim	Sim	Sim	Sim	Sim
$replaceRoot	Sim	Sim	Sim	Sim	Sim
$sample	Sim	Sim	Sim	Sim	Sim
$set	-	-	Não	Sim	Não
$setWindowFields	-	-	Não	Sim (8.0.2+)	Não
$skip	Sim	Sim	Sim	Sim	Sim
$sort	Sim	Sim	Sim	Sim	Sim
$sortByCount	Não	Não	Não	Sim (8.0.1+)	Não
$unionWith	-	-	Não	Não	Não
$unset	-	-	Não	Sim	Não
$unwind	Sim	Sim	Sim	Sim	Sim
$substituir por	Não	Não	Não	Sim	Não
$VectorSearch	Não	Não	Não	Sim	Não

Command	3.6	4,0	5,0	8.0	Cluster elástico
$listaSearchIndexes	-	-	-	Sim (8.0.1+)	Não

Operadores de string

Command	3.6	4,0	5,0	8.0	Cluster elástico
$concat	Sim	Sim	Sim	Sim	Sim
$índiceOfBytes	Sim	Sim	Sim	Sim	Sim
$indexOfCP	Sim	Sim	Sim	Sim	Sim
$ltrim	Não	Sim	Sim	Sim	Não
$regexFind	-	-	Sim	Sim	Não
$ regexFindAll	-	-	Sim	Sim	Não
$regexMatch	-	-	Sim	Sim	Não
$replaceAll	-	-	Sim	Sim	Não
$replaceOne	-	-	Sim	Sim	Não
$rtrim	Não	Sim	Sim	Sim	Não
$split	Sim	Sim	Sim	Sim	Sim
$strcasecmp	Sim	Sim	Sim	Sim	Sim

Command	3.6	4,0	5,0	8.0	Cluster elástico
$ str LenBytes	Sim	Sim	Sim	Sim	Sim
$strLenCP	Sim	Sim	Sim	Sim	Sim
$substr	Sim	Sim	Sim	Sim	Sim
$substrBytes	Sim	Sim	Sim	Sim	Sim
$substrCP	Sim	Sim	Sim	Sim	Sim
$toLower	Sim	Sim	Sim	Sim	Sim
$toUpper	Sim	Sim	Sim	Sim	Sim
$trim	Não	Sim	Sim	Sim	Não

Variáveis do sistema

Command	3.6	4,0	5,0	8.0	Cluster elástico
$$CURRENT	Não	Não	Não	Não	Não
$$DESCEND	Sim	Sim	Sim	Sim	Sim
$$KEEP	Sim	Sim	Sim	Sim	Sim
$$PRUNE	Sim	Sim	Sim	Sim	Sim
$$REMOVE	Não	Não	Não	Não	Não
$ ROOT	Sim	Sim	Sim	Sim	Sim

Operador de pesquisa de texto

Command	3.6	4,0	5,0	8.0	Cluster elástico
$meta	Não	Não	Sim	Sim	Não
$search	Não	Não	Sim	Sim	Não

Operadores de conversão de tipo

Command	3.6	4,0	5,0	8.0	Cluster elástico
$convert	Não	Sim	Sim	Sim	Sim
$isNumber	-	-	Não	Sim (8.0.1+)	Não
$toBool	Não	Sim	Sim	Sim	Sim
$toDate	Não	Sim	Sim	Sim	Sim
$toDecimal	Não	Sim	Sim	Sim	Sim
$toDouble	Não	Sim	Sim	Sim	Sim
$toInt	Não	Sim	Sim	Sim	Sim
$toLong	Não	Sim	Sim	Sim	Sim
$ para ObjectId	Não	Sim	Sim	Sim	Sim
$toString	Não	Sim	Sim	Sim	Sim
$ para UUID	-	-	-	Sim (8.0.1+)	Não

Operadores variáveis

Command	3.6	4,0	5,0	8.0	Cluster elástico
$let	Sim	Sim	Sim	Sim	Sim
$map	Sim	Sim	Sim	Sim	Sim

Operadores de trigonometria

Command	3.6	4,0	5,0	8.0	Cluster elástico
$ acos	-	-	-	Sim (8.0.1+)	Não
$acosh	-	-	-	Sim (8.0.1+)	Não
$asin	-	-	-	Sim (8.0.1+)	Não
$asinh	-	-	-	Sim (8.0.1+)	Não
$satan	-	-	-	Sim (8.0.1+)	Não
$atan2	-	-	-	Sim (8.0.1+)	Não
$atanah	-	-	-	Sim (8.0.1+)	Não
$cos	-	-	-	Sim (8.0.1+)	Não
$ cosh	-	-	-	Sim (8.0.1+)	Não
$ graus ToRadians	-	-	-	Sim (8.0.1+)	Não
$ radianos ToDegrees	-	-	-	Sim (8.0.1+)	Não
$sin	-	-	-	Sim (8.0.1+)	Não

Command	3.6	4,0	5,0	8.0	Cluster elástico
$sinh	-	-	-	Sim (8.0.1+)	Não
$ bronzeado	-	-	-	Sim (8.0.1+)	Não
$tanh	-	-	-	Sim (8.0.1+)	Não

Operadores bit a bit (agregação)

Command	3.6	4,0	5,0	8.0	Cluster elástico
$ bit e	-	-	-	Sim (8.0.1+)	Não
$bitNot	-	-	-	Sim (8.0.1+)	Não
$ bit ou	-	-	-	Sim (8.0.1+)	Não
$ bitX ou	-	-	-	Sim (8.0.1+)	Não

Operadores de registro de data e hora

Command	3.6	4,0	5,0	8.0	Cluster elástico
Incremento de $ T	-	-	-	Sim (8.0.1+)	Não
$ tSecond	-	-	-	Sim (8.0.1+)	Não

Operadores diversos

Command	3.6	4,0	5,0	8.0	Cluster elástico
$getField	-	-	Sim	Sim	Não
$rand	-	-	Sim (5.0.1+)	Sim	Não
$sampleRate	-	-	Não	Sim (8.0.1+)	Não
$ sigmóide	-	-	-	Sim (8.0.1+)	Não

Tipos de dados

Command	3.6	4,0	5,0	8.0	Cluster elástico
Inteiro de 32 bit (int)	Sim	Sim	Sim	Sim	Sim
Inteiro de 64 bits (longo)	Sim	Sim	Sim	Sim	Sim
Array	Sim	Sim	Sim	Sim	Sim
Dados binários	Sim	Sim	Sim	Sim	Sim
Booleano	Sim	Sim	Sim	Sim	Sim
Data	Sim	Sim	Sim	Sim	Sim
DBPointer	Não	Não	Não	Não	Não
DBRefs	Não	Não	Não	Não	Não
Decimal128	Sim	Sim	Sim	Sim	Sim

Command	3.6	4,0	5,0	8.0	Cluster elástico
Duplo	Sim	Sim	Sim	Sim	Sim
JavaScript	Não	Não	Não	Não	Não
JavaScript(com escopo)	Não	Não	Não	Não	Não
MaxKey	Sim	Sim	Sim	Sim	Sim
MinKey	Sim	Sim	Sim	Sim	Sim
Null	Sim	Sim	Sim	Sim	Sim
Objeto	Sim	Sim	Sim	Sim	Sim
ObjectId	Sim	Sim	Sim	Sim	Sim
Expressão Regular	Sim	Sim	Sim	Sim	Sim
String	Sim	Sim	Sim	Sim	Sim
Símbolo	Não	Não	Não	Não	Não
Timestamp	Sim	Sim	Sim	Sim	Sim
Não definido	Não	Não	Não	Não	Não

Índices e propriedades de índice

Tópicos

- Índices
- Propriedades de índice

Índices

Command	3.6	4,0	5,0	8.0	Cluster elástico
2dsphere	Sim	Sim	Sim	Sim	Sim
Índice 2d	Não	Não	Não	Não	Não
Índice Composto	Sim	Sim	Sim	Sim	Sim
Índice com hash	Não	Não	Não	Não	Não
Índice de várias chaves	Sim	Sim	Sim	Sim	Sim
Índice de campo único	Sim	Sim	Sim	Sim	Sim
Índice de texto	Não	Não	Sim	Sim	Não
Curinga	Não	Não	Não	Não	Não

Propriedades de índice

Command	3.6	4,0	5,0	8.0	Cluster elástico
Contexto	Sim	Sim	Sim	Sim	Sim
Diferenciação entre maiúsculas e minúsculas	Não	Não	Não	Sim	Não

Command	3.6	4,0	5,0	8.0	Cluster elástico
Oculto	Não	Não	Não	Sim (8.0.1+)	Não
Parcial	Não	Não	Sim	Sim	Não
Sparse	Sim	Sim	Sim	Sim	Sim
Texto	Não	Não	Sim	Sim	Não
TTL	Sim	Sim	Sim	Sim	Sim
Exclusivo	Sim	Sim	Sim	Sim	Sim
Vector	Não	Não	Sim	Sim	Não

Para obter informações detalhadas sobre operadores específicos do MongoDB, consulte os seguintes tópicos:

- Operadores de pipeline de agregação
- Geoespacial
- Operadores de projeção
- Operadores de atualização

Operadores de pipeline de agregação

Esta seção fornece informações detalhadas sobre operadores de pipeline de agregação suportados pelo Amazon DocumentDB.

Tópicos

- $$DESCEND
- $$KEEP
- $$PRUNE
- $ ROOT
- $abs

- $ acos
- $acosh
- $add
- $adicionar ToSet
- $addFields
- $tudo ElementsTrue
- $and
- $ qualquer ElementTrue
- $matriz ElemAt
- $matriz ToObject
- $asin
- $asinh
- $satan
- $atan2
- $atanah
- $avg
- $binarySize
- $ bit e
- $bitNot
- $ bit ou
- $ bitX ou
- $ fundo
- $bottomN
- $bsonSize
- $bucket
- $bucketAuto
- $ceil
- $changeStream
- $ alterar StreamSplitLargeEvent

- $cmp
- $collStats
- $concat
- $concatArrays
- $cond
- $convert
- $cos
- $ cosh
- $ count (estágio)
- $ count (acumulador)
- $covariancePop
- $covarianceSamp
- $currentOp
- $dateAdd
- $dateDiff
- $data FromParts
- $data FromString
- $dateSubtract
- $data ToParts
- $data ToString
- $dateTrunc
- $dia OfMonth
- $dia OfWeek
- $dia OfYear
- $ graus ToRadians
- $denseRank
- $divide
- $documentNumber
- $eq

- $exp
- $facet
- $filter
- $first
- $primeiroN
- $floor
- $geoNear
- $graphLookup
- $group
- $gt
- $gte
- $hour
- $ifNull
- $in
- $índice OfArray
- $índice OfBytes
- $indexOfCP
- $indexStats
- $isArray
- $isNumber
- $ iso DayOfWeek
- $isoWeek
- $ iso WeekYear
- $last
- $LastN
- $let
- $limit
- $lista SearchIndexes
- $literal

- $ln
- $log
- $log10
- $lt
- $lte
- $lookup
- $ltrim
- $map
- $match
- $max
- $ máximo
- $ mediana
- $meta
- $merge
- $mergeObjects
- $millisecond
- $min
- $ min
- $minute
- $mod
- $month
- $multiply
- $natural
- $ne
- $not
- $objeto ToArray
- $or
- $out
- $ percentil

- $pow
- $push
- $project
- $ radianos ToDegrees
- $rand
- $range
- $rank
- $redact
- $reduce
- $regexFind
- $ regex FindAll
- $regexMatch
- $replaceAll
- $replaceOne
- $replaceRoot
- $ substituir por
- $reverseArray
- $round
- $rtrim
- $sample
- $search
- $second
- $set
- $setDifference
- $setEquals
- $setIntersection
- $ set IsSubset
- $setUnion
- $set WindowFields

- $ sigmóide
- $sin
- $sinh
- $skip
- $slice
- $size
- $sort
- $SortArray
- $ordenar ByCount
- $split
- $sqrt
- $ std DevPop
- $ std DevSamp
- $ str LenBytes
- $strLenCP
- $strcasecmp
- $substr
- $substrBytes
- $substrCP
- $subtract
- $sum
- $switch
- $ bronzeado
- $tanh
- $top
- $ para PN
- $trunc
- Incremento de $ T
- $ tSecond
- $toBool

- $toDate
- $toDecimal
- $toDouble
- $toInt
- $toLong
- $toLower
- $ para ObjectId
- $toString
- $toUpper
- $ para UUID
- $trim
- $type
- $unset
- $unwind
- $VectorSearch
- $week
- $year
- $zip

$$DESCEND

O `$$DESCEND` operador no Amazon DocumentDB é um operador especial de matriz posicional usado no estágio do `$redact` pipeline. Ele instrui o pipeline de agregação a descer até o documento atual e processar todos os campos, independentemente do nível de aninhamento.

Quando o `$redact` estágio encontra o `$$DESCEND` operador, ele mantém todos os campos do documento atual visíveis e os processa mais adiante. Isso é útil quando você deseja redigir ou remover seletivamente determinados campos com base em uma condição, mantendo a estrutura do documento.

Parâmetros

Nenhum.

Exemplo (MongoDB Shell)

Neste exemplo, usaremos o `$redact` estágio com o `$$DESCEND` operador para exibir seletivamente documentos em que o `code` campo é igual a “Reg”.

Crie documentos de amostra

```
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" }
]);
```

Exemplo de consulta

```
db.patient.aggregate([
  { $redact: {
    $cond: {
      if: { $eq: ["Reg", "$code"] },
      then: "$$DESCEND",
      else: "$$PRUNE"
    }
  }}
]);
```

Saída

```
{
  "_id": 2,
  "code": "Reg",
  "patient": "Jane Doe",
  "DOB": "3/27/1989",
  "Hospital": "Cherry Hill"
}
```

Exemplos de código

Para ver um exemplo de código para usar o `$$DESCEND` comando, escolha a guia do idioma que você deseja 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

A variável $$KEEP do sistema é usada com o $redact estágio no pipeline de agregação para manter o documento ou campo atual inalterado e incluí-lo na saída.

Parâmetros

Nenhum

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso de $$KEEP em um pipeline de agregação do Amazon DocumentDB. Os documentos só são mantidos se o acesso for igual a “público”, caso contrário, eles serão removidos.

Crie documentos de amostra

```
db.articles.insertMany([
  { title: "Article A", access: "public", content: "Visible content" },
  { title: "Article B", access: "private", content: "Hidden content" }
]);
```

Exemplo de consulta

```
db.articles.aggregate([
  {
    $redact: {
      $cond: [
        { $eq: ["$access", "public"] },
        "$$KEEP",
        "$$PRUNE"
```

```
      ]
    }
  }
]);
```

Saída

```
[
  {
    "_id" : ObjectId("..."),
    "title" : "Article A",
    "access" : "public",
    "content" : "Visible content"
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o $$KEEP comando, escolha a guia do idioma que você deseja 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

A variável $$PRUNE do sistema é usada com o `$redact` estágio no pipeline de agregação para excluir documentos ou níveis de documentos incorporados dos resultados. Quando uma condição é avaliada como$$PRUNE, o documento ou subdocumento atual é removido da saída. Geralmente é usado com $$DESCEND (para manter e percorrer o documento) ou $$KEEP (para manter o documento em todos os níveis).

Parâmetros

Nenhum. A variável $$PRUNE do sistema é usada sem nenhum parâmetro e deve ser usada com`$redact`.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar $$PRUNE with `$redact` para excluir usuários com mais de 30 anos dos resultados.

Crie documentos de amostra

```
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" } }
])
```

Exemplo de consulta

```
db.users.aggregate([
  {
    $redact: {
      $cond: {
        if: { $gt: ["$age", 30] },
        then: "$$PRUNE",
        else: "$$DESCEND"
      }
    }
  }
```

```
])
```

Saída

```
[
  {
    "_id": 2,
    "name": "Saanvi Sarkar",
    "age": 28,
    "address": {
      "street": "456 Oak Rd",
      "city": "Someplace",
      "state": "NY"
    }
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o $$PRUNE comando, escolha a guia do idioma que você deseja 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()
```

$ ROOT

O $ROOT operador no Amazon DocumentDB é usado para referenciar todo o documento de entrada em um pipeline de agregação. Ele permite que você acesse e manipule o documento completo, incluindo todos os seus campos e estruturas aninhados.

Parâmetros

Nenhum

Exemplo (MongoDB Shell)

Este exemplo demonstra o uso $R00T para criar um registro de auditoria que captura o documento original completo junto com os metadados sobre quando ele foi processado.

Crie documentos de amostra

```
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
  }
]);
```

Exemplo de consulta

```
db.orders.aggregate([
  {
    $project: {
      processedAt: new Date(),
      originalDocument: "$$ROOT",
      summary: {
        $concat: [
          "Order ",
```

```
          "$orderId",
          " for ",
          "$customer",
          " - Total: $",
          { $toString: "$totalAmount" }
        ]
      }
    }
  }
]);
```

Saída

```
[
  {
    _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'
  }
```

```
]
```

Exemplos de código

Para ver um exemplo de código para usar o $ROOT comando, escolha a guia do idioma que você deseja 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

O `$abs` operador no Amazon DocumentDB retorna o valor absoluto de um número. Ele pode ser usado no pipeline de agregação para realizar operações matemáticas em campos numéricos.

Parâmetros

- `number`: a expressão numérica para a qual o valor absoluto será retornado.

Exemplo (MongoDB Shell)

Este exemplo demonstra o uso do `$abs` operador para encontrar o valor absoluto de um campo numérico.

Crie documentos de amostra

```
db.numbers.insertMany([
  { "_id": 1, "value": -5 },
  { "_id": 2, "value": 10 },
  { "_id": 3, "value": -3.14 },
  { "_id": 4, "value": 0 }
]);
```

Exemplo de consulta

```
db.numbers.aggregate([
  { $project: {
    "_id": 1,
    "absolute_value": { $abs: "$value" }
  }}
]);
```

Saída

```
[
  { "_id": 1, "absolute_value": 5 },
  { "_id": 2, "absolute_value": 10 },
  { "_id": 3, "absolute_value": 3.14 },
  { "_id": 4, "absolute_value": 0 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$abs` comando, escolha a guia do idioma que você deseja 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

Novo na versão 8.0.1.

O `$acos` operador no Amazon DocumentDB retorna o arco-seno (cosseno inverso) de um valor. O valor de entrada deve estar entre -1 e 1.

Parâmetros

- `expression`: uma expressão que resolve para um número entre -1 e 1.

O resultado está em radianos. Para obter graus, `$radiansToDegrees` aplique na saída.

O tipo de retorno é `double` por padrão. Se a entrada for decimal de 128 bits, a saída também será decimal de 128 bits.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$acos` operador para calcular o arco cosseno de valores numéricos.

Crie documentos de amostra

```
db.values.insertMany([
  { "_id": 1, "value": 1 },
  { "_id": 2, "value": 0 },
  { "_id": 3, "value": -1 }
]);
```

Exemplo de consulta

```
db.values.aggregate([
  { $project: {
    "angle": { $acos: "$value" }
  }}
```

```
]);
```

Saída

```
[
  { "_id": 1, "angle": 0 },
  { "_id": 2, "angle": 1.5707963267948966 },
  { "_id": 3, "angle": 3.141592653589793 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$acos` operador, escolha a guia do idioma que você deseja 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()
```

$acosh

Novo na versão 8.0.1.

O `$acosh` operador no Amazon DocumentDB retorna o cosseno hiperbólico inverso (arccosseno hiperbólico) de um valor. O valor de entrada deve ser maior ou igual a 1.

Parâmetros

- `expression`: uma expressão que resolve para um número maior ou igual a 1.

O tipo de retorno é `double` por padrão. Se a entrada for decimal de 128 bits, a saída também será decimal de 128 bits.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$acosh` operador para calcular o cosseno hiperbólico inverso de valores numéricos.

Crie documentos de amostra

```
db.values.insertMany([
```

```
  { "_id": 1, "value": 1 },
  { "_id": 2, "value": 2 },
  { "_id": 3, "value": 10 }
]);
```

Exemplo de consulta

```
db.values.aggregate([
  { $project: {
    "result": { $acosh: "$value" }
  }}
]);
```

Saída

```
[
  { "_id": 1, "result": 0 },
  { "_id": 2, "result": 1.3169578969248166 },
  { "_id": 3, "result": 2.993222846126381 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$acosh` operador, escolha a guia do idioma que você deseja 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

O `$add` operador no Amazon DocumentDB é usado para somar números ou datas. Ele pode ser usado para realizar operações aritméticas em campos numéricos ou para realizar aritmética de data adicionando várias unidades de tempo a um campo de data.

Parâmetros

- `expression1`: o primeiro número ou data a ser adicionado.
- `expression2`: o segundo número ou data a ser adicionado.
- `expression3`: (opcional) Números ou datas adicionais a serem adicionados.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$add` operador para somar dois números.

Crie documentos de amostra

```
db.numbers.insertMany([
  { _id: 1, a: 5, b: 3 },
  { _id: 2, a: 10, b: 7 },
  { _id: 3, a: 2, b: 8 }
]);
```

Exemplo de consulta

```
db.numbers.aggregate([
  { $project: {
    _id: 1,
    sum: { $add: ["$a", "$b"] }
  }}
])
```

Saída

```
[
  { "_id" : 1, "sum" : 8 },
  { "_id" : 2, "sum" : 17 },
  { "_id" : 3, "sum" : 10 }
]
```

Neste exemplo, o `$add` operador é usado para adicionar os valores dos b campos a e para cada documento e armazenar o resultado no `sum` campo.

Exemplos de código

Para ver um exemplo de código para usar o `$add` comando, escolha a guia do idioma que você deseja 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()
```

$adicionar ToSet

O operador `$addToSet` de agregação retorna uma matriz de valores exclusivos de uma expressão especificada para cada grupo. Ele é usado dentro do `$group` estágio para acumular valores distintos, eliminando automaticamente as duplicatas.

Parâmetros

- `expression`: A expressão a ser avaliada para cada documento no grupo.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$addToSet` operador para coletar cidades exclusivas nas quais os pedidos foram feitos para cada cliente.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.orders.aggregate([
  {
    $group: {
      _id: "$customer",
      cities: { $addToSet: "$city" }
    }
  }
]);
```

Saída

```
[
  { _id: 'Bob', cities: [ 'Seattle', 'Boston' ] },
  { _id: 'Alice', cities: [ 'Seattle', 'Portland' ] }
```

```
]
```

Exemplo de uso do operador de janela (MongoDB Shell)

Novo na versão 8.0.2.

O `$addToSet` operador também pode ser usado como operador de janela no `$setWindowFields` palco. Nesse contexto, ele retorna uma matriz dos valores exclusivos da expressão especificada para os documentos em cada janela. Você especifica o operador no `output` campo e, opcionalmente, define os limites da janela com um `window` documento.

> **Note**
>
> Quando usado como operador de janela em`$setWindowFields`, `$addToSet` é limitado a 100 MB de dados intermediários. Uma operação que excede esse limite retorna um erro.

Crie documentos de amostra

```
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" }
]);
```

Exemplo de consulta

O exemplo a seguir particiona os documentos por`customer`, classifica cada partição por `day` e retorna o conjunto de `city` valores exclusivos desde o início da partição até o documento atual.

```
db.visits.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$customer",
      sortBy: { day: 1 },
      output: {
        citiesSoFar: {
          $addToSet: "$city",
          window: { documents: ["unbounded", "current"] }
        }
```

```
      }
    }
  }
]);
```

Saída

```
[
  { "_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 é aumentado com `citiesSoFar` o conjunto de `city` valores exclusivos em sua partição, desde o início até o documento atual. Assim como no formulário acumulador, a ordem dos valores na matriz retornada não é garantida.

Exemplos de código

Para ver um exemplo de código para usar o `$addToSet` operador, escolha a guia do idioma que você deseja usar. Os exemplos a seguir mostram o uso do acumulador (em`$group`) e o uso do operador de janela (em`$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

O `$addFields` estágio no pipeline de agregação do Amazon DocumentDB permite que você adicione novos campos computados aos documentos. Isso pode ser útil para adicionar dados derivados ou transformados aos documentos.

Parâmetros

- `newField`: o nome do novo campo a ser adicionado.
- `expression`: uma expressão que se resolve com o valor do novo campo.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar `$addFields` para adicionar um novo campo `TotalInventory` que calcula o estoque total com base nos campos `Inventory.OnHand` e. `Inventory.OrderQnty`

Crie documentos de amostra

```
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
  }
]);
```

Exemplo de consulta

```
db.example.aggregate([
  {
    $addFields: {
      TotalInventory: { $add: ["$Inventory.OnHand", "$Inventory.OrderQnty"] }
    }
  }
])
```

Saída

```
[
  {
    "_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
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$addFields` comando, escolha a guia do idioma que você deseja 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()
```

$tudo ElementsTrue

Novo a partir da versão 4.0

O `$allElementsTrue` operador é usado para verificar se todos os elementos em uma matriz são avaliados como um valor verdadeiro.

Parâmetros

- `expression`: uma expressão que é avaliada como uma matriz.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso de `$allElementsTrue` para verificar se todos os elementos em uma matriz são verdadeiros.

Crie documentos de amostra

```
db.collection.insert([
  { "name": "John", "scores": [100, 90, 80] },
  { "name": "Jane", "scores": [80, 85, 0] },
  { "name": "Bob", "scores": [90, 95, null] }
```

```
])
```

Exemplo de consulta

```
db.collection.find({
  "scores": { "$allElementsTrue": [{ "$gt": 0 }] }
})
```

Saída

```
[
  { "_id" : ObjectId("..."), "name" : "John", "scores" : [ 100, 90, 80 ] },
  { "_id" : ObjectId("..."), "name" : "Bob", "scores" : [ 90, 95, null ] }
]
```

Neste exemplo, a consulta verifica se todos os elementos na `scores` matriz são maiores que 0. O documento com `"name": "Jane"` é excluído porque a `scores` matriz contém um 0, que é um valor falso.

Exemplos de código

Para ver um exemplo de código para usar o `$allElementsTrue` comando, escolha a guia do idioma que você deseja 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

O operador $and de agregação avalia várias expressões e retorna `true` somente se todas as expressões forem avaliadas como. `true` Se alguma expressão for`false`, ela retornará`false`.

Parâmetros

- `expressions`: uma matriz de expressões para avaliar.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do $and operador para verificar se os produtos atendem a vários critérios.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.products.aggregate([
  {
    $project: {
      name: 1,
      price: 1,
      inStock: 1,
      affordable: {
        $and: [
          { $lt: ["$price", 100] },
          { $eq: ["$inStock", true] }
        ]
      }
    }
  }
]);
```

Saída

```
[
  { _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 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o operador de $and agregação, escolha a guia do idioma que você deseja 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()
```

$ qualquer ElementTrue

Novo da versão 4.0

O `$anyElementTrue` operador é usado para determinar se algum elemento em uma matriz é verdadeiro.

Parâmetros

- `field`: um campo de matriz para avaliar.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso de `$anyElementTrue` para verificar se algum elemento em uma matriz é verdadeiro.

Crie documentos de amostra

```
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] }
])
```

Exemplo de consulta

```
db.grades.aggregate([
  {
    $project: {
      student: 1,
      isAnyTrue: { $anyElementTrue: ["$scores"] },
      _id: 0
    }
  }
])
```

Saída

```
[
  { student: 'Tim', isAnyTrue: false },
  { student: 'Bob', isAnyTrue: false },
  { student: 'Ivy', isAnyTrue: true }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$anyElementTrue` comando, escolha a guia do idioma que você deseja 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

O `$arrayElemAt` operador no Amazon DocumentDB permite que você recupere um elemento de uma matriz pela posição do índice. Isso é particularmente útil quando você precisa acessar um elemento específico em um campo de matriz em seus documentos.

Parâmetros

- `array`: a matriz de entrada da qual um elemento deve ser recuperado.
- `index`: a posição do índice com base em zero do elemento a ser recuperado. Esse valor deve ser um número inteiro não negativo.

Exemplo (MongoDB Shell)

Neste exemplo, demonstraremos como usar o `$arrayElemAt` operador para recuperar elementos específicos da `flight_miles` matriz na `miles` coleção.

Crie documentos de amostra

```
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 ]}
]);
```

Exemplo de consulta

```
db.miles.aggregate([
  { $project: {
    "_id": 1,
    "first_mile": { $arrayElemAt: [ "$flight_miles", 0 ] },
    "last_mile": { $arrayElemAt: [ "$flight_miles", -1 ] }
  }}
]);
```

Saída

```
{ "_id" : 1, "first_mile" : 1205, "last_mile" : 880 }
{ "_id" : 2, "first_mile" : 1205, "last_mile" : 2780 }
{ "_id" : 3, "first_mile" : 1205, "last_mile" : 880 }
```

Neste exemplo, usamos o `$arrayElemAt` operador para recuperar o primeiro e o último elementos da `flight_miles` matriz para cada documento.

Exemplos de código

Para ver um exemplo de código para usar o `$arrayElemAt` comando, escolha a guia do idioma que você deseja 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

O `$arrayTo0bject` operador no Amazon DocumentDB é o inverso do `$objectToArray` operador. Ele pega uma matriz de documentos de pares de valores-chave e os converte em um único documento. Isso é particularmente útil quando você precisa converter uma matriz de pares de valores-chave novamente em uma estrutura de objeto ou documento.

Parâmetros

- `array expression`: uma expressão que se resolve em uma matriz. Os elementos da matriz devem ser documentos com dois campos: k (a chave) e v (o valor).

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar `$arrayTo0bject` para converter uma matriz de pares de valores-chave novamente em um documento.

Crie documentos de amostra

```
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 } }
]);
```

Exemplo de consulta

```
db.videos.aggregate([
  { $project: { name: 1, videos: { $objectToArray: "$inventory" } } },
  { $project: { name: 1, inventory: { $arrayToObject: "$videos" } } }
]);
```

Saída

```
{ "_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 } }
```

Neste exemplo, o `$objectToArray` operador é usado para converter o `inventory` objeto em uma matriz de pares de valores-chave. O `$arrayToObject` operador é então usado para converter a matriz novamente em um documento, restaurando a estrutura original do objeto.

Exemplos de código

Para ver um exemplo de código para usar o `$arrayToObject` comando, escolha a guia do idioma que você deseja 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

Novo na versão 8.0.1.

O `$asin` operador no Amazon DocumentDB retorna o arco seno (seno inverso) de um valor. O valor de entrada deve estar entre -1 e 1.

Parâmetros

- `expression`: uma expressão que resolve para um número entre -1 e 1.

O resultado está em radianos. Para obter graus, `$radiansToDegrees` aplique na saída.

O tipo de retorno é `double` por padrão. Se a entrada for decimal de 128 bits, a saída também será decimal de 128 bits.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$asin` operador para calcular o arco seno de valores numéricos.

Crie documentos de amostra

```
db.values.insertMany([
  { "_id": 1, "value": 0 },
  { "_id": 2, "value": 0.5 },
  { "_id": 3, "value": 1 }
]);
```

Exemplo de consulta

```
db.values.aggregate([
  { $project: {
    "angle": { $asin: "$value" }
  }}
]);
```

Saída

```
[
  { "_id": 1, "angle": 0 },
  { "_id": 2, "angle": 0.5235987755982989 },
  { "_id": 3, "angle": 1.5707963267948966 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$asin` operador, escolha a guia do idioma que você deseja 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()
```

$asinh

Novo na versão 8.0.1.

O `$asinh` operador no Amazon DocumentDB retorna o seno hiperbólico inverso de um valor. Use-o no pipeline de agregação para realizar cálculos trigonométricos hiperbólicos em campos numéricos.

Parâmetros

- `expression`: Uma expressão que se resolve em um número.

O resultado está em radianos. Para obter graus, `$radiansToDegrees` aplique na saída.

O tipo de retorno é `double` por padrão. Se a entrada for decimal de 128 bits, a saída também será decimal de 128 bits.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$asinh` operador para calcular o seno hiperbólico inverso dos valores numéricos.

Crie documentos de amostra

```
db.values.insertMany([
  { "_id": 1, "value": 0 },
  { "_id": 2, "value": 1 },
  { "_id": 3, "value": -1 }
]);
```

Exemplo de consulta

```
db.values.aggregate([
  { $project: {
    "result": { $asinh: "$value" }
  }}
]);
```

Saída

```
[
  { "_id": 1, "result": 0 },
  { "_id": 2, "result": 0.881373587019543 },
  { "_id": 3, "result": -0.881373587019543 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$asinh` operador, escolha a guia do idioma que você deseja 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()
```

$satan

Novo na versão 8.0.1.

O `$atan` operador no Amazon DocumentDB retorna a tangente inversa (arco tangente) de um valor.

A expressão de entrada deve ser resolvida com um valor numérico.

O resultado está em radianos. Para obter graus, `$radiansToDegrees` aplique na saída.

O tipo de retorno é `double` por padrão. Se a entrada for decimal de 128 bits, a saída também será decimal de 128 bits.

Parâmetros

- `expression`: uma expressão que se resolve em um número.

Comportamento

null e NaN

Exemplo	Resultados
`{ $atan: NaN }`	`NaN`
`{ $atan: null }`	`null`

Quando a entrada está `null` ou o campo referenciado está ausente, o resultado é`null`. Uma entrada de `NaN` produtos`NaN`.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$atan` operador para calcular o arco tangente de um valor.

Crie documentos de amostra

```
db.values.insertMany([
  { "_id": 1, "value": 0 },
  { "_id": 2, "value": 1 },
  { "_id": 3, "value": -1 }
]);
```

Exemplo de consulta

```
db.values.aggregate([
  { $project: {
    "angle": { $atan: "$value" }
  }}
]);
```

Saída

```
[
  { "_id": 1, "angle": 0 },
  { "_id": 2, "angle": 0.7853981633974483 },
  { "_id": 3, "angle": -0.7853981633974483 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$atan` operador, escolha a guia do idioma que você deseja 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

Novo na versão 8.0.1.

O `$atan2` operador no Amazon DocumentDB retorna a tangente inversa (tangente do arco) dey / x, onde y e x são o primeiro e o segundo valores passados para a expressão, respectivamente.

Ambas as expressões devem ser resolvidas em valores numéricos. O operador usa o sinal de ambos os argumentos para determinar o quadrante correto.

O resultado está em radianos. Para obter graus, `$radiansToDegrees` aplique na saída.

O tipo de retorno é `double` por padrão. Se as entradas forem decimais de 128 bits, a saída também será decimal de 128 bits.

Parâmetros

- `expression 1`(y): Uma expressão que se resolve em um número representando a coordenada y.
- `expression 2`(x): Uma expressão que se resolve em um número representando a coordenada x.

Comportamento

null e NaN

Se um dos argumentos for `null` (ou fizer referência a um campo ausente), o resultado será`null`. Se um dos argumentos for`NaN`, o resultado será`NaN`. Quando um argumento é `null` e o outro é`NaN`, `null` tem precedência.

Exemplo	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`

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$atan2` operador para calcular o arco tangente de y/x para um conjunto de pontos.

Crie documentos de amostra

```
db.points.insertMany([
```

```
  { "_id": 1, "y": 1, "x": 1 },
  { "_id": 2, "y": 1, "x": 0 },
  { "_id": 3, "y": -1, "x": -1 }
]);
```

Exemplo de consulta

```
db.points.aggregate([
  { $project: {
    "angle": { $atan2: ["$y", "$x"] }
  }}
]);
```

Saída

```
[
  { "_id": 1, "angle": 0.7853981633974483 },
  { "_id": 2, "angle": 1.5707963267948966 },
  { "_id": 3, "angle": -2.356194490192345 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$atan2` operador, escolha a guia do idioma que você deseja 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()
```

$atanah

Novo na versão 8.0.1.

O `$atanh` operador no Amazon DocumentDB retorna a tangente hiperbólica inversa (tangente do arco hiperbólico) de um valor.

A expressão de entrada deve ser resolvida para um número no intervalo de `-1` até `1` (inclusive). Valores fora desse intervalo causam um erro.

O tipo de retorno é `double` por padrão. Se a entrada for decimal de 128 bits, a saída também será decimal de 128 bits.

Parâmetros

- `expression`: uma expressão que se resolve em um número entre `-1` e`1`, inclusive.

Comportamento

null, NaN e +/- Infinito

Exemplo	Resultados
`{ $atanh: NaN }`	`NaN`
`{ $atanh: null }`	`null`
`{ $atanh: 1 }`	`Infinity`
`{ $atanh: -1 }`	`-Infinity`
`{ $atanh: Infinity }` ou `{ $atanh: -Infinity }`	Lança um erro.

Quando a entrada está `null` ou o campo referenciado está ausente, o resultado é`null`. Uma entrada de `NaN` produtos`NaN`. Os valores limite `1` e `-1` produzem `Infinity` e, `-Infinity` respectivamente. Qualquer valor fora do `[-1, 1]` intervalo (inclusive`±Infinity`) causa um erro.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$atanh` operador para calcular a tangente hiperbólica inversa de um valor.

Crie documentos de amostra

```
db.values.insertMany([
  { "_id": 1, "value": 0 },
  { "_id": 2, "value": 0.5 },
  { "_id": 3, "value": -0.5 }
]);
```

Exemplo de consulta

```
db.values.aggregate([
  { $project: {
    "result": { $atanh: "$value" }
  }}
]);
```

Saída

```
[
  { "_id": 1, "result": 0 },
  { "_id": 2, "result": 0.5493061443340548 },
  { "_id": 3, "result": -0.5493061443340548 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$atanh` operador, escolha a guia do idioma que você deseja 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

O operador `$avg` de agregação no Amazon DocumentDB calcula o valor médio da expressão especificada nos documentos que são inseridos no palco. Esse operador é útil para calcular a média de um campo numérico ou expressão em um conjunto de documentos.

Parâmetros

- `expression`: A expressão a ser usada para calcular a média. Isso pode ser um caminho de campo (por exemplo`"$field"`) ou uma expressão (por exemplo`{ $multiply: ["$field1", "$field2"] }`).

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$avg` operador para calcular a pontuação média em um conjunto de documentos estudantis.

Crie documentos de amostra

```
db.students.insertMany([
  { name: "John", score: 85 },
  { name: "Jane", score: 92 },
  { name: "Bob", score: 78 },
  { name: "Alice", score: 90 }
]);
```

Exemplo de consulta

```
db.students.aggregate([
  { $group: {
    _id: null,
    avgScore: { $avg: "$score" }
  }}
]);
```

Saída

```
[
  {
    "_id": null,
    "avgScore": 86.25
  }
]
```

Exemplo de uso do operador de janela (MongoDB Shell)

Novo na versão 8.0.2.

O `$avg` operador também pode ser usado como operador de janela no `$setWindowFields` palco. Nesse contexto, ele retorna a média da expressão especificada para os documentos em cada janela. Você especifica o operador no `output` campo e, opcionalmente, define os limites da janela com um `window` documento.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

O exemplo a seguir particiona os documentos por`city`, classifica cada partição por `day` e retorna uma média de execução `temp` do início da partição até o documento atual.

```
db.readings.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$city",
      sortBy: { day: 1 },
      output: {
        runningAvg: {
          $avg: "$temp",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Saída

```
[
  { "_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 é aumentado com `runningAvg` a média de `temp` dentro de sua partição até e incluindo o documento atual.

Exemplos de código

Para ver um exemplo de código para usar o `$avg` operador, escolha a guia do idioma que você deseja usar. Os exemplos a seguir mostram o uso do acumulador (em`$group`) e o uso do operador de janela (em`$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

Novo na versão 8.0.1.

O `$binarySize` operador no Amazon DocumentDB retorna o tamanho em bytes de uma determinada string ou valor de dados binários. Para valores de string, essa é a contagem de bytes UTF-8 codificados, não a contagem de caracteres. Multi-byte UTF-8 caracteres (como caracteres acentuados ou caracteres CJK) produzirão uma contagem de bytes maior do que o número de caracteres na string.

Parâmetros

- `expression`: uma expressão que se resolve em uma string ou valor de dados binários.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$binarySize` operador para retornar o tamanho do byte dos campos de string.

Crie documentos de amostra

```
db.docs.insertMany([
  {_id: 1, content: "hello"},
  {_id: 2, content: "Amazon DocumentDB"},
  {_id: 3, content: ""}
]);
```

Exemplo de consulta

```
db.docs.aggregate([
  { $project: { size: { $binarySize: "$content" } } }
]);
```

Saída

```
[
  {_id: 1, size: 5},
  {_id: 2, size: 17},
  {_id: 3, size: 0}
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$binarySize` operador, escolha a guia do idioma que você deseja 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 e

Novo na versão 8.0.1.

O `$bitAnd` operador no Amazon DocumentDB executa uma operação AND bit a bit em valores inteiros ou longos.

Parâmetros

- `expressions`: uma matriz de duas ou mais expressões, que podem ser resolvidas em números inteiros ou longos.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$bitAnd` operador para executar AND bit a bit em dois campos.

Crie documentos de amostra

```
db.flags.insertMany([
  {_id: 1, a: 13, b: 10},
  {_id: 2, a: 7, b: 5},
  {_id: 3, a: 15, b: 9}
]);
```

Exemplo de consulta

```
db.flags.aggregate([
  { $project: { result: { $bitAnd: ["$a", "$b"] } } }
]);
```

Saída

```
[
  {_id: 1, result: 8},
  {_id: 2, result: 5},
  {_id: 3, result: 9}
]
```

Em binário: 13 (1101) E 10 (1010) = 8 (1000); 7 (0111) E 5 (0101) = 5 (0101); 15 (1111) E 9 (1001) = 9 (1001).

Exemplos de código

Para ver um exemplo de código para usar o `$bitAnd` operador, escolha a guia do idioma que você deseja 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: { $bitAnd: ["$a", "$b"] } } }
    ]).toArray();
    console.log(result);
  } finally {
    await client.close();
  }
}
example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    try:
        db = client['test']
        collection = db['flags']
        result = list(collection.aggregate([
            {'$project': {'result': {'$bitAnd': ['$a', '$b']}}}
        ]))
        print(result)
    finally:
        client.close()

example()
```

$bitNot

Novo na versão 8.0.1.

O `$bitNot` operador no Amazon DocumentDB executa uma operação NOT bit a bit em um valor inteiro ou longo, retornando o complemento bit a bit. Para um determinado inteiro ou n longo, o resultado é - (n+1).

Parâmetros

- `expression`: uma expressão que é resolvida para um número inteiro ou longo. Retorna o complemento bit a bit, que é - (n+1) para um determinado inteiro ou n longo.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$bitNot` operador para calcular o complemento bit a bit de valores inteiros.

Crie documentos de amostra

```
db.numbers.insertMany([
  {_id: 1, value: 0},
  {_id: 2, value: 5},
  {_id: 3, value: -3}
]);
```

Exemplo de consulta

```
db.numbers.aggregate([
  { $project: { result: { $bitNot: "$value" } } }
]);
```

Saída

```
[
  {_id: 1, result: -1},
  {_id: 2, result: -6},
  {_id: 3, result: 2}
]
```

Em binário: NÃO 0 = -1; NÃO 5 = -6; NÃO -3 = 2.

Exemplos de código

Para ver um exemplo de código para usar o `$bitNot` operador, escolha a guia do idioma que você deseja 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 ou

Novo na versão 8.0.1.

O `$bitOr` operador no Amazon DocumentDB executa uma operação OR bit a bit em valores inteiros ou longos.

Parâmetros

- `expressions`: uma matriz de duas ou mais expressões, que podem ser resolvidas em números inteiros ou longos.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$bitOr` operador para realizar OR bit a bit em dois campos.

Crie documentos de amostra

```
db.flags.insertMany([
  {_id: 1, a: 13, b: 10},
  {_id: 2, a: 7, b: 5},
  {_id: 3, a: 15, b: 9}
]);
```

Exemplo de consulta

```
db.flags.aggregate([
  { $project: { result: { $bitOr: ["$a", "$b"] } } }
]);
```

Saída

```
[
  {_id: 1, result: 15},
  {_id: 2, result: 7},
```

```
  {_id: 3, result: 15}
]
```

Em binário: 13 (1101) OR 10 (1010) = 15 (1111); 7 (0111) OR 5 (0101) = 7 (0111); 15 (1111) OR 9 (1001) = 15 (1111).

Exemplos de código

Para ver um exemplo de código para usar o `$bitOr` operador, escolha a guia do idioma que você deseja 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 ou

Novo na versão 8.0.1.

O `$bitXor` operador no Amazon DocumentDB executa uma operação XOR bit a bit em valores inteiros ou longos.

Parâmetros

- `expressions`: uma matriz de duas ou mais expressões, que podem ser resolvidas em números inteiros ou longos.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$bitXor` operador para realizar XOR bit a bit em dois campos.

Crie documentos de amostra

```
db.flags.insertMany([
  {_id: 1, a: 13, b: 10},
  {_id: 2, a: 7, b: 5},
  {_id: 3, a: 15, b: 9}
]);
```

Exemplo de consulta

```
db.flags.aggregate([
  { $project: { result: { $bitXor: ["$a", "$b"] } } }
]);
```

Saída

```
[
  {_id: 1, result: 7},
  {_id: 2, result: 2},
  {_id: 3, result: 6}
]
```

Em binário: 13 (1101) XOR 10 (1010) = 7 (0111); 7 (0111) XOR 5 (0101) = 2 (0010); 15 (1111) XOR 9 (1001) = 6 (0110).

Exemplos de código

Para ver um exemplo de código para usar o `$bitXor` operador, escolha a guia do idioma que você deseja 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: { $bitXor: ["$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()
```

$ fundo

Novo na versão 8.0.1.

Use o `$bottom` acumulador no `$group` estágio para retornar o documento com a classificação mais baixa por grupo, de acordo com uma ordem de classificação especificada.

Parâmetros

- `sortBy`: um documento especificando a ordem de classificação. Use 1 para ascender ou `-1` descer.
- `output`: uma expressão que especifica os campos a serem retornados do documento inferior.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$bottom` acumulador para encontrar a menor venda (menor quantidade) por item em uma coleção de vendas.

Crie documentos de amostra

```
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 }
])
```

Exemplo de consulta

```
db.sales.aggregate([
  { $group: { _id: "$item", bottomSale: { $bottom: { sortBy: { quantity: -1 }, output:
 { quantity: "$quantity", price: "$price" } } } } }
])
```

Saída

```
[
  { "_id": "xyz", "bottomSale": { "quantity": 7, "price": 6 } },
  { "_id": "abc", "bottomSale": { "quantity": 5, "price": 8 } }
]
```

Exemplo de uso do operador de janela (MongoDB Shell)

Novo na versão 8.0.2.

O `$bottom` operador também pode ser usado como operador de janela no `$setWindowFields` palco. Nesse contexto, ele retorna o documento `output` da classificação inferior (de acordo com o próprio operador`sortBy`) entre os documentos em cada janela. Você especifica o operador no `output` campo e, opcionalmente, define os limites da janela com um `window` documento.

> **Note**
>
> Quando usado como operador de janela em`$setWindowFields`, `$bottom` é limitado a 100 MB de dados intermediários. Uma operação que excede esse limite retorna um erro.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

O exemplo a seguir particiona os documentos por`player`, classifica cada partição por `round` e retorna a rodada de menor pontuação vista desde o início da partição até o documento atual.

```
db.matchScores.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$player",
      sortBy: { round: 1 },
      output: {
        worstRoundSoFar: {
          $bottom: { sortBy: { score: -1 }, output: { round: "$round", score:
 "$score" } },
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Saída

```
[
  { "_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 é aumentado com `worstRoundSoFar` a rodada de menor pontuação em sua partição, incluindo o documento atual.

Exemplos de código

Para ver um exemplo de código para usar o `$bottom` operador, escolha a guia do idioma que você deseja usar. Os exemplos a seguir mostram o uso do acumulador (em`$group`) e o uso do operador de janela (em`$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()
```

$bottomN

Novo na versão 8.0.1.

Use o `$bottomN` acumulador no `$group` estágio para retornar os N elementos inferiores dentro de um grupo de acordo com uma ordem de classificação especificada. Se um grupo contiver menos de N elementos, `$bottomN` retornará todos os elementos do grupo.

Parâmetros

- `n`: um número inteiro positivo ou uma expressão que se resolve em um, especificando quantos resultados inferiores devem ser retornados por grupo.
- `sortBy`: um documento especificando a ordem de classificação. Use `1` para ascender ou `-1` descer.
- `output`: uma expressão que especifica os campos a serem retornados de cada um dos N documentos inferiores.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$bottomN` acumulador para encontrar as duas últimas vendas (menor quantidade) por item em uma coleção de vendas.

Crie documentos de amostra

```
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 }
])
```

Exemplo de consulta

```
db.sales.aggregate([
  { $group: { _id: "$item", bottomTwoSales: { $bottomN: { n: 2, sortBy: { quantity:
 -1 }, output: { quantity: "$quantity", price: "$price" } } } } }
])
```

Saída

```
[
  { "_id": "xyz", "bottomTwoSales": [{ "quantity": 9, "price": 6 }, { "quantity": 3,
 "price": 12 }] },
  { "_id": "abc", "bottomTwoSales": [{ "quantity": 7, "price": 8 }, { "quantity": 5,
 "price": 10 }] }
]
```

Exemplo de uso do operador de janela (MongoDB Shell)

Novo na versão 8.0.2.

O `$bottomN` operador também pode ser usado como operador de janela no `$setWindowFields` palco. Nesse contexto, ele retorna uma matriz dos N documentos `output` da parte inferior (de acordo com o próprio operador`sortBy`) entre os documentos em cada janela. Você especifica o operador no `output` campo e, opcionalmente, define os limites da janela com um `window` documento.

> **Note**
>
> Quando usado como operador de janela em`$setWindowFields`, `$bottomN` é limitado a 100 MB de dados intermediários. Uma operação que excede esse limite retorna um erro.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

O exemplo a seguir particiona os documentos por`player`, classifica cada partição por `round` e retorna as duas rodadas de menor pontuação vistas desde o início da partição até o documento atual.

```
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"] }
        }
      }
    }
  }
]);
```

Saída

```
[
  { "_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 é aumentado com `bottomTwoRoundsSoFar` as duas rodadas de menor pontuação dentro de sua partição, incluindo o documento atual. Se a janela contiver menos de dois documentos, todos eles serão retornados.

Exemplos de código

Para ver um exemplo de código para usar o `$bottomN` operador, escolha a guia do idioma que você deseja usar. Os exemplos a seguir mostram o uso do acumulador (em`$group`) e o uso do operador de janela (em`$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

Novo na versão 8.0.1.

O `$bsonSize` operador no Amazon DocumentDB retorna o tamanho em bytes de um documento quando codificado como BSON.

Parâmetros

- `expression`: uma expressão que se resolve em um documento ou objeto.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$bsonSize` operador para retornar o tamanho BSON de cada documento.

Crie documentos de amostra

```
db.items.insertMany([
  {_id: 1, name: "a", tags: ["x"]},
  {_id: 2, name: "ab", tags: ["x", "y", "z"]},
  {_id: 3, name: "abc", tags: []}
]);
```

Exemplo de consulta

```
db.items.aggregate([
  { $project: { docSize: { $bsonSize: "$$ROOT" } } }
]);
```

Saída

```
[
  {_id: 1, docSize: 46},
  {_id: 2, docSize: 65},
  {_id: 3, docSize: 39}
]
```

> **Note**
>
> Os tamanhos exatos dependem da codificação BSON dos campos do documento.

Exemplos de código

Para ver um exemplo de código para usar o `$bsonSize` operador, escolha a guia do idioma que você deseja 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('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

Novo a partir da versão 8.0

Não é compatível com o cluster elástico.

O estágio `$bucket` de agregação no Amazon DocumentDB permite agrupar documentos de entrada em buckets com base em uma expressão e limites de bucket especificados. Isso pode ser útil para analisar dados que se enquadram em determinadas faixas de valores ou categorias.

Parâmetros

- `groupBy`(obrigatório): a expressão que especifica o valor pelo qual agrupar.
- `boundaries`(obrigatório): uma matriz de valores duplos que define os limites do intervalo. Os documentos são atribuídos a compartimentos com base no valor da `groupBy` expressão que está dentro dos limites especificados.
- `default`(opcional): um valor literal que é gerado para documentos cujo valor de `groupBy` expressão não se enquadra em nenhum dos limites especificados.
- `output`(opcional): um objeto que especifica as informações a serem enviadas para cada bucket. Você pode usar operadores de acumulador como`$sum`, `$avg`$min, e `$max` para calcular agregações para cada bucket.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$bucket` estágio para agrupar dados de vendas por faixa de preço.

Crie documentos de amostra

```
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") }
]);
```

Exemplo 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 }
  }
])
```

Saída

```
[
  { _id: 0, count: 1, totalQuantity: 3 },
  { _id: 10, count: 2, totalQuantity: 4 },
  { _id: 20, count: 2, totalQuantity: 2 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$bucket` comando, escolha a guia do idioma que você deseja 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

Novo na versão 8.0.2.

O estágio `$bucketAuto` de agregação no Amazon DocumentDB agrupa automaticamente os documentos de entrada em um número específico de buckets com base em uma expressão especificada. Ao `$bucket` contrário de quando você define explicitamente os limites do intervalo, determina `$bucketAuto` automaticamente os limites para que os documentos sejam distribuídos da forma mais uniforme possível pelo número solicitado de compartimentos. Isso pode ser útil quando você deseja categorizar os dados em um número fixo de grupos sem conhecer a distribuição dos valores com antecedência.

Parâmetros

- `groupBy`(obrigatório): a expressão que especifica o valor pelo qual agrupar.

- `buckets`(obrigatório): um número inteiro positivo que especifica o número de compartimentos nos quais os documentos são agrupados. Dependendo dos documentos de entrada e da `groupBy` expressão, o número real de compartimentos pode ser menor que o número especificado.
- `output`(opcional): um objeto que especifica as informações a serem produzidas para cada bucket. Você pode usar operadores de acumulador como`$sum`, `$avg``$min`, e `$max` para calcular agregações para cada bucket. Se omitido, cada bucket gera um `count` campo por padrão.
- `granularity`(opcional): uma string que especifica a série numérica preferida a ser usada para os limites do intervalo. Isso ajusta os limites computados a uma progressão numérica padrão. Esse parâmetro estará disponível somente se todos os `groupBy` valores forem numéricos e nenhum deles for`NaN`. Os valores válidos são: `R5` `R10` `R20` `R40``R80`,`1-2-5`,`E6`,`E12`,`E24`, `E48``E96`,`E192`,, `POWERSOF2` e.

Saída

Cada bucket é retornado como um documento separado que contém um `_id` objeto descrevendo os limites do bucket:

- `_id.min`: O limite inferior inclusivo do balde.
- `_id.max`: O limite superior do balde. Esse limite é exclusivo para cada compartimento, exceto o último compartimento da série, onde é inclusivo.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$bucketAuto` estágio para agrupar dados de vendas em grupos por preço.

Crie documentos de amostra

```
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") }
]);
```

Exemplo de consulta

O comando a seguir agrupa os documentos de amostra em três grupos por preço:

```
db.sales.aggregate([
  {
    $bucketAuto: {
      groupBy: "$price",
      buckets: 3,
      output: {
        "count": { $sum: 1 },
        "totalQuantity": { $sum: "$quantity" }
      }
    }
  }
])
```

Saída

O comando retorna a seguinte saída:

```
[
  { _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 }
]
```

Exemplo de consulta com granularidade

O exemplo a seguir adiciona a `granularity` opção de ajustar os limites do bucket computado às potências de 2.

```
db.sales.aggregate([
  {
    $bucketAuto: {
      groupBy: "$price",
      buckets: 2,
      granularity: "POWERSOF2",
      output: {
        "count": { $sum: 1 },
        "totalQuantity": { $sum: "$quantity" }
      }
    }
  }
```

```
])
```

Saída

O comando retorna a seguinte saída:

```
[
  { _id: { min: 4, max: 16 }, count: 3, totalQuantity: 7 },
  { _id: { min: 16, max: 32 }, count: 2, totalQuantity: 2 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$bucketAuto` comando, escolha a guia do idioma que você deseja 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

Novo a partir da versão 4.0

O `$ceil` operador no Amazon DocumentDB, assim como no MongoDB, arredonda um número até o número inteiro mais próximo. Isso é útil quando você precisa realizar operações matemáticas em campos numéricos e garantir que o resultado seja um número inteiro.

Parâmetros

- `expression`: A expressão numérica a ser arredondada para cima.

Exemplo (MongoDB Shell)

Este exemplo demonstra como usar o `$ceil` operador para arredondar um campo numérico.

Crie documentos de amostra

```
db.numbers.insertMany([
  { "_id": 1, "value": 3.14 },
  { "_id": 2, "value": -2.7 },
  { "_id": 3, "value": 0 }
])
```

Exemplo de consulta

```
db.numbers.aggregate([
  { $project: {
    "roundedUp": { $ceil: "$value" }
  }}
])
```

Saída

```
{ "_id": 1, "roundedUp": 4 }
{ "_id": 2, "roundedUp": -2 }
{ "_id": 3, "roundedUp": 0 }
```

Exemplos de código

Para ver um exemplo de código para usar o `$ceil` comando, escolha a guia do idioma que você deseja 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

Não é suportado pelo cluster elástico.

O estágio `$changeStream` de agregação abre um cursor de fluxo de alterações para monitorar as alterações em tempo real em uma coleção. Ele retorna documentos de eventos de alteração quando ocorrem operações de inserção, atualização, substituição ou exclusão.

Parâmetros

- `fullDocument`: especifica se o documento completo deve ser devolvido para operações de atualização. As opções são `default` ou `updateLookup`.
- `resumeAfter`: opcional. Retomar o token para continuar a partir de um ponto específico no fluxo de alterações.
- `startAtOperationTime`: opcional. Carimbo de data/hora a partir do qual iniciar o fluxo de alterações.
- `allChangesForCluster`: opcional. Valor booleano. Quando`true`, observa todas as alterações no cluster (para banco de dados administrativo). Quando `false` (padrão), observa somente a coleção especificada.
- `showExpandedEvents`: opcional. Valor booleano. Quando`true`, inclui notificações de eventos de linguagem de definição de dados (DDL) expandida, como `createCollection` `ecreateIndexes`, no fluxo de alterações. Quando `false` (padrão), esses eventos não são retornados. Disponível no Amazon DocumentDB 8.0 a partir da versão 8.0.2 do mecanismo.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$changeStream` palco para monitorar alterações em uma coleção.

Exemplo 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()));
```

```
}
```

Saída

```
{
  _id: { _data: '...' },
  operationType: 'insert',
  clusterTime: Timestamp(1, 1234567890),
  fullDocument: { _id: 1, item: 'Widget', qty: 10 },
  ns: { db: 'test', coll: 'inventory' },
  documentKey: { _id: 1 }
}
```

Exemplos de código

Para ver um exemplo de código para usar o estágio de `$changeStream` agregação, escolha a guia do idioma que você deseja 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()
```

$ alterar StreamSplitLargeEvent

Esse estágio de agregação não é suportado pelos clusters elásticos.

Esse estágio está disponível a partir da versão 8.0.2 do mecanismo Amazon DocumentDB.

O estágio `$changeStreamSplitLargeEvent` de agregação é usado em um pipeline $changeStream para dividir eventos que excedam 16 MB em fragmentos menores. Ele retorna cada fragmento sequencialmente por meio do cursor do fluxo de mudança.

Parâmetros

Nenhum. A entrada para o `$changeStreamSplitLargeEvent` palco deve ser um documento vazio.

Exemplo — MongoDB Shell

O exemplo a seguir demonstra o uso do `$changeStreamSplitLargeEvent` palco para dividir um evento de alteração de grandes dimensões em vários fragmentos.

Exemplo 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()));
}
```

Saída

```
{
  _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: [] }
}
```

Exemplos de código

Para ver um exemplo de código para usar o estágio de `$changeStreamSplitLargeEvent` agregação, escolha a guia do idioma que você deseja 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

O `$cmp` operador no Amazon DocumentDB é usado para comparar dois valores e retornar um valor inteiro que indica sua ordem relativa. É um operador de comparação que compara duas expressões e retorna um valor inteiro de -1, 0 ou 1, dependendo se o primeiro valor é menor, igual ou maior que o segundo valor, respectivamente.

Parâmetros

- `expression1`: A primeira expressão a ser comparada.
- `expression2`: A segunda expressão a ser comparada.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$cmp` operador para comparar dois valores numéricos.

Crie documentos de amostra

```
db.collection.insertMany([
  { _id: 1, value1: 10, value2: 20 },
  { _id: 2, value1: 15, value2: 15 },
  { _id: 3, value1: 20, value2: 10 }
]);
```

Exemplo de consulta

```
db.collection.find({
  $expr: {
    $cmp: ["$value1", "$value2"]
  }
})
```

Saída

```
[
  { "_id" : 1, "value1" : 10, "value2" : 20 },
  { "_id" : 3, "value1" : 20, "value2" : 10 }
]
```

Neste exemplo, o $cmp operador compara os value2 campos value1 e de cada documento. O resultado é:

```
- `$cmp: ["$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).
```

Exemplos de código

Para ver um exemplo de código para usar o $cmp comando, escolha a guia do idioma que você deseja usar:

Node.js

Aqui está um exemplo de uso do $cmp operador em um Node.js aplicativo com o mongodb driver:

```
const { MongoClient } = require('mongodb');

const client = new MongoClient('mongodb://<username>:<password>@<cluster-
endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');

async function main() {
  await client.connect();
  const db = client.db('test');
  const collection = db.collection('mycollection');

  // Insert sample documents
  await collection.insertMany([
    { _id: 1, value1: 10, value2: 20 },
    { _id: 2, value1: 15, value2: 15 },
    { _id: 3, value1: 20, value2: 10 }
  ]);

  // Query using $cmp operator
  const result = await collection.find({
    $expr: {
      $cmp: ['$value1', '$value2']
```

```
    }
  }).toArray();

  console.log(result);

  await client.close();
}

main();
```

Python

Aqui está um exemplo de uso do $cmp operador em um aplicativo Python com o pymongo driver:

```
from pymongo import MongoClient

client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
db = client['test']
collection = db['mycollection']

# Insert sample documents
collection.insert_many([
    {'_id': 1, 'value1': 10, 'value2': 20},
    {'_id': 2, 'value1': 15, 'value2': 15},
    {'_id': 3, 'value1': 20, 'value2': 10}
])

# Query using $cmp operator
result = list(collection.find({
    '$expr': {
        '$cmp': ['$value1', '$value2']
    }
}))

print(result)

client.close()
```

A saída dos exemplos Node.js e do Python será a mesma do exemplo do MongoDB Shell:

```
[
```

```
  { "_id" : 1, "value1" : 10, "value2" : 20 },
  { "_id" : 2, "value1" : 15, "value2" : 15 },
  { "_id" : 3, "value1" : 20, "value2" : 10 }
]
```

$collStats

Novo da versão 4.0

O estágio `$collStats` de agregação no Amazon DocumentDB fornece estatísticas sobre a coleção especificada, semelhante ao comando `db.collection.stats()` no shell do MongoDB. Esse estágio pode ser usado para recuperar informações sobre a coleção, como o número de documentos, o tamanho total da coleção e várias métricas de desempenho.

Parâmetros

- `latencyStats`: (opcional) Um documento que especifica as opções para coletar estatísticas de latência. Esse parâmetro não é suportado no Amazon DocumentDB.
- `recordStats`: (opcional) Um documento que especifica as opções para coletar estatísticas de registros. Esse parâmetro não é suportado no Amazon DocumentDB.
- `queryExecStats`: (opcional) Um documento que especifica as opções para coletar estatísticas de execução da consulta. Esse parâmetro não é suportado no Amazon DocumentDB.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o estágio de `$collStats` agregação para recuperar estatísticas sobre uma coleção nomeada `test` no banco de dados. db

Crie documentos de amostra

```
db.test.insertMany([
  { "name": "John", "age": 30 },
  { "name": "Jane", "age": 25 },
  { "name": "Bob", "age": 35 }
]);
```

Exemplo de consulta

```
db.test.aggregate([
```

```
  { $collStats: {} }
]);
```

Saída

```
{
  "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)
}
```

Exemplos de código

Para ver um exemplo de código para usar o `$collStats` comando, escolha a guia do idioma que você deseja usar:

Node.js

Aqui está um exemplo de como usar o estágio de `$collStats` agregação em um Node.js aplicativo usando o driver oficial do MongoDB Node.js :

```
const { MongoClient } = require('mongodb');

async function runCollStatsExample() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('db');
  const collection = db.collection('test');

  const result = await collection.aggregate([
    { $collStats: {} }
  ]).toArray();

  console.log(result);

  await client.close();
}

runCollStatsExample();
```

Python

Aqui está um exemplo de como usar o estágio de `$collStats` agregação em um aplicativo Python usando o PyMongo driver:

```
from pymongo import MongoClient

def run_coll_stats_example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['db']
    collection = db['test']

    result = list(collection.aggregate([
        { '$collStats': {} }
    ]))
```

```
    print(result)

    client.close()

run_coll_stats_example()
```

$concat

O operador de `$concat` agregação no Amazon DocumentDB concatena (ou combina) várias sequências de caracteres em um documento para produzir uma única sequência de caracteres que pode ser retornada ao aplicativo. Isso reduz o trabalho realizado no aplicativo, pois a manipulação de strings é realizada no nível do banco de dados.

Parâmetros

- `expression1`: a primeira string a ser concatenada.
- `expression2`: a segunda string a ser concatenada.
- `...`: sequências adicionais para concatenar (opcional).

Exemplo (MongoDB Shell)

Neste exemplo, concatenamos o nome e o sobrenome dos usuários para produzir o nome completo de cada pessoa.

Crie documentos de amostra

```
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"}
])
```

Exemplo de consulta

```
db.people.aggregate([
  { $project: { full_name: { $concat: [ "$first_name", " ", "$last_name"] } } }
```

```
])
```

Saída

```
{ "_id" : 1, "full_name" : "Jane Doe" }
{ "_id" : 2, "full_name" : "John Doe" }
{ "_id" : 3, "full_name" : "Steve Smith" }
```

Exemplos de código

Para ver um exemplo de código para usar o `$concat` comando, escolha a guia do idioma que você deseja 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

O operador `$concatArrays` de agregação no Amazon DocumentDB é usado para concatenar duas ou mais matrizes em uma única matriz. Isso pode ser útil quando você precisa combinar várias matrizes de dados em uma única matriz para processamento ou análise adicionais.

Parâmetros

- `array1`: a primeira matriz a ser concatenada.
- `array2`: a segunda matriz a ser concatenada.
- `[array3, ...]`: (opcional) Matrizes adicionais a serem concatenadas.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$concatArrays` operador para combinar duas matrizes em uma única matriz.

Crie documentos de amostra

```
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"]
  }
]);
```

Exemplo de consulta

```
db.collection.aggregate([
  {
    $project: {
      _id: 0,
      name: 1,
      all_activities: { $concatArrays: ["$hobbies", "$skills"] }
    }
  }
]);
```

Saída

```
[
  {
    "name": "John Doe",
    "all_activities": [
      "reading",
      "swimming",
      "programming",
      "design"
    ]
  },
  {
    "name": "Jane Smith",
    "all_activities": [
      "hiking",
      "cooking",
      "marketing",
      "analysis"
    ]
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$concatArrays` comando, escolha a guia do idioma que você deseja 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

O `$cond` operador no Amazon DocumentDB é usado para avaliar uma expressão condicional e retornar uma das duas expressões de resultado possíveis.

Parâmetros

- `if`: A expressão booleana a ser avaliada.
- `then`: A expressão a ser retornada se a `if` expressão for verdadeira.
- `else`: A expressão a ser retornada se a `if` expressão for falsa.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$cond` operador para retornar um valor com base na idade de uma pessoa.

Crie documentos de amostra

```
db.people.insertMany([
  { _id: 1, name: "John Doe", age: 35 },
  { _id: 2, name: "Jane Doe", age: 25 },
  { _id: 3, name: "Bob Smith", age: 65 }
]);
```

Exemplo 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"
            }
          }
        }
      }
    }
  }
])
```

Saída

```
[
  { "_id" : 1, "name" : "John Doe", "ageGroup" : "middle-aged" },
  { "_id" : 2, "name" : "Jane Doe", "ageGroup" : "young" },
  { "_id" : 3, "name" : "Bob Smith", "ageGroup" : "elderly" }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$cond` comando, escolha a guia do idioma que você deseja 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

Novo a partir da versão 4.0

O `$convert` operador no Amazon DocumentDB é usado para converter um valor de um tipo de dados para outro. Esse operador é útil quando você precisa realizar operações em dados de diferentes tipos, como converter uma string em um número ou uma data em um timestamp.

Parâmetros

- `to`: o tipo de dados de destino para o qual converter o valor. Os valores compatíveis são `"string"`, `"double"`, `"long"`, `"int"`, `"date"` e `"boolean"`.
- `from`: o tipo de dados atual do valor. Se não for especificado, o Amazon DocumentDB tentará detectar automaticamente o tipo de dados.
- `onError`: (opcional) O valor a ser retornado se a conversão falhar. Pode ser um valor específico ou um dos seguintes valores especiais:`"null"`,`"zerofill"`, ou`"error"`.

- onNull: (opcional) O valor a ser retornado se o valor de entrada fornull. Pode ser um valor específico ou um dos seguintes valores especiais:"null","zerofill", ou"error".

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra a conversão de um valor de string em uma data usando o $convert operador.

Crie documentos de amostra

```
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" }
]);
```

Exemplo de consulta

```
db.users.aggregate([
  {
    $project: {
      _id: 1,
      name: 1,
      joinedOn: {
        $convert: {
          input: "$joinedOn",
          to: "date",
          onError: "null",
          onNull: "null"
        }
      }
    }
  }
])
```

Saída

```
[
  { "_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 }
```

```
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$convert` comando, escolha a guia do idioma que você deseja 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

Novo na versão 8.0.1.

O `$cos` operador no Amazon DocumentDB retorna o cosseno de um valor que é medido em radianos. Use-o no pipeline de agregação para realizar cálculos trigonométricos em campos numéricos.

Parâmetros

- `expression`: uma expressão que se resolve em um número em radianos.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$cos` operador para calcular o cosseno dos valores angulares em radianos.

Crie documentos de amostra

```
db.angles.insertMany([
  { "_id": 1, "angle": 0 },
  { "_id": 2, "angle": 1.0471975511965976 },
  { "_id": 3, "angle": 3.141592653589793 }
]);
```

Exemplo de consulta

```
db.angles.aggregate([
  { $project: {
    "cosine": { $cos: "$angle" }
  }}
]);
```

Saída

```
[
  { "_id": 1, "cosine": 1 },
  { "_id": 2, "cosine": 0.5000000000000001 },
  { "_id": 3, "cosine": -1 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$cos` operador, escolha a guia do idioma que você deseja 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

Novo na versão 8.0.1.

O `$cosh` operador no Amazon DocumentDB retorna o cosseno hiperbólico de um valor. Use-o no pipeline de agregação para realizar cálculos trigonométricos hiperbólicos em campos numéricos.

Parâmetros

- `expression`: uma expressão que se resolve em um número.

O tipo de retorno é `double` por padrão. Se a entrada for decimal de 128 bits, a saída também será decimal de 128 bits.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$cosh` operador para calcular o cosseno hiperbólico de valores numéricos.

Crie documentos de amostra

```
db.values.insertMany([
  { "_id": 1, "value": 0 },
  { "_id": 2, "value": 1 },
  { "_id": 3, "value": -1 }
]);
```

Exemplo de consulta

```
db.values.aggregate([
  { $project: {
    "result": { $cosh: "$value" }
  }}
]);
```

Saída

```
[
  { "_id": 1, "result": 1 },
  { "_id": 2, "result": 1.5430806348152437 },
```

```
  { "_id": 3, "result": 1.5430806348152437 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$cosh` operador, escolha a guia do idioma que você deseja 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 (estágio)

O estágio `$count` de agregação no Amazon DocumentDB é usado para contar o número de documentos que passam para o estágio. Geralmente, é usado como o estágio final em um pipeline de agregação para retornar a contagem total de documentos correspondentes aos estágios anteriores.

Parâmetros

- `field`: O campo a ser contado. Esse parâmetro é opcional e, se não for fornecido, o estágio contará o número total de documentos de entrada.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$count` estágio para obter o número total de documentos em uma coleção.

Crie documentos de amostra

```
db.users.insertMany([
  { name: "John", age: 30 },
  { name: "Jane", age: 25 },
  { name: "Bob", age: 35 },
  { name: "Alice", age: 28 }
]);
```

Exemplo de consulta

```
db.users.aggregate([
```

```
  { $count: "total" }
]);
```

Saída

```
{ "total" : 4 }
```

O exemplo agrega a `users` coleção e usa o `$count` estágio para contar o número total de documentos.

Exemplos de código

Para ver um exemplo de código para usar o `$count` comando, escolha a guia do idioma que você deseja usar:

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)

Novo na versão 8.0.1.

Use o `$count` acumulador dentro de um `$group` estágio para retornar o número de documentos em cada grupo. Ele aceita um objeto vazio `{}` como argumento.

Isso é diferente do estágio do `$count` pipeline, que é um estágio independente que conta todos os documentos que passam pelo pipeline. Em vez disso, o `$count` acumulador conta os documentos dentro de cada grupo produzido por`$group`.

Sintaxe

```
{ $count: {} }
```

Parâmetros

- O `$count` acumulador não aceita argumentos. Ele aceita um objeto vazio `{}` e retorna a contagem de documentos em cada grupo.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$count` acumulador para contar o número de produtos em cada categoria.

Crie documentos de amostra

```
db.products.insertMany([
  { name: "Widget", category: "A" },
  { name: "Gadget", category: "A" },
  { name: "Doohickey", category: "B" },
  { name: "Thingamajig", category: "B" },
  { name: "Whatsit", category: "B" }
])
```

Exemplo de consulta

```
db.products.aggregate([
  { $group: { _id: "$category", count: { $count: {} } } }
])
```

Saída

```
[
  { "_id": "A", "count": 2 },
  { "_id": "B", "count": 3 }
]
```

Exemplo de uso do operador de janela (MongoDB Shell)

Novo na versão 8.0.2.

O `$count` acumulador também pode ser usado como operador de janela no `$setWindowFields` palco. Nesse contexto, ele retorna o número de documentos em cada janela. Você especifica o operador no `output` campo e, opcionalmente, define os limites da janela com um `window` documento.

Crie documentos de amostra

```
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 }
])
```

Exemplo de consulta

O exemplo a seguir particiona os documentos por`category`, classifica cada partição por `day` e retorna uma contagem contínua de documentos desde o início da partição até o documento atual.

```
db.events.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$category",
      sortBy: { day: 1 },
      output: {
        runningCount: {
          $count: {},
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
])
```

Saída

```
[
  { "_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 é aumentado com `runningCount` o número de documentos em sua partição até o documento atual, inclusive.

Exemplos de código

Para ver um exemplo de código para usar o `$count` acumulador, escolha a guia do idioma que você deseja usar. Os exemplos a seguir mostram o uso do acumulador (em`$group`) e o uso do operador de janela (em`$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

Novo na versão 8.0.2.

O `$covariancePop` operador no Amazon DocumentDB retorna a covariância populacional de duas expressões numéricas. É um operador de janela usado somente no `$setWindowFields` estágio;

não é válido nos $bucket estágios $group ou. Você especifica o operador no output campo do estágio e, opcionalmente, define os limites da janela com um window documento.

Parâmetros

- $covariancePopusa uma matriz de dois elementos[<expression1>, <expression2>], em que cada elemento é uma expressão numérica (um caminho de campo"$x", como uma expressão ou uma constante). Ele retorna a covariância populacional das duas expressões nos documentos na janela.

Exemplo (MongoDB Shell)

O exemplo a seguir particiona os documentos porseries, classifica cada partição por t e calcula a covariância da população de x e y sobre toda a partição.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.measurements.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$series",
      sortBy: { t: 1 },
      output: {
        covariance: {
          $covariancePop: [ "$x", "$y" ],
          window: { documents: ["unbounded", "unbounded"] }
        }
      }
    }
  }
```

```
]);
```

Saída

```
[
  { "_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 }
]
```

Cada documento em uma partição recebe a mesma `covariance` covariância populacional de `x` e `y` entre todos os documentos nessa partição: 2 para séries `A` (x e y aumentam juntos) e `-2` para séries `B` (y diminui à medida que x aumenta).

Exemplos de código

Para ver um exemplo de código para usar o `$covariancePop` operador no `$setWindowFields` estágio, escolha a guia do idioma que você deseja 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 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

Novo na versão 8.0.2.

O `$covarianceSamp` operador no Amazon DocumentDB retorna a covariância de amostra de duas expressões numéricas. É um operador de janela usado somente no `$setWindowFields` estágio; não é válido nos `$bucket` estágios `$group` ou. Você especifica o operador no `output` campo do estágio e, opcionalmente, define os limites da janela com um `window` documento.

Parâmetros

- `$covarianceSamp`usa uma matriz de dois elementos`[ <expression1>, <expression2> ]`, em que cada elemento é uma expressão numérica (um caminho de campo`"$x"`, como uma expressão ou uma constante). Ele retorna a covariância da amostra das duas expressões nos documentos na janela.

Exemplo (MongoDB Shell)

O exemplo a seguir particiona os documentos por`series`, classifica cada partição por `t` e calcula a covariância da amostra de `x` e `y` sobre toda a partição.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.measurements.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$series",
      sortBy: { t: 1 },
      output: {
        covariance: {
          $covarianceSamp: [ "$x", "$y" ],
          window: { documents: ["unbounded", "unbounded"] }
        }
      }
    }
  }
]);
```

Saída

```
[
  { "_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 }
```

```
]
```

Cada documento em uma partição recebe a mesma `covariance` covariância de amostra de `x` e `y` entre todos os documentos nessa partição: `3` para séries `A` (x e y aumentam juntos) e `-3` para séries `B` (y diminui à medida que x aumenta). A covariância da amostra se divide por `n - 1` em vez den, portanto, é maior em magnitude do que a covariância da população para os mesmos dados.

Exemplos de código

Para ver um exemplo de código para usar o `$covarianceSamp` operador no `$setWindowFields` estágio, escolha a guia do idioma que você deseja 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 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

O estágio `$currentOp` de agregação retorna informações sobre as operações atualmente em execução no banco de dados. Esse estágio é útil para monitorar consultas e operações ativas em um pipeline de agregação.

Parâmetros

- `allUsers`(opcional): quando definido como`true`, retorna as operações para todos os usuários. O padrão é `false`.
- `idleConnections`(opcional): quando definido como`true`, inclui conexões inativas. O padrão é `false`.
- `idleCursors`(opcional): quando definido como`true`, inclui informações sobre cursores inativos. O padrão é `false`.
- `idleSessions`(opcional): quando definido como`true`, inclui informações sobre sessões ociosas. O padrão é `true`.
- `localOps`(opcional): quando definido como`true`, inclui operações locais. O padrão é `false`.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do estágio de `$currentOp` agregação para recuperar informações sobre operações de leitura ativas.

Exemplo de consulta

```
db.aggregate([
  { $currentOp: { allUsers: true, idleConnections: false } },
  { $match: { op: "query" } }
])
```

Saída

```
[
  {
    "opid": "12345",
    "active": true,
    "op": "query",
    "ns": "test.users",
    "secs_running": 2
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$currentOp` comando, escolha a guia do idioma que você deseja 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

Novo a partir da versão 5.0

O operador `$dateAdd` de agregação no Amazon DocumentDB permite que você adicione uma duração a um valor de data e hora.

Parâmetros

- `date`: um valor de data e hora ao qual adicionar uma duração.
- `duration`: a duração a ser adicionada ao `date` valor. Isso pode ser especificado como um objeto com chaves para `years`, `months`, `weeks`, `days`, `hours`, `minutes`, e `seconds`.
- `timezone`: (opcional) O fuso horário a ser usado ao realizar a adição da data. Se não for especificado, o fuso horário padrão do cluster Amazon DocumentDB será usado.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$dateAdd` operador para adicionar 2 dias e 12 horas a uma data.

Crie documentos de amostra

```
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") }
]);
```

Exemplo de consulta

```
db.events.aggregate([
  {
    $project: {
      _id: 1,
      eventDate: 1,
      eventDatePlustwodaysandtwelvehours: {
        $dateAdd: {
          startDate: {
            $dateAdd: {
              startDate: "$eventDate",
              unit: "day",
              amount: 2
            }
          },
          unit: "hour",
          amount: 12
        }
      }
    }
  }
])
```

Saída

```
[
  {
    "_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)"
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$dateAdd` comando, escolha a guia do idioma que você deseja 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

Novo da versão 5.0

Não é compatível com o cluster elástico.

O operador `$dateDiff` de agregação calcula a diferença entre duas datas em unidades especificadas. Ele retorna o número de limites da unidade cruzados entre as datas de início e término.

Parâmetros

- `startDate`: A expressão da data inicial.
- `endDate`: A expressão da data final.
- `unit`: A unidade de tempo para a diferença. As unidades suportadas são `year` `quarter`,`month`,`week`,`day`, `hour`,`minute`,`second`,, `millisecond` e.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$dateDiff` operador para calcular o número de dias entre a realização do pedido e a entrega.

Crie documentos de amostra

```
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")
  }
]);
```

Exemplo de consulta

```
db.shipments.aggregate([
  {
    $project: {
      orderId: 1,
      orderDate: 1,
      deliveryDate: 1,
      daysToDeliver: {
        $dateDiff: {
          startDate: "$orderDate",
          endDate: "$deliveryDate",
          unit: "day"
        }
      }
    }
  }
]);
```

Saída

```
[
  {
    _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
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$dateDiff` comando, escolha a guia do idioma que você deseja 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()
```

$data FromParts

Novo a partir das versões 5.0.1 e 8.0

O operador de `$dateFromParts` agregação no Amazon DocumentDB constrói e retorna um valor de data e hora de suas partes constituintes, como ano, mês, dia, hora, minuto, segundo e milissegundo.

Parâmetros

- `year`: O ano civil. Obrigatório quando não está usando peças ISO 8601.
- `month`,`day`,`hour`,`minute`,`second`,`millisecond`: (opcional) Os componentes restantes de data e hora. Os componentes omitidos assumem como padrão o menor valor válido.

- `isoWeekYear`,`isoWeek`,`isoDayOfWeek`: (opcional) Componentes de data ISO 8601. Use-os em vez de `yearmonth`, e `day` para criar uma data a partir de uma data da semana ISO 8601.
- `timezone`: (opcional) O fuso horário a ser usado ao criar a data. Se não for especificado, o UTC será usado.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$dateFromParts` operador para criar uma data a partir de componentes individuais.

Crie documentos de amostra

```
db.parts.insertMany([
  { _id: 1, y: 2023, m: 4, d: 1 },
  { _id: 2, y: 2023, m: 12, d: 25 }
]);
```

Exemplo de consulta

```
db.parts.aggregate([
  {
    $project: {
      _id: 1,
      constructedDate: {
        $dateFromParts: { year: "$y", month: "$m", day: "$d" }
      }
    }
  }
])
```

Saída

```
[
  { "_id": 1, "constructedDate": ISODate("2023-04-01T00:00:00Z") },
  { "_id": 2, "constructedDate": ISODate("2023-12-25T00:00:00Z") }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$dateFromParts` comando, escolha a guia do idioma que você deseja 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()
```

$data FromString

O operador `$dateFromString` de agregação no Amazon DocumentDB permite que você converta uma string de data e hora em um objeto de data. Isso é útil quando seus dados são armazenados como cadeias de caracteres, mas você precisa realizar operações com base em datas nos dados.

Parâmetros

- `dateString`: uma string que representa uma data e hora.
- `format`: (opcional) Uma string que especifica o formato do`dateString`. Se não for fornecido, o Amazon DocumentDB tentará analisar a string no formato. ISO-8601
- `timezone`: (opcional) Uma string que especifica o fuso horário. Se não for fornecido, o Amazon DocumentDB usará o fuso horário do servidor.
- `onError`: (opcional) Especifica a ação a ser tomada se a conversão falhar. Os valores possíveis são `'error'` (o padrão, que gera um erro), `'null'` (retorna`null`) ou `'replace'` (substitui o valor pela string de substituição especificada na `onErrorMessage` opção).
- `onErrorMessage`: (opcional) Se `onError` estiver definida como`'replace'`, essa opção especifica a string de substituição.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar `$dateFromString` para converter uma string de data em um objeto de data no Amazon DocumentDB.

Crie documentos de amostra

```
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"}
]);
```

Exemplo de consulta

```
db.missionLog.aggregate([
  {
    $project: {
      event: '$event',
      logDate: {
        $dateFromString: {
          dateString: '$logDate'
        }
      }
    }
  }
]);
```

Saída

```
[
  {
    "_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")
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$dateFromString` comando, escolha a guia do idioma que você deseja 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

Novo a partir da versão 5.0

O operador `$dateSubtract` de agregação no Amazon DocumentDB permite que você subtraia uma duração especificada de um valor de data.

Parâmetros

- `date`: uma expressão de data que se transforma em uma data ou carimbo de data/hora.

- `subtrahend`: uma expressão de duração que especifica a quantidade de tempo a ser subtraída da `date` expressão.
- `unit`: uma string que especifica a unidade de tempo para a `subtrahend` expressão. As unidades suportadas são “ano”, “trimestre”, “mês”, “semana”, “dia”, “hora”, “minuto”, “segundo” e “milissegundo”.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$dateSubtract` operador para calcular a data de um ano atrás a partir da data atual.

Crie documentos de amostra

```
db.events.insertOne({
  eventName: "Player joined",
  eventTime: ISODate("2023-04-01T12:00:00Z")
});
```

Exemplo de consulta

```
db.events.aggregate([
  {
    $project: {
      eventName: 1,
      oneYearAgo: {
        $dateSubtract: {
          startDate: "$eventTime",
          amount: 1,
          unit: "year"
        }
      }
    }
  }
])
```

Saída

```
{
  "_id" : ObjectId("64567890abcdef012345678"),
  "eventName" : "Player joined",
```

```
  "oneYearAgo" : ISODate("2022-04-01T12:00:00Z")
}
```

Exemplos de código

Para ver um exemplo de código para usar o `$dateSubtract` comando, escolha a guia do idioma que você deseja 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()
```

$data ToParts

Novo a partir das versões 5.0.1 e 8.0

O operador de `$dateToParts` agregação no Amazon DocumentDB retorna um documento que contém as partes constituintes de um valor de data e hora, como ano, mês, dia, hora, minuto, segundo e milissegundo.

Parâmetros

- `date`: o valor de data e hora para dividir em partes.
- `timezone`: (opcional) O fuso horário a ser usado ao extrair as partes da data. Se não for especificado, as peças serão devolvidas em UTC.

- `iso8601`: (opcional) Se definido como`true`, o documento retornado usa partes de data ISO 8601 (`isoWeekYear`,`isoWeek`,`isoDayOfWeek`). O padrão é `false`.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$dateToParts` operador para retornar as partes individuais de uma data.

Crie documentos de amostra

```
db.events.insertMany([
  { _id: 1, eventDate: ISODate("2023-04-01T10:30:15Z") },
  { _id: 2, eventDate: ISODate("2023-12-25T18:45:00Z") }
]);
```

Exemplo de consulta

```
db.events.aggregate([
  {
    $project: {
      _id: 1,
      parts: { $dateToParts: { date: "$eventDate" } }
    }
  }
])
```

Saída

```
[
  {
    "_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 }
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$dateToParts` comando, escolha a guia do idioma que você deseja 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()
```

$data ToString

O operador `$dateToString` de agregação no Amazon DocumentDB é usado para converter um valor de data ou timestamp em uma representação de string. Isso é útil quando você precisa formatar a data e a hora de uma forma específica para exibição ou processamento posterior.

Parâmetros

- `date`: o valor da data ou do carimbo de data/hora a ser convertido em uma string.
- `format`: uma string que especifica o formato no qual a data deve ser representada. A string de formato pode incluir vários especificadores de formato, como `%Y` para o ano de quatro dígitos, para o mês de dois dígitos, `%m` para o dia do mês com dois dígitos, `%d` etc.
- `timezone`: (opcional) O fuso horário a ser usado para a conversão. Se não for especificado, o fuso horário do servidor que hospeda o cluster Amazon DocumentDB será usado.
- `onNull`: (opcional) O valor a ser retornado se o `date` parâmetro for`null`.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$dateToString` operador para formatar o `logDate` campo da `missionLog` coleção.

Crie documentos de amostra

```
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") }
]);
```

Exemplo de consulta

```
db.missionLog.aggregate([
  {
    $project: {
      event: "$event",
      logDateFormatted: {
        $dateToString: {
          format: "%Y-%m-%d %H:%M:%S",
          date: "$logDate"
        }
      }
    }
  }
])
```

Saída

```
[
  {
    "_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"
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$dateToString` comando, escolha a guia do idioma que você deseja usar:

Node.js

Aqui está um exemplo de uso do `$dateToString` operador em um Node.js aplicativo:

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Aqui está um exemplo de uso do `$dateToString` operador em um aplicativo Python:

```
from pymongo import MongoClient

def 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

Novo da versão 8.0

Não é compatível com o cluster elástico.

O operador `$dateTrunc` de agregação no Amazon DocumentDB trunca uma data para uma unidade especificada.

Parâmetros

- `date`: uma expressão de data que se transforma em uma data ou carimbo de data/hora.
- `unit`: uma string que especifica a unidade de tempo para a expressão subtrahend. As unidades suportadas são `year` `quarter`month,`week`,`day`, `hour`minute,`second`,, `millisecond` e.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$dateTrunc` operador para truncar uma data até a hora.

Crie documentos de amostra

```
db.events.insertMany([
  {
    eventName: "Event 1",
    eventTime: ISODate("2025-04-01T12:15:00Z")
  },
  {
    eventName: "Event 2",
    eventTime: ISODate("2025-08-15T14:33:22Z")
  },
]);
```

Exemplo de consulta

```
db.events.aggregate([
  {
    $project: {
      eventName: 1,
      eventTime: 1,
      truncatedToHour: {
        $dateTrunc: {
          date: "$eventTime",
          unit: "hour"
        }
```

```
      }
    }
  }
]);
```

Saída

```
[
  {
    _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')
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$dateTrunc` comando, escolha a guia do idioma que você deseja 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()
```

$dia OfMonth

O operador `$dayOfMonth` de agregação no Amazon DocumentDB recupera o dia do mês (de 1 a 31) de uma determinada data. Esse operador é útil para agrupar, filtrar ou extrair o dia do mês dos campos de data em seus documentos.

Parâmetros

- `date expression`: a expressão de data pode ser um campo de data do documento, um objeto de data ou uma string de data.

Exemplo (MongoDB Shell)

Este exemplo demonstra como usar o `$dayOfMonth` operador para extrair o dia do mês de um campo de data no documento.

Crie documentos de amostra

```
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") }
]);
```

Exemplo de consulta

```
db.events.aggregate([
  { $project: {
    eventDay: { $dayOfMonth: "$eventDate" }
  }}
])
```

Saída

```
[
```

```
  { "_id" : 1, "eventDay" : 15 },
  { "_id" : 2, "eventDay" : 28 },
  { "_id" : 3, "eventDay" : 1 },
  { "_id" : 4, "eventDay" : 30 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$dayOfMonth` comando, escolha a guia do idioma que você deseja 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()
```

$dia OfWeek

O `$dayOfWeek` operador no Amazon DocumentDB extrai o dia da semana de um determinado campo de data. Ele retorna o dia da semana como um número entre 1 (domingo) e 7 (sábado), que é o mesmo comportamento do MongoDB.

Parâmetros

- `date field`: O campo de data do qual extrair o dia da semana.

Exemplo (MongoDB Shell)

Este exemplo demonstra como usar o `$dayOfWeek` operador para extrair o dia da semana do `date` campo na `weather` coleção.

Crie documentos de amostra

```
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")
  }
]);
```

Exemplo de consulta

```
db.weather.aggregate([
  {
    $project: {
      dayOfWeek: { $dayOfWeek: "$date" }
    }
  }
]).pretty();
```

Saída

```
{ "_id" : ObjectId("64272c6663f4f8ce422c2d91"), "dayOfWeek" : 7 }
{ "_id" : ObjectId("64272c6663f4f8ce422c2d92"), "dayOfWeek" : 1 }
{ "_id" : ObjectId("64272c6663f4f8ce422c2d93"), "dayOfWeek" : 2 }
```

Exemplos de código

Para ver um exemplo de código para usar o `$dayOfWeek` comando, escolha a guia do idioma que você deseja 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()
```

$dia OfYear

O `$dayOfYear` operador no Amazon DocumentDB retorna o dia do ano de uma data como um número entre 1 e 366 (365 em anos não bissextos).

Parâmetros

- `expression`: o campo de data ou expressão do qual extrair o dia do ano.

Exemplo (MongoDB Shell)

Este exemplo demonstra como usar o `$dayOfYear` operador para extrair o dia do ano de um campo de data em uma coleção do Amazon DocumentDB.

Crie documentos de amostra

```
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")
  }
])
```

Exemplo de consulta

```
db.weather.aggregate([
  {
    $project: {
      dayOfYear: { $dayOfYear: "$date" }
    }
  }
]).pretty()
```

Saída

```
{ "_id" : ObjectId("642b86fc7d8e07af279bbe63"), "dayOfYear" : 74 }
{ "_id" : ObjectId("642b86fc7d8e07af279bbe64"), "dayOfYear" : 365 }
```

Exemplos de código

Para ver um exemplo de código para usar o `$dayOfYear` comando, escolha a guia do idioma que você deseja 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()
```

$ graus ToRadians

Novo na versão 8.0.1.

O `$degreesToRadians` operador no Amazon DocumentDB converte um valor de graus em radianos.

Parâmetros

- `expression`: uma expressão que se resolve em um valor numérico em graus.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$degreesToRadians` operador para converter valores de graus em radianos.

Crie documentos de amostra

```
db.angles.insertMany([
  { "_id": 1, "degrees": 0 },
  { "_id": 2, "degrees": 90 },
  { "_id": 3, "degrees": 180 },
  { "_id": 4, "degrees": 360 }
]);
```

Exemplo de consulta

```
db.angles.aggregate([
```

```
  { $project: {
    "radians": { $degreesToRadians: "$degrees" }
  }}
]);
```

Saída

```
[
  { "_id": 1, "radians": 0 },
  { "_id": 2, "radians": 1.5707963267948966 },
  { "_id": 3, "radians": 3.141592653589793 },
  { "_id": 4, "radians": 6.283185307179586 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$degreesToRadians` operador, escolha a guia do idioma que você deseja 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

Novo na versão 8.0.2.

O operador de `$denseRank` janela no Amazon DocumentDB retorna a posição de cada documento em sua partição, de acordo com a `sortBy` ordem das `$setWindowFields` etapas. Documentos com o mesmo `sortBy` valor recebem a mesma classificação, e o próximo valor distinto usa a posição imediatamente seguinte, sem deixar lacunas (por exemplo,`1, 2, 2, 3`). É especificado no `output` campo do `$setWindowFields` estágio.

É um operador de janela usado somente no `$setWindowFields` palco.

Parâmetros

- `$denseRank`usa um documento vazio `{}` como valor.
- Ele não aceita uma `window` moldura.
- Isso requer exatamente uma `sortBy` chave no `$setWindowFields` estágio.

Exemplo (MongoDB Shell)

O exemplo a seguir divide os documentos por`game`, classifica cada partição `score` em ordem decrescente e atribui a cada jogador uma classificação densa.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.players.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$game",
      sortBy: { score: -1 },
      output: {
        denseRank: { $denseRank: {} }
      }
    }
  }
]);
```

Saída

```
[
  { "_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` partição, os jogadores são classificados por ordem decrescente`score`. Bob e Carol empatam `85` e dividem a classificação`2`, e Dave segue na classificação `3` sem diferença.

Exemplos de código

Para ver um exemplo de código para usar o `$denseRank` operador no `$setWindowFields` estágio, escolha a guia do idioma que você deseja 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 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

O `$divide` operador no pipeline de agregação do Amazon DocumentDB é usado para dividir um número por outro. É um operador útil para realizar operações matemáticas em campos numéricos em seus documentos.

Parâmetros

- `numerator`: O dividendo ou o número a ser dividido.
- `denominator`: O divisor ou o número pelo qual dividir.

Exemplo (MongoDB Shell)

Este exemplo demonstra como usar o `$divide` operador para calcular a taxa horária dos funcionários com base no salário anual e no número de horas de trabalho por ano.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.employees.aggregate([
  {
    $project: {
      name: 1,
      hourlyRate: { $divide: ["$salary", "$hoursPerYear"] }
    }
  }
])
```

Saída

```
[
  { "_id" : 1, "name" : "John Doe", "hourlyRate" : 38.46153846153846 },
  { "_id" : 2, "name" : "Jane Smith", "hourlyRate" : 43.26923076923077 },
  { "_id" : 3, "name" : "Bob Johnson", "hourlyRate" : 36.05769230769231 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$divide` comando, escolha a guia do idioma que você deseja 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

Novo na versão 8.0.2.

O operador de `$documentNumber` janela no Amazon DocumentDB retorna um número sequencial distinto para cada documento em sua partição, de acordo com a ordem das `$setWindowFields` etapas`sortBy`. Ao contrário de `$rank` e`$denseRank`, ele não trata especialmente os vínculos: cada documento recebe um número exclusivo (por exemplo,`1, 2, 3, 4`) e documentos com `sortBy` valores iguais são numerados na ordem em que são encontrados. É especificado no `output` campo do `$setWindowFields` estágio.

É um operador de janela usado somente no `$setWindowFields` palco.

Parâmetros

- `$documentNumber`usa um documento vazio `{}` como valor.
- Ele não aceita uma `window` moldura.
- Isso requer exatamente uma `sortBy` chave no `$setWindowFields` estágio.

Exemplo (MongoDB Shell)

O exemplo a seguir particiona os documentos por`game`, classifica cada partição `score` em ordem decrescente e atribui a cada jogador um número de documento sequencial.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.players.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$game",
      sortBy: { score: -1 },
      output: {
        documentNumber: { $documentNumber: {} }
      }
    }
  }
]);
```

Saída

```
[
  { "_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` partição, cada jogador recebe um número sequencial distinto em ordem decrescente`score`. Bob e Carol empatam`85`, mas ainda estão numerados `2` e `3` na ordem em que foram encontrados.

Exemplos de código

Para ver um exemplo de código para usar o `$documentNumber` operador no `$setWindowFields` estágio, escolha a guia do idioma que você deseja 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 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

O operador `$eq` de agregação compara dois valores e retorna `true` se eles forem iguais, caso contrário, retorna. `false`

Parâmetros

- `expression1`: o primeiro valor a ser comparado.
- `expression2`: O segundo valor a ser comparado.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$eq` operador para verificar se as quantidades do produto correspondem aos valores-alvo.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.inventory.aggregate([
  {
    $project: {
      item: 1,
      qty: 1,
      target: 1,
      meetsTarget: { $eq: ["$qty", "$target"] }
    }
  }
```

```
]);
```

Saída

```
[
  { _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 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o operador de `$eq` agregação, escolha a guia do idioma que você deseja 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

Novo a partir da versão 4.0

O `$exp` operador no Amazon DocumentDB permite que você eleve a constante e para um determinado número.

Parâmetros

- `expression`: A expressão a ser avaliada. Isso pode ser qualquer expressão de agregação válida, incluindo referências de campo, operações aritméticas e outros estágios de agregação.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$exp` operador para encontrar todos os documentos em que o `quantity` campo é maior que o `price` campo.

Crie documentos de amostra

```
db.items.insertMany([
  { item: "canvas", quantity: 4 },
  { item: "journal", quantity: 2 }
]);
```

Exemplo de consulta

```
db.items.aggregate([
  { $project: {
    "quantityRaised": {$exp: "$quantity"}}
  }
]);
```

Saída

```
[
  {
    _id: ObjectId('6920b785311cf98b79d2950d'),
    quantityRaised: 54.598150033144236
  },
  {
    _id: ObjectId('6920b785311cf98b79d2950e'),
    quantityRaised: 7.38905609893065
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$exp` comando, escolha a guia do idioma que você deseja 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

Novo na versão 8.0.2.

O estágio `$facet` de agregação no Amazon DocumentDB processa vários pipelines de agregação em um único estágio no mesmo conjunto de documentos de entrada. Cada campo de saída define seu próprio pipeline e cada pipeline recebe os mesmos documentos de entrada. Isso permite calcular várias agregações independentes, como contagens, agrupamentos e estatísticas resumidas, em uma única consulta, em vez de executar a coleção por meio de várias consultas separadas.

Cada subpipeline é executado de forma independente e produz uma matriz de documentos que são armazenados em seu campo de saída correspondente. Como os subpipelines não compartilham o estado, a saída de um pipeline não afeta outro.

Sintaxe

```
{
  $facet: {
    <outputField1>: [ <stage1>, <stage2>, ... ],
    <outputField2>: [ <stage1>, <stage2>, ... ],
    ...
  }
}
```

Parâmetros

- `outputField`(obrigatório): o nome de um campo de saída que contém a matriz de resultados do pipeline de agregação associado. Você pode especificar um ou mais campos de saída. Cada campo de saída é mapeado para uma matriz de estágios do pipeline de agregação que é aplicada aos documentos de entrada.

Saída

O `$facet` estágio retorna um único documento. Cada campo de saída que você especificar contém uma matriz de documentos produzidos pelo subpipeline correspondente.

Limitações

- Um `$facet` subpipeline não pode incluir os seguintes estágios:`$changeStream`,`$collStats`, `$facet` (o aninhamento não é

permitido)`$geoNear`,`$indexStats`,`$listSearchIndexes`,`$merge`,`$out`,`$planCacheStats`, `$search``$searchMeta`, e. `$vectorSearch`

- O documento que o `$facet` estágio retorna deve permanecer dentro do limite de tamanho do documento BSON de 16 MB.
- Um `$sort` estágio anterior a um `$facet` estágio não pode determinar a ordem de um acumulador sensível à ordem (`$first`,,, `$last` `$push` `$firstN``$lastN`, ou`$mergeObjects`) em um `$bucketAuto` estágio`$group`,`$bucket`, ou quando essa ordem precisaria ser aplicada além do limite. `$facet` Isso se aplica se o estágio de agrupamento estiver dentro de um `$facet` subpipeline ou após o `$facet` estágio. Nesse caso, o Amazon DocumentDB retorna um erro em vez de executar o pipeline. Para solicitar esse acumulador, coloque o `$sort` estágio dentro da `$facet` subtubulação, imediatamente antes do estágio de agrupamento. Order-insensitive acumuladores, como`$sum`,`$min`, e `$max``$count`, não são afetados.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$facet` estágio para calcular várias agregações sobre os mesmos dados de vendas em uma única consulta.

Crie documentos de amostra

```
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") }
]);
```

Exemplo de consulta

O comando a seguir é usado `$facet` para calcular, em um único estágio, a quantidade total vendida por item, as estatísticas gerais de preços e a contagem 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" }
      ]
    }
  }
])
```

Saída

O comando retorna a seguinte saída:

```
[
  {
    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 }
    ]
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$facet` comando, escolha a guia do idioma que você deseja 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

O `$filter` operador no Amazon DocumentDB é usado para aplicar uma expressão de filtro a cada elemento de uma matriz e retornar uma matriz contendo somente os elementos que correspondem à condição especificada. Esse operador é útil quando você precisa realizar operações complexas de filtragem em campos de matriz em seus documentos.

Parâmetros

- `input`: O campo de matriz a ser filtrado.
- `as`: o nome da variável a ser usado para cada elemento da `input` matriz dentro da `cond` expressão.
- `cond`: a expressão booleana que determina se um determinado elemento deve ser incluído na matriz de saída.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$filter` operador para projetar o cliente de cada pedido e criar um novo campo de matriz paidItems contendo somente os itens da matriz de itens em

que o preço é maior que 15. Essencialmente, ele filtra os itens de cada pedido para incluir apenas produtos que custam mais de 15.

Crie documentos de amostra

```
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 }
  ]}
]);
```

Exemplo de consulta

```
db.orders.aggregate([
  {
    $project: {
      customer: 1,
      paidItems: {
        $filter: {
          input: "$items",
          as: "item",
          cond: { $gt: ["$$item.price", 15] }
        }
      }
    }
  }
]).pretty();
```

Saída

```
[
  {
    _id: 1,
    customer: 'abc123',
```

```
    paidItems: [ { name: 'Product B', price: 20, qty: 1 } ]
  },
  { _id: 2, customer: 'def456', paidItems: [] },
  { _id: 3, customer: 'ghi789', paidItems: [] }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$filter` comando, escolha a guia do idioma que você deseja 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

Novo na versão 5.0.

Não é compatível com o cluster elástico.

O `$first` operador no Amazon DocumentDB retorna o primeiro documento de um conjunto agrupado de documentos. É comumente usado em pipelines de agregação para recuperar o primeiro documento que corresponda a uma condição específica.

Parâmetros

- `expression`: a expressão a ser retornada como o primeiro valor em cada grupo.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$first` operador para recuperar o valor do primeiro item encontrado para cada categoria durante a agregação.

Nota: `$first` retorna o primeiro documento com base na ordem atual dos documentos no pipeline. Para garantir um pedido específico (por exemplo, por data, preço etc.), um `$sort` estágio deve ser usado antes do `$group` estágio.

Crie documentos de amostra

```
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" }
]);
```

Exemplo de consulta

```
db.products.aggregate([
  { $group: { _id: "$category", firstItem: { $first: "$item" } } }
]);
```

Saída

```
[
  { "_id" : "food", "firstItem" : "abc" },
```

```
  { "_id" : "toy", "firstItem" : "xyz" }
]
```

Exemplo de uso do operador de janela (MongoDB Shell)

Novo na versão 8.0.2.

O `$first` operador também pode ser usado como operador de janela no `$setWindowFields` palco. Nesse contexto, ele retorna o valor `expression` do primeiro documento em cada janela. Você especifica o operador no `output` campo e, opcionalmente, define os limites da janela com um `window` documento.

> **Note**
>
> Quando usado como operador de janela em`$setWindowFields`, `$first` é limitado a 100 MB de dados intermediários. Uma operação que excede esse limite retorna um erro.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

O exemplo a seguir particiona os documentos por`ticker`, classifica cada partição por `hour` e retorna o primeiro preço (mais antigo) registrado em cada partição.

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
      sortBy: { hour: 1 },
      output: {
        firstPrice: {
```

```
          $first: "$price",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Saída

```
[
  { "_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 é aumentado com `firstPrice` o preço do primeiro documento em sua janela (o mais antigo `hour` na partição).

Exemplos de código

Para ver um exemplo de código para usar o `$first` operador, escolha a guia do idioma que você deseja usar. Os exemplos a seguir mostram o uso do acumulador (em`$group`) e o uso do operador de janela (em`$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()
```

$primeiroN

Novo na versão 8.0.1.

O `$firstN` operador no Amazon DocumentDB retorna os primeiros N elementos. Quando usado como acumulador em um `$group` estágio, ele retorna uma matriz dos primeiros N valores em cada grupo. Quando usado como um operador de expressão de matriz, ele retorna os primeiros N elementos de uma matriz.

Parâmetros

- `input`: a expressão que resolve para o campo ou matriz do qual retornar valores.
- n: um número inteiro positivo que especifica quantos valores devem ser retornados. Quando usado como acumulador em um `$group` estágio, também n pode ser uma expressão, desde que seja resolvido para um número inteiro positivo com base no campo do grupo. `_id`

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$firstN` acumulador para recuperar as duas primeiras quantidades de cada item durante a agregação.

> **Note**
>
> `$firstN`seleciona valores na ordem em que os documentos chegam ao `$group` estágio. Para retornar os primeiros N valores de uma ordenação específica (por exemplo, por data ou pontuação), adicione um `$sort` estágio anterior`$group`.

Crie documentos de amostra

```
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") }
]);
```

Exemplo de consulta

```
db.sales.aggregate([
  { $group: { _id: "$item", firstTwoQuantities: { $firstN: { input: "$quantity", n:
 2 } } } }
]);
```

Saída

```
[
  { "_id": "abc", "firstTwoQuantities": [10, 5] },
  { "_id": "xyz", "firstTwoQuantities": [15, 7] }
]
```

Exemplo de uso de expressão (MongoDB Shell)

O `$firstN` operador também pode ser usado como uma expressão em um `$project` estágio para retornar os primeiros N elementos de um campo de matriz.

Crie documentos de amostra

```
db.inventory.insertMany([
  { _id: 1, item: "abc", tags: ["red", "green", "blue", "yellow", "purple"] },
  { _id: 2, item: "xyz", tags: ["alpha", "beta", "gamma"] }
]);
```

Exemplo de consulta

```
db.inventory.aggregate([
  { $project: {
      firstThreeTags: { $firstN: { input: "$tags", n: 3 } }
    }}
]);
```

Saída

```
[
  { "_id": 1, "firstThreeTags": ["red", "green", "blue"] },
  { "_id": 2, "firstThreeTags": ["alpha", "beta", "gamma"] }
]
```

Exemplo de uso do operador de janela (MongoDB Shell)

Novo na versão 8.0.2.

O `$firstN` operador também pode ser usado como operador de janela no `$setWindowFields` palco. Nesse contexto, ele retorna uma matriz de até os primeiros n valores dos documentos em cada janela. Você especifica o operador no `output` campo e, opcionalmente, define os limites da janela com um `window` documento.

> **Note**
>
> Quando usado como operador de janela em`$setWindowFields`, `$firstN` é limitado a 100 MB de dados intermediários. Uma operação que excede esse limite retorna um erro.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

O exemplo a seguir particiona os documentos por`ticker`, classifica cada partição por `hour` e retorna os dois primeiros preços vistos desde o início da partição até o documento atual.

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
      sortBy: { hour: 1 },
      output: {
        firstTwoSoFar: {
          $firstN: { input: "$price", n: 2 },
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Saída

```
[
  { "_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 é aumentado com `firstTwoSoFar` os dois primeiros `price` valores em sua janela (em `hour` ordem).

Exemplos de código

Para ver um exemplo de código para usar o `$firstN` operador, escolha a guia do idioma que você deseja usar. Os exemplos a seguir mostram o uso do acumulador (in`$group`), o uso da expressão (in`$project`) e o uso do operador de janela (in`$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

Novo a partir da versão 4.0.

O `$floor` operador no Amazon DocumentDB retorna o maior número inteiro menor ou igual ao número especificado. Esse operador é útil para arredondar valores numéricos.

Parâmetros

- `expression`: A expressão numérica a ser arredondada para baixo.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$floor` operador para arredondar um valor decimal até o número inteiro mais próximo.

Crie documentos de amostra

```
db.numbers.insertOne({ value: 3.14 });
```

Exemplo de consulta

```
db.numbers.aggregate([
  { $project: { _id: 0, floored: { $floor: "$value" } } }
]);
```

Saída

```
{ "floored" : 3 }
```

Exemplos de código

Para ver um exemplo de código para usar o `$floor` comando, escolha a guia do idioma que você deseja 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

O estágio `$geoNear` de agregação retorna os documentos em ordem de proximidade a um ponto especificado. Ele calcula a distância do ponto e inclui a distância nos documentos de saída.

Parâmetros

- `near`: o ponto a partir do qual calcular distâncias, especificado como GeoJSON ou coordenadas legadas.
- `distanceField`: o nome do campo para armazenar a distância calculada.
- `spherical`: Booleano indicando se a geometria esférica deve ser usada (necessária para pontos GeoJSON).
- `maxDistance`: opcional. Distância máxima do ponto central.
- `minDistance`: opcional. Distância mínima do ponto central.
- `query`: opcional. Critérios de filtro adicionais a serem aplicados.
- `limit`: opcional. Número máximo de documentos a serem devolvidos.
- `key`: opcional. Campo a ser usado para consulta geoespacial quando existirem vários índices geoespaciais.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$geoNear` palco para encontrar as lojas mais próximas de um determinado local.

Crie documentos de amostra

```
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] } }
]);
```

Exemplo de consulta

```
db.stores.aggregate([
  {
    $geoNear: {
      near: { type: "Point", coordinates: [-122.4, 37.8] },
      distanceField: "distance",
      spherical: true
    }
  }
```

```
]);
```

Saída

```
[
  { _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 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o estágio de `$geoNear` agregação, escolha a guia do idioma que você deseja 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

Novo na versão 8.0.2.

O estágio `$graphLookup` de agregação no Amazon DocumentDB realiza uma pesquisa recursiva em uma coleção. Para cada documento de entrada, `$graphLookup` começa com um ou mais valores e combina repetidamente um campo de conexão com um campo de destino, seguindo a cadeia de documentos correspondentes até que não haja mais correspondências ou uma profundidade de recursão opcional seja alcançada. Isso é útil para percorrer dados hierárquicos ou semelhantes a gráficos, como estruturas de relatórios organizacionais, árvores de categorias ou relacionamentos de rede.

Parâmetros

- `from`: A coleção de destino a ser pesquisada recursivamente. A `from` coleção deve estar no mesmo banco de dados.
- `startWith`: a expressão que especifica o valor ou valores com os quais iniciar a pesquisa recursiva. Esse valor é comparado com o `connectToField` dos documentos na `from` coleção. Se for `startWith` avaliada como uma matriz, a pesquisa será executada em todos os elementos da matriz ao mesmo tempo.
- `connectFromField`: o nome do campo cujo valor é usado para se conectar ao próximo documento na pesquisa recursiva. Se o valor for uma matriz, cada elemento será seguido individualmente pela travessia.
- `connectToField`: o nome do campo na `from` coleção que é comparado com o valor do`connectFromField`.
- `as`: o nome do campo de matriz adicionado a cada documento de saída. Esse campo contém os documentos encontrados durante a pesquisa recursiva. Para obter informações sobre a ordem desses documentos, consulte o item Ordenação de resultados na seção Comportamento a seguir.
- `maxDepth`: um número inteiro não negativo que especifica o número máximo de iterações recursivas a serem executadas após a pesquisa inicial. Esse parâmetro é opcional. Um valor de `0` executa somente a correspondência inicial e nenhuma recursão. Se você omitir`maxDepth`, a pesquisa continuará recursivamente até que não haja mais documentos correspondentes.
- `depthField`: o nome do campo a ser adicionado a cada documento na `as` matriz que contém a profundidade de recursão na qual o documento foi encontrado. Esse parâmetro é opcional. A profundidade é representada como a `NumberLong` e começa em zero, então a pesquisa inicial corresponde a uma profundidade de`0`.
- `restrictSearchWithMatch`: um documento que especifica uma condição de correspondência adicional aplicada aos documentos da pesquisa. Esse parâmetro é opcional. A condição usa a mesma sintaxe de consulta como um `$match` estágio e não pode incluir expressões de agregação.

> **Note**
>
> O `$graphLookup` palco está sujeito a um limite de memória de 100 MB e não suporta`allowDiskUse:true`. Uma operação que excede esse limite retorna um erro. Se a agregação incluir outros estágios, a `allowDiskUse:true` opção permanecerá em vigor para esses estágios.

Comportamento

- Cross-collection traversal: `from` defina a coleção que você deseja pesquisar. Essa pode ser a mesma coleção em que o pipeline é executado ou uma coleção diferente no mesmo banco de dados.
- Ordenação dos resultados: os documentos na `as` matriz não são retornados em nenhuma ordem garantida.

Exemplo (MongoDB Shell)

O exemplo a seguir é usado `$graphLookup` para percorrer uma hierarquia de relatórios de funcionários e, para cada funcionário, colocar toda a cadeia de gerentes acima deles.

Os comandos a seguir, executados no shell mongosh, inserem exemplos de documentos de funcionários que estabelecem uma hierarquia de relatórios:

```
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" }
]);
```

A agregação a seguir retorna a cadeia de gerenciamento completa de cada funcionário em uma nova `reportingHierarchy` matriz:

```
db.employees.aggregate([
  {
    $graphLookup: {
      from: "employees",
      startWith: "$reportsTo",
      connectFromField: "reportsTo",
      connectToField: "name",
      as: "reportingHierarchy",
      depthField: "level"
    }
  }
]);
```

Essa operação retorna a seguinte saída:

```
[
  {
    _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 do funcionário ganha uma `reportingHierarchy` matriz que contém os documentos do gerente correspondentes recursivamente. O `level` campo registra a profundidade da recursão na qual cada gerente foi encontrado. Asya não tem `reportsTo` valor, então sua `reportingHierarchy` matriz está vazia.

Exemplo com maxDepth e restrict SearchWithMatch (MongoDB Shell)

O exemplo a seguir percorre uma rede de conexões entre pessoas e usuários `maxDepth` para limitar a recursão e incluir somente documentos que correspondam `restrictSearchWithMatch` a uma condição adicional. A pesquisa encontra as conexões de uma pessoa inicial, com até dois níveis de profundidade, mantendo apenas as pessoas cuja `hobbies` matriz contém`golf`.

Os comandos a seguir, executados no shell mongosh, inserem exemplos de documentos nos quais cada pessoa `friends` aponta para frente por meio de uma cadeia unidirecional de conexões:

```
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" ] }
]);
```

A agregação a seguir começa com Tanya, percorre sua rede de conexões com até dois níveis de profundidade e mantém apenas as pessoas que `golf` listam como hobby:

```
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 } }
]);
```

Essa operação retorna a seguinte saída:

```
[
  {
    _id: 1,
    name: 'Tanya',
    golfConnections: [
      { name: 'Carole' },
      { name: 'Joseph' },
      { name: 'Angelo' }
    ]
  }
]
```

Somente as conexões cuja `hobbies` matriz contém `golf` são incluídas e a travessia é interrompida após dois níveis de recursão. Shirley é excluída porque ela não lista`golf`, e Miguel é excluído porque está a três níveis de distância de Tanya, além do de. `maxDepth` 2 Como as conexões formam uma cadeia unidirecional que nunca aponta para Tanya, o documento inicial não é correspondido durante a pesquisa recursiva.

Exemplos de código

Para ver um exemplo de código que cria a hierarquia de relatórios de funcionários a partir do primeiro exemplo usando um driver MongoDB, escolha a guia do idioma que você deseja 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 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

O estágio `$group` de agregação no Amazon DocumentDB permite que você agrupe documentos por uma expressão especificada e realize várias operações acumulativas nos dados agrupados. Isso pode ser útil para tarefas como calcular totais, médias ou outras estatísticas com base nos dados agrupados.

Parâmetros

- `_id`: especifica a expressão pela qual os documentos de entrada devem ser agrupados. Isso pode ser um nome de campo, uma expressão computada ou uma combinação de ambos.
- `accumulator expressions`: (opcional) Uma ou mais expressões de acumulador que devem ser aplicadas aos dados agrupados. Essas expressões usam os operadores de acumulador mencionados acima.

Exemplo (MongoDB Shell)

O exemplo a seguir agrupa clientes por cidade e calcula o valor total do pedido para cada cidade.

Crie documentos de amostra

```
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 }] }
]);
```

Exemplo de consulta

```
db.customers.aggregate([
  {
    $group: {
      _id: "$city",
      totalOrders: { $sum: { $sum: "$orders.amount" } }
    }
  }
]);
```

Saída

```
[
  { _id: 'Chicago', totalOrders: 150 },
  { _id: 'Los Angeles', totalOrders: 450 },
  { _id: 'New York', totalOrders: 500 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$group` comando, escolha a guia do idioma que você deseja 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

O operador `$gt` de agregação compara dois valores e retorna `true` se o primeiro valor for maior que o segundo, caso contrário, retorna. `false`

Parâmetros

- `expression1`: o primeiro valor a ser comparado.
- `expression2`: O segundo valor a ser comparado.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$gt` operador para identificar produtos que excedem um limite de preço.

Crie documentos de amostra

```
db.products.insertMany([
  { _id: 1, name: "Laptop", price: 1200 },
  { _id: 2, name: "Mouse", price: 25 },
  { _id: 3, name: "Monitor", price: 400 }
]);
```

Exemplo de consulta

```
db.products.aggregate([
  {
    $project: {
      name: 1,
```

```
      price: 1,
      expensive: { $gt: ["$price", 100] }
    }
  }
]);
```

Saída

```
[
  { _id: 1, name: 'Laptop', price: 1200, expensive: true },
  { _id: 2, name: 'Mouse', price: 25, expensive: false },
  { _id: 3, name: 'Monitor', price: 400, expensive: true }
]
```

Exemplos de código

Para ver um exemplo de código para usar o operador de `$gt` agregação, escolha a guia do idioma que você deseja 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

O operador `$gte` de agregação compara dois valores e retorna `true` se o primeiro valor for maior ou igual ao segundo, caso contrário, retorna. `false`

Parâmetros

- `expression1`: o primeiro valor a ser comparado.
- `expression2`: O segundo valor a ser comparado.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$gte` operador para verificar se os alunos foram aprovados em um exame.

Crie documentos de amostra

```
db.students.insertMany([
  { _id: 1, name: "Alice", score: 85 },
  { _id: 2, name: "Bob", score: 60 },
  { _id: 3, name: "Charlie", score: 72 }
]);
```

Exemplo de consulta

```
db.students.aggregate([
  {
    $project: {
      name: 1,
      score: 1,
      passed: { $gte: ["$score", 70] }
    }
  }
]);
```

Saída

```
[
  { _id: 1, name: 'Alice', score: 85, passed: true },
  { _id: 2, name: 'Bob', score: 60, passed: false },
  { _id: 3, name: 'Charlie', score: 72, passed: true }
]
```

Exemplos de código

Para ver um exemplo de código para usar o operador de `$gte` agregação, escolha a guia do idioma que você deseja 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

O `$hour` operador extrai o componente de hora de um campo de data ou carimbo de data/hora.

Parâmetros

- `dateExpression`: a data na qual o operador é aplicado. Isso deve ser resolvido para uma data BSON válida (por exemplo, um campo como $createdAt ou uma data literal).

Um parâmetro também pode ser especificado como um documento no seguinte formato:

{data:<dateExpression>, fuso horário:<timezoneExpression>}

Isso permite aplicar operações de data com reconhecimento de fuso horário.

```
- `<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.
```

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$hour` operador para extrair o componente de hora de um campo de data e agrupar os dados adequadamente.

Crie documentos de amostra

```
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") }
]);
```

Exemplo de consulta

```
db.events.aggregate([
  {
    $project: {
      hour: { $hour: "$timestamp" }
    }
  },
  {
    $group: {
      _id: "$hour",
      count: { $sum: 1 }
    }
  },
  {
    $sort: { _id: 1 }
  }
]);
```

Saída

```
[
  { "_id": 8, "count": 1 },
  { "_id": 10, "count": 1 },
  { "_id": 12, "count": 1 },
  { "_id": 16, "count": 1 },
  { "_id": 23, "count": 1 }
]
```

Essa consulta agrupa os eventos pelo componente horário do `timestamp` campo e conta o número de eventos para cada hora.

Exemplos de código

Para ver um exemplo de código para usar o `$hour` comando, escolha a guia do idioma que você deseja 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

O `$ifNull` operador é usado para retornar um valor especificado se a expressão de entrada for avaliada como nula ou indefinida. Esse operador pode ser útil em cenários em que você deseja fornecer um valor padrão ou lidar com null/undefined casos.

Parâmetros

- `expression`: A expressão a ser avaliada.
- `replacement`: o valor a ser retornado se for <`expression`> avaliado como nulo ou indefinido.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$ifNull` operador para fornecer um valor padrão quando o `name` campo é nulo ou indefinido.

Crie documentos de amostra

```
db.users.insertMany([
  { _id: 1, name: "John" },
  { _id: 2, name: null },
  { _id: 3 }
]);
```

Exemplo de consulta

```
db.users.aggregate([
  {
    $project: {
      _id: 1,
      name: { $ifNull: ["$name", "No Name"] }
    }
  }
]);
```

Saída

```
[
  { "_id": 1, "name": "John" },
  { "_id": 2, "name": "No Name" },
  { "_id": 3, "name": "No Name" }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$ifNull` comando, escolha a guia do idioma que você deseja 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

O operador `$in` de agregação verifica se existe um valor especificado em uma matriz. Ele retorna `true` se o valor for encontrado na matriz e de `false` outra forma.

Parâmetros

- `value`: O valor a ser pesquisado.
- `array`: A matriz na qual pesquisar.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$in` operador para verificar se existe uma habilidade específica no conjunto de habilidades de cada funcionário.

Crie documentos de amostra

```
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"] }
]);
```

Exemplo de consulta

```
db.employees.aggregate([
  {
    $project: {
      name: 1,
      hasPython: { $in: ["Python", "$skills"] }
    }
  }
]);
```

Saída

```
[
  { _id: 1, name: 'Sarah', hasPython: true },
  { _id: 2, name: 'Mike', hasPython: false },
  { _id: 3, name: 'Emma', hasPython: true }
]
```

Exemplos de código

Para ver um exemplo de código para usar o operador de `$in` agregação, escolha a guia do idioma que você deseja 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

O `$indexOfArray` operador no Amazon DocumentDB é usado para encontrar o índice da primeira ocorrência de um elemento especificado em uma matriz. Esse operador retorna a posição do índice

com base em zero do primeiro elemento na matriz que corresponde ao valor especificado. Se o valor não for encontrado, ele retornará -1.

Parâmetros

- `array`: A matriz a ser pesquisada.
- `value`: o valor a ser pesquisado na matriz.
- `start`: (opcional) A posição na matriz a partir da qual iniciar a pesquisa. O valor padrão é 0.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o OfArray operador $index para encontrar o índice da primeira ocorrência do elemento “mango” na matriz “fruits” de cada documento.

Crie documentos de amostra

```
db.collection.insertMany([
  { _id: 1, fruits: ["apple", "banana", "cherry", "durian"] },
  { _id: 2, fruits: ["mango", "orange", "pineapple"] },
  { _id: 3, fruits: ["kiwi", "lemon", "mango"] }
]);
```

Exemplo de consulta

```
db.collection.aggregate([
  {
    $project: {
      _id: 1,
      fruitIndex: { $indexOfArray: ["$fruits", "mango"] }
    }
  }
]);
```

Saída

```
{ "_id" : 1, "fruitIndex" : 1 }
{ "_id" : 2, "fruitIndex" : 0 }
{ "_id" : 3, "fruitIndex" : 2 }
```

Exemplos de código

Para ver um exemplo de código para usar o `$indexOfArray` comando, escolha a guia do idioma que você deseja 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

O OfBytes operador $index no Amazon DocumentDB é usado para encontrar o índice inicial de uma substring dentro de uma string, com base nas posições dos bytes dos caracteres. Isso pode ser útil ao trabalhar com dados de texto que podem conter caracteres de vários bytes, como aqueles encontrados em scripts não latinos.

Parâmetros

- `string`: A string de entrada a ser pesquisada.
- `substring`: a substring a ser pesquisada na string de entrada.
- `[<start>]`: (opcional) A posição inicial (com base em zero) da pesquisa. Se não for especificado, a pesquisa começa no início da string.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso de `$indexOfBytes` para encontrar o índice do primeiro caractere de hífen em um conjunto de cadeias de caracteres representando a localização das mesas.

Crie documentos de amostra

```
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" }
]);
```

Exemplo de consulta

```
db.people.aggregate([
  { $project: { stateLocation: { $indexOfBytes: ["$Desk", "-"] } } }
]);
```

Saída

```
{ "_id" : 1, "stateLocation" : 11 }
{ "_id" : 2, "stateLocation" : 6 }
{ "_id" : 3, "stateLocation" : 7 }
{ "_id" : 4, "stateLocation" : 8 }
```

Exemplos de código

Para ver um exemplo de código para usar o `$indexOfBytes` comando, escolha a guia do idioma que você deseja 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

O `$indexOfCP` operador no Amazon DocumentDB é usado para encontrar o índice, em pontos de código (CP), da primeira ocorrência de uma substring especificada em uma expressão de string. Isso pode ser útil ao analisar e extrair conteúdo de campos de string.

Parâmetros

- `string expression`: A string a ser pesquisada.
- `substring`: A substring a ser pesquisada.
- `[<start>]`: (opcional) A posição para iniciar a pesquisa (índice baseado em zero). O padrão é 0.

Exemplo (MongoDB Shell)

Neste exemplo, usamos o `$indexOfCP` operador para encontrar o índice da primeira ocorrência do caractere hífen - no campo Mesa de cada documento.

Crie documentos de amostra

```
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"}
])
```

Exemplo de consulta

```
db.people.aggregate([
  { $project: { stateLocation: { $indexOfCP: [ "$Desk", "-"] } } }
])
```

Saída

```
{ "_id" : 1, "stateLocation" : 10 }
{ "_id" : 2, "stateLocation" : 6 }
{ "_id" : 3, "stateLocation" : 7 }
{ "_id" : 4, "stateLocation" : 8 }
```

Exemplos de código

Para ver um exemplo de código para usar o `$indexOfCP` comando, escolha a guia do idioma que você deseja 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

O estágio `$indexStats` de agregação no Amazon DocumentDB fornece uma visão sobre o uso de índices em uma coleção. Esse operador permite monitorar os padrões de acesso de seus índices, o que pode ajudá-lo a tomar decisões informadas sobre gerenciamento e otimização de índices.

Parâmetros

Nenhum

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$indexStats` operador para analisar o uso do índice em uma coleção do Amazon DocumentDB.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.grocery.aggregate([
  { $indexStats: {} }
]);
```

Saída

```
[
  {
    "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")
    }
  }
]
```

Neste exemplo, o `$indexStats` operador mostra que o `_id_` índice foi acessado 5 vezes e o `product_1` índice foi acessado 10 vezes desde a última reinicialização ou reinicialização do servidor.

Exemplos de código

Para ver um exemplo de código para usar o `$indexStats` comando, escolha a guia do idioma que você deseja 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

O `$isArray` operador no Amazon DocumentDB é usado para verificar se um campo em um documento é uma matriz. Esse operador pode ser útil em pipelines de agregação e expressões condicionais para lidar com campos do tipo matriz.

Parâmetros

- `field`: o caminho do campo para verificar se é uma matriz.

Exemplo (MongoDB Shell)

Este exemplo demonstra como usar o `$isArray` operador para identificar documentos nos quais o campo “inventário” é uma matriz.

Crie documentos de amostra

```
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}]}
]);
```

Exemplo de consulta

```
db.videos.aggregate([
  {
    $match: {
      $isArray: "$inventory"
    }
  },
  {
    $project: {
      _id: 1,
      name: 1,
```

```
      "inventory.location": 1,
      "inventory.count": 1
    }
  }
]).pretty();
```

Saída

```
{
  "_id": 4,
  "name": "Bravemind",
  "inventory": [
    {
      "location": "Mason City",
      "count": 250
    },
    {
      "location": "Des Moines",
      "count": 1000
    },
    {
      "location": "Ames",
      "count": 500
    }
  ]
}
```

Exemplos de código

Para ver um exemplo de código para usar o `$isArray` comando, escolha a guia do idioma que você deseja 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()
```

$isNumber

Novo na versão 8.0.1.

O `$isNumber` operador no Amazon DocumentDB retorna um booleano indicando se uma determinada expressão é resolvida para um tipo numérico (inteiro, decimal, duplo, longo). Observe que as matrizes não são consideradas numéricas, mesmo que todos os elementos na matriz sejam números. Por exemplo, `$isNumber` aplicado aos `[1, 2, 3]` retornos de valor`false`.

Parâmetros

- `expression`: qualquer expressão para verificar se ela é resolvida para um tipo numérico.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$isNumber` operador para verificar se os valores dos campos são numéricos.

Crie documentos de amostra

```
db.data.insertMany([
  {_id: 1, value: 42},
  {_id: 2, value: "hello"},
  {_id: 3, value: 3.14},
  {_id: 4, value: null}
]);
```

Exemplo de consulta

```
db.data.aggregate([
```

```
  { $project: { isNum: { $isNumber: "$value" } } }
]);
```

Saída

```
[
  {_id: 1, isNum: true},
  {_id: 2, isNum: false},
  {_id: 3, isNum: true},
  {_id: 4, isNum: false}
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$isNumber` operador, escolha a guia do idioma que você deseja 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

O `$isoDayOfWeek` operador no Amazon DocumentDB retorna o dia ISO da semana de uma data como um valor inteiro. O sistema ISO de datas semanais define cada semana começando na segunda-feira e terminando no domingo, com a semana 1 sendo a semana que contém a primeira quinta-feira do ano.

Parâmetros

- `expression`: a expressão de data para a qual retornar o dia ISO da semana.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$isoDayOfWeek` operador para recuperar o dia ISO da semana para um conjunto de documentos de eventos.

Crie documentos de amostra

```
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") }
]);
```

Exemplo de consulta

```
db.events.aggregate([
  { $project: {
    _id: 1,
    eventDate: 1,
    isoDayOfWeek: { $isoDayOfWeek: "$eventDate" }
  }}
]);
```

Saída

```
[
  { "_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 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$isoDayOfWeek` comando, escolha a guia do idioma que você deseja 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

O `$isoWeek` operador no Amazon DocumentDB retorna o número ISO da semana de uma data. O sistema de data da semana ISO é uma forma de numerar as semanas em um ano, em que a primeira semana de um novo ano é a semana que contém a primeira quinta-feira desse ano. Isso é diferente do calendário gregoriano, em que a primeira semana de um novo ano é a semana que contém 1º de janeiro.

Parâmetros

Nenhum

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$isoWeek` operador para recuperar o número da semana ISO de uma determinada data.

Crie documentos de amostra

```
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") }
])
```

Exemplo de consulta

```
db.dates.aggregate([
  {
    $project: {
      _id: 1,
      isoWeek: { $isoWeek: "$date" }
    }
  }
])
```

Saída

```
[
  { "_id": 1, "isoWeek": 52 },
  { "_id": 2, "isoWeek": 52 },
  { "_id": 3, "isoWeek": 1 }
```

```
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$isoWeek` comando, escolha a guia do idioma que você deseja 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

O `$isoWeekYear` operador no Amazon DocumentDB retorna o número do ano da semana ISO 8601 para uma determinada data. O número ISO do ano semanal difere do ano civil gregoriano, pois o ano da semana pode ser diferente do ano civil, especialmente no início e no final do ano.

Parâmetros

- `expression`: a expressão de data para a qual retornar o número do ano da semana ISO 8601.

Exemplo (MongoDB Shell)

Este exemplo demonstra como usar o `$isoWeekYear` operador para recuperar o ano da semana ISO 8601 para o campo de data de cada documento na coleção de eventos.

Crie documentos de amostra

```
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") }
]);
```

Exemplo de consulta

```
db.events.aggregate([
```

```
  { $project: {
    name: 1,
    isoWeekYear: { $isoWeekYear: "$date" }
  }}
]);
```

Saída

```
[
  { "_id" : 1, "name" : "Event 1", "isoWeekYear" : 2023 },
  { "_id" : 2, "name" : "Event 2", "isoWeekYear" : 2023 },
  { "_id" : 3, "name" : "Event 3", "isoWeekYear" : 2023 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$isoWeekYear` comando, escolha a guia do idioma que você deseja 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

O `$last` operador no Amazon DocumentDB é usado para retornar o último elemento em uma matriz que corresponda aos critérios da consulta. É particularmente útil para recuperar o elemento mais recente ou o último em uma matriz que satisfaça uma condição específica.

Parâmetros

- `expression`: A expressão para corresponder aos elementos da matriz.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$last` operador em combinação com `$filter` para recuperar o último elemento de uma matriz que atenda a uma condição específica (por exemplo, assunto é 'ciência').

Crie documentos de amostra

```
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 }
    ]
  }
]);
```

Exemplo de consulta

```
db.collection.aggregate([
    { $match: { name: "John" } },
    {
      $project: {
        name: 1,
        lastScienceScore: {
          $last: {
            $filter: {
              input: "$scores",
              as: "score",
              cond: { $eq: ["$$score.subject", "science"] }
```

```
            }
          }
        }
      }
    }
  ]);
```

Saída

```
[
  {
    _id: 1,
    name: 'John',
    lastScienceScore: { subject: 'science', score: 90 }
  }
]
```

Exemplo de uso do operador de janela (MongoDB Shell)

Novo na versão 8.0.2.

O `$last` operador também pode ser usado como operador de janela no `$setWindowFields` palco. Nesse contexto, ele retorna o valor `expression` do último documento em cada janela. Você especifica o operador no `output` campo e, opcionalmente, define os limites da janela com um `window` documento.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

O exemplo a seguir particiona os documentos por`ticker`, classifica cada partição por `hour` e retorna o último preço (mais recente) registrado em cada partição.

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
      sortBy: { hour: 1 },
      output: {
        lastPrice: {
          $last: "$price",
          window: { documents: ["unbounded", "unbounded"] }
        }
      }
    }
  }
]);
```

Saída

```
[
  { "_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 é aumentado com `lastPrice` o preço do último documento em sua janela (o mais recente `hour` na partição).

Exemplos de código

Para ver um exemplo de código para usar o `$last` operador, escolha a guia do idioma que você deseja usar. Os exemplos a seguir mostram o uso da expressão (in`$project`) e o uso do operador de janela (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()
```

$LastN

Novo na versão 8.0.1.

O `$lastN` operador no Amazon DocumentDB retorna os últimos N elementos. Quando usado como acumulador em um `$group` estágio, ele retorna uma matriz dos últimos N valores em cada grupo. Quando usado como um operador de expressão de matriz, ele retorna os últimos N elementos de uma matriz.

Parâmetros

- `input`: a expressão que resolve para o campo ou matriz do qual retornar valores.
- n: um número inteiro positivo que especifica quantos valores devem ser retornados. Quando usado como acumulador em um `$group` estágio, também n pode ser uma expressão, desde que seja resolvido para um número inteiro positivo com base no campo do grupo. `_id`

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$lastN` acumulador para recuperar as duas últimas quantidades de cada item durante a agregação.

> **Note**
>
> `$lastN`seleciona valores na ordem em que os documentos chegam ao `$group` estágio. Para retornar os últimos N valores de uma ordem específica (por exemplo, por data ou pontuação), adicione um `$sort` estágio anterior`$group`.

Crie documentos de amostra

```
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") }
]);
```

Exemplo de consulta

```
db.sales.aggregate([
  { $group: { _id: "$item", lastTwoQuantities: { $lastN: { input: "$quantity", n:
 2 } } } }
]);
```

Saída

```
[
  { "_id": "abc", "lastTwoQuantities": [5, 8] },
  { "_id": "xyz", "lastTwoQuantities": [7, 3] }
]
```

Exemplo de uso de expressão (MongoDB Shell)

O `$lastN` operador também pode ser usado como uma expressão dentro de um `$project` estágio para retornar os últimos N elementos de um campo de matriz.

Crie documentos de amostra

```
db.inventory.insertMany([
  { _id: 1, item: "abc", tags: ["red", "green", "blue", "yellow", "purple"] },
  { _id: 2, item: "xyz", tags: ["alpha", "beta", "gamma"] }
]);
```

Exemplo de consulta

```
db.inventory.aggregate([
  { $project: {
      lastThreeTags: { $lastN: { input: "$tags", n: 3 } }
    }}
]);
```

Saída

```
[
  { "_id": 1, "lastThreeTags": ["blue", "yellow", "purple"] },
  { "_id": 2, "lastThreeTags": ["alpha", "beta", "gamma"] }
]
```

Exemplo de uso do operador de janela (MongoDB Shell)

Novo na versão 8.0.2.

O `$lastN` operador também pode ser usado como operador de janela no `$setWindowFields` palco. Nesse contexto, ele retorna uma matriz de até os últimos n valores dos documentos em cada janela. Você especifica o operador no `output` campo e, opcionalmente, define os limites da janela com um `window` documento.

> **Note**
>
> Quando usado como operador de janela em`$setWindowFields`, `$lastN` é limitado a 100 MB de dados intermediários. Uma operação que excede esse limite retorna um erro.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

O exemplo a seguir particiona os documentos por`ticker`, classifica cada partição por `hour` e retorna os dois preços mais recentes vistos desde o início da partição até o documento atual.

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
```

```
      sortBy: { hour: 1 },
      output: {
        lastTwoSoFar: {
          $lastN: { input: "$price", n: 2 },
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Saída

```
[
  { "_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 é aumentado com `lastTwoSoFar` os dois últimos `price` valores em sua janela (em `hour` ordem).

Exemplos de código

Para ver um exemplo de código para usar o `$lastN` operador, escolha a guia do idioma que você deseja usar. Os exemplos a seguir mostram o uso do acumulador (in`$group`), o uso da expressão (in`$project`) e o uso do operador de janela (in`$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

O `$let` operador no Amazon DocumentDB é usado para vincular variáveis a valores e usar essas variáveis na expressão. Ele permite que você defina variáveis locais que podem ser usadas em expressões subsequentes no mesmo estágio do pipeline de agregação.

Parâmetros

- `vars`: um objeto que define as variáveis a serem usadas na expressão.
- `in`: a expressão na qual as variáveis definidas no parâmetro vars são usadas.

Exemplo (MongoDB Shell)

Este exemplo demonstra o uso do `$let` operador para calcular a área de um retângulo.

Crie documentos de amostra

```
db.shapes.insertMany([
  { name: "Rectangle 1", length: 5, width: 3 },
  { name: "Rectangle 2", length: 7, width: 4 },
  { name: "Rectangle 3", length: 6, width: 2 }
]);
```

Exemplo de consulta

```
db.shapes.aggregate([
  {
    $project: {
      name: 1,
      area: {
        $let: {
          vars: {
            length: "$length",
            width: "$width"
```

```
          },
          in: {
            $multiply: ["$$length", "$$width"]
          }
        }
      }
    }
  }
])
```

Saída

```
[
  {
    "_id": ObjectId("6161e5b1a3eba3c7f2960d03"),
    "name": "Rectangle 1",
    "area": 15
  },
  {
    "_id": ObjectId("6161e5b1a3eba3c7f2960d04"),
    "name": "Rectangle 2",
    "area": 28
  },
  {
    "_id": ObjectId("6161e5b1a3eba3c7f2960d05"),
    "name": "Rectangle 3",
    "area": 12
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$let` comando, escolha a guia do idioma que você deseja 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

O `$limit` operador no Amazon DocumentDB é usado para restringir o número de documentos retornados por uma consulta. É semelhante ao operador `$limit` MongoDB, mas há algumas considerações específicas ao usá-lo com o Amazon DocumentDB.

No Amazon DocumentDB, o `$limit` operador é útil para paginação, onde você deseja recuperar um subconjunto do total de documentos correspondentes. Ele permite que você controle o número de documentos retornados em cada resposta, melhorando o desempenho e reduzindo a quantidade de dados transferidos pela rede.

Parâmetros

- `limit`: O número máximo de documentos a serem devolvidos. Esse deve ser um valor inteiro não negativo.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$limit` operador para retornar no máximo um documento que corresponda ao filtro especificado.

Crie documentos de amostra

```
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" }
]);
```

Exemplo de consulta

```
db.test.createIndex({ comments: "text" });

db.test.find({
  $and: [
    { star_rating: 5 },
    { $text: { $search: "couch" } }
  ]
}).limit(1);
```

Saída

```
[ { _id: 2, star_rating: 5, comments: 'comfortable couch' } ]
```

Exemplos de código

Para ver um exemplo de código para usar o `$limit` comando, escolha a guia do idioma que você deseja 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

Novo na versão 8.0.1.

O estágio `$listSearchIndexes` de agregação no Amazon DocumentDB retorna informações sobre os índices de pesquisa existentes em uma coleção. Deve ser o primeiro estágio em um pipeline de agregação.

Parâmetros

- `name`: (opcional) O nome do índice de pesquisa sobre o qual retornar informações. Se omitido, todos os índices de pesquisa na coleção serão retornados.

Campos de saída

Cada documento devolvido contém os seguintes campos:

- `name`: O nome do índice de pesquisa.
- `status`: O estado de construção do índice. Um dos `READY` (criado e válido), `BUILDING` (uma compilação simultânea está em andamento) ou `FAILED` (a construção do índice não foi concluída com êxito).

- `queryable`: um booleano que indica se o índice pode ser usado atualmente para atender consultas. Isso é `false` para índices ocultos (que permanecemREADY) e para `FAILED` índices. Um índice oculto é o caso em que `status` estáREADY, mas `queryable` está`false`.
- `latestDefinitionVersion`: Um documento contendo `version` (a versão do formato do índice) e `createdAt` (a hora em que o índice foi criado).

Sintaxe

```
db.collection.aggregate([
  { $listSearchIndexes: {} }
])

// Or filter by name:
db.collection.aggregate([
  { $listSearchIndexes: { name: "mySearchIndex" } }
])
```

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$listSearchIndexes` estágio para listar todos os índices de pesquisa em uma coleção.

Exemplo de consulta

```
db.movies.aggregate([
  { $listSearchIndexes: {} }
]);
```

Saída

```
[
  {
    "name": "default",
    "status": "READY",
    "queryable": true,
    "latestDefinitionVersion": {
      "version": 2,
      "createdAt": ISODate("2026-05-11T21:12:57.974Z")
    }
  }
```

```
]
```

Um índice oculto permanece, `READY` mas relata`queryable: false`, porque não pode ser usado para atender consultas enquanto estiver oculto:

```
[
  {
    "name": "myHiddenIndex",
    "status": "READY",
    "queryable": false,
    "latestDefinitionVersion": {
      "version": 2,
      "createdAt": ISODate("2026-05-11T21:12:57.974Z")
    }
  }
]
```

Para filtrar por um nome de índice específico:

```
db.movies.aggregate([
  { $listSearchIndexes: { name: "default" } }
]);
```

Exemplos de código

Para ver um exemplo de código para usar o `$listSearchIndexes` estágio, escolha a guia do idioma que você deseja 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

O `$literal` operador no Amazon DocumentDB é usado para representar um valor literal em um estágio do pipeline de agregação. Ele permite que você inclua um valor específico, como um número, string ou booleano, sem interpretá-lo como uma referência de campo ou expressão.

Esse operador é particularmente útil quando você precisa incluir um valor literal como parte de um pipeline de agregação mais complexo, como ao criar filtros de consulta dinâmicos ou realizar cálculos.

Parâmetros

Nenhum

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$literal` operador para incluir um valor literal em um pipeline de agregação. O `$literal` operador é usado para incluir o valor 18 como um valor literal na expressão $gt. Isso permite que o pipeline de agregação compare o campo de idade com o valor literal 18 para determinar se a pessoa é adulta.

Crie documentos de amostra

```
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" }
]);
```

Exemplo de consulta

```
db.collection.aggregate([
  {
    $project: {
```

```
      name: 1,
      age: 1,
      city: 1,
      isAdult: { $gt: ["$age", { $literal: 18 }] }
    }
  }
]);
```

Saída

```
[
  {
    "_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
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$literal` comando, escolha a guia do idioma que você deseja 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

Novo a partir da versão 4.0.

O `$ln` operador no Amazon DocumentDB calcula o logaritmo natural (base e) de um número especificado. Ele retorna o logaritmo do número para a base e.

Parâmetros

- `expression`: O número para o qual o logaritmo natural será calculado.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$log` operador para calcular o logaritmo natural de um número.

Crie documentos de amostra

```
db.numbers.insertMany([
  { _id: 1, value: 10 },
  { _id: 2, value: 100 },
  { _id: 3, value: 1000 }
]);
```

Exemplo de consulta

```
db.numbers.aggregate([
  { $project: {
    _id: 1,
    naturalLog: { $ln: "$value" }
  }}
]);
```

Saída

```
[
  { "_id" : 1, "naturalLog" : 2.302585092994046 },
  { "_id" : 2, "naturalLog" : 4.605170185988092 },
  { "_id" : 3, "naturalLog" : 6.907755278982137 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$ln` comando, escolha a guia do idioma que você deseja 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

Novo a partir da versão 4.0.

O `$log` operador no Amazon DocumentDB calcula o logaritmo natural de um número. Ele retorna o logaritmo de base e do número especificado.

Parâmetros

- `expression`: O número para o qual o logaritmo natural será calculado.
- `base`: Valor base para calcular o log.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$log` operador para calcular o logaritmo natural de um número.

Crie documentos de amostra

```
db.numbers.insertMany([
  { _id: 1, value: 10 },
```

```
  { _id: 2, value: 100 },
  { _id: 3, value: 1000 }
]);
```

Exemplo de consulta

```
db.numbers.aggregate([
  { $project: {
    _id: 1,
    naturalLog: { $log: ["$value", 10] }
  }}
]);
```

Saída

```
[
  { "_id" : 1, "naturalLog" : 1 },
  { "_id" : 2, "naturalLog" : 2 },
  { "_id" : 3, "naturalLog" : 2.9999999999999996 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$log` comando, escolha a guia do idioma que você deseja 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

Novo a partir da versão 4.0.

O `$log10` operador no Amazon DocumentDB é usado para calcular o logaritmo de base 10 de um número. É útil para realizar cálculos logarítmicos em campos numéricos dentro do pipeline de agregação.

Parâmetros

- `expression`: a expressão numérica para a qual o logaritmo de base 10 deve ser calculado.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$log10` operador para calcular o logaritmo de base 10 de um campo numérico.

Crie documentos de amostra

```
db.numbers.insertMany([
  { _id: 1, value: 1 },
  { _id: 2, value: 10 },
  { _id: 3, value: 100 },
  { _id: 4, value: 1000 }
]);
```

Exemplo de consulta

```
db.numbers.aggregate([
  {
    $project: {
      _id: 1,
      log10Value: { $log10: "$value" }
    }
  }
]);
```

Saída

```
[
  { "_id": 1, "log10Value": 0 },
  { "_id": 2, "log10Value": 1 },
  { "_id": 3, "log10Value": 2 },
  { "_id": 4, "log10Value": 3 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$log10` comando, escolha a guia do idioma que você deseja 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

O operador `$lt` de agregação compara dois valores e retorna `true` se o primeiro valor for menor que o segundo, caso contrário, retorna. `false`

Parâmetros

- `expression1`: o primeiro valor a ser comparado.
- `expression2`: O segundo valor a ser comparado.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$lt` operador para identificar itens com baixo estoque.

Crie documentos de amostra

```
db.warehouse.insertMany([
  { _id: 1, item: "Bolts", stock: 5 },
  { _id: 2, item: "Nuts", stock: 25 },
  { _id: 3, item: "Screws", stock: 8 }
]);
```

Exemplo de consulta

```
db.warehouse.aggregate([
  {
    $project: {
      item: 1,
      stock: 1,
      lowStock: { $lt: ["$stock", 10] }
    }
  }
]);
```

Saída

```
[
  { _id: 1, item: 'Bolts', stock: 5, lowStock: true },
  { _id: 2, item: 'Nuts', stock: 25, lowStock: false },
  { _id: 3, item: 'Screws', stock: 8, lowStock: true }
]
```

Exemplos de código

Para ver um exemplo de código para usar o operador de `$lt` agregação, escolha a guia do idioma que você deseja 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

O operador `$lte` de agregação compara dois valores e retorna `true` se o primeiro valor for menor ou igual ao segundo, caso contrário, retorna. `false`

Parâmetros

- `expression1`: o primeiro valor a ser comparado.
- `expression2`: O segundo valor a ser comparado.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$lte` operador para identificar itens econômicos.

Crie documentos de amostra

```
db.menu.insertMany([
  { _id: 1, dish: "Salad", price: 8 },
  { _id: 2, dish: "Pasta", price: 12 },
  { _id: 3, dish: "Soup", price: 6 }
]);
```

Exemplo de consulta

```
db.menu.aggregate([
  {
    $project: {
      dish: 1,
      price: 1,
      affordable: { $lte: ["$price", 10] }
    }
  }
]);
```

Saída

```
[
  { _id: 1, dish: 'Salad', price: 8, affordable: true },
  { _id: 2, dish: 'Pasta', price: 12, affordable: false },
  { _id: 3, dish: 'Soup', price: 6, affordable: true }
]
```

Exemplos de código

Para ver um exemplo de código para usar o operador de `$lte` agregação, escolha a guia do idioma que você deseja 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

O estágio `$lookup` de agregação no Amazon DocumentDB permite que você realize uma junção externa esquerda entre duas coleções. Essa operação permite combinar dados de várias coleções com base nos valores de campo correspondentes. É particularmente útil quando você precisa incorporar dados de coleções relacionadas aos resultados da consulta.

Parâmetros

- `from`: O nome da coleção com a qual realizar a junção.
- `localField`: O campo dos documentos de entrada a serem comparados com `foreignField` o.
- `foreignField`: O campo dos documentos na `from` coleção a ser comparado com `localField` o.
- `as`: o nome do novo campo a ser adicionado aos documentos de saída contendo os documentos correspondentes da `from` coleção.
- `pipeline`: Opcional se `localField` e `foreignField` forem especificados. Um pipeline de agregação para ser executado na `from` coleção. O pipeline retorna documentos da `from` coleção. Para retornar todos os documentos, especifique um pipeline vazio`[]`. O pipeline não pode incluir os `$merge` estágios `$out` ou. Consulte Condições de junção e subconsultas correlacionadas (MongoDB Shell).
- `let`: opcional. Um documento que define variáveis a serem usadas nas `pipeline` etapas. Use as expressões variáveis para acessar os campos dos documentos de entrada que estão unidos

aopipeline. Para referenciar uma variável nos estágios do pipeline, use a "$$<variable>" sintaxe. Consulte Condições de junção e subconsultas correlacionadas (MongoDB Shell).

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra uma $lookup operação simples que une dados da orders coleta à customers coleção.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.customers.aggregate([
  {
    $lookup: {
      from: "orders",
      localField: "_id",
      foreignField: "customer_id",
      as: "orders"
    }
  }
]);
```

Saída

```
[
  {
    _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 } ]
  }
]
```

Condições de junção e subconsultas correlacionadas (MongoDB Shell)

Novo na versão 8.0.2.

Além de uma única igualdade de igualdade, `$lookup` no Amazon DocumentDB é possível executar um pipeline de agregação na coleção unida. Isso suporta subconsultas não correlacionadas, que executam a mesma subconsulta para cada documento de entrada, e subconsultas correlacionadas, que usam variáveis definidas nos campos de cada documento de entrada para retornar resultados correlacionados a esse documento.

Sintaxe

Para executar uma subconsulta correlacionada, defina variáveis dos campos do documento de entrada com `let` e faça referência a elas nas `pipeline` etapas:

```
{
  $lookup: {
    from: <collection to join>,
    let: { <var_1>: <expression>, ..., <var_n>: <expression> },
    pipeline: [ <pipeline to run on the joined collection> ],
    as: <output array field>
  }
}
```

Parâmetros

- `let`: um documento que define variáveis dos campos dos documentos de entrada. Para referenciar uma variável nos estágios do pipeline, use a "`$$<variable>`" sintaxe. As variáveis podem ser acessadas pelos estágios do`pipeline`, incluindo `$lookup` estágios adicionais

aninhados no`pipeline`. Um `$match` estágio requer o uso do `$expr` operador para acessar as variáveis. Omita `let` para executar uma subconsulta não correlacionada.

- `pipeline`: O pipeline de agregação a ser executado na `from` coleção. O pipeline não pode acessar diretamente os campos dos documentos de entrada; em vez disso, defina variáveis `let` e faça referência às variáveis. Para retornar todos os documentos, especifique um pipeline vazio`[]`.

Os operadores `$eq`$gt`,`$gte`,`$lt`, e de `$lte` comparação colocados em um `$expr` operador dentro de um `$match` estágio podem usar um índice na `from` coleção quando comparam um campo da coleção unida com uma `let` variável ou constante.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

O exemplo a seguir une cada cliente com seus pedidos e mantém somente os pedidos que `total` são maiores ou iguais ao `min_total` valor do próprio cliente. A condição de junção e o filtro por cliente são as duas `let` variáveis de referência em: `$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"
    }
  }
]);
```

Saída

```
[
  { _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, a subconsulta é avaliada com `min_total` os valores `_id` e valores desse cliente, de modo que cada documento receba seu próprio resultado correlacionado.

Subconsultas correlacionadas usando sintaxe concisa (MongoDB Shell)

Novo na versão 8.0.2.

O Amazon DocumentDB também oferece suporte à sintaxe concisa de subconsultas correlacionadas, que combina a correspondência de igualdade em e com a. `localField` `foreignField` `pipeline` A sintaxe concisa elimina a exigência de expressar a correspondência de igualdade dentro de um `$expr` operador em um `$match` estágio.

Sintaxe

```
{
  $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>
  }
```

```
}
```

Exemplo de consulta

O exemplo a seguir retorna os mesmos resultados do exemplo anterior de subconsulta correlacionada. A igualdade entre `_id` e `customer_id` é expressa com `localField` e `foreignField` em vez de uma `$expr` condição:

```
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"
    }
  }
]);
```

Saída

```
[
  { _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: [] }
]
```

Estágios suportados em subconsultas correlacionadas

Quando um `$lookup` pipeline usa`let`, os seguintes estágios podem fazer referência às "`$$<variable>`" variáveis:

- `$match`(usando o `$expr` operador)
- `$project`
- `$addFields`
- `$set`

- `$group`
- `$bucket`
- `$bucketAuto`
- `$sortByCount`
- `$lookup` (aninhada)
- `$graphLookup`

Estágios que não avaliam expressões variáveis, como`$sort`,`$skip`,`$limit`,`$count`,`$sample`, e `$unwind`$unset, também podem ser usados no pipeline, mas não podem referenciar as variáveis.

Os seguintes estágios não são suportados em um `$lookup` pipeline que usa`let`: `$facet`$redact,`$replaceRoot`, `$replaceWith`$geoNear,`$setWindowFields`,`$fill`, e `$match` com`$text`.

Restrições

- `pipeline`Não podem incluir os `$merge` estágios `$out` ou.

Exemplos de código

Para ver um exemplo de código para usar o `$lookup` comando, escolha a guia do idioma que você deseja 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()
```

Exemplos de códigos de subconsulta correlacionados

Para ver um exemplo de código para usar o `$lookup` estágio com uma subconsulta correlacionada (usando `let` e`pipeline`), escolha a guia do idioma que você deseja 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

Novo a partir da versão 4.0.

Não é compatível com o cluster elástico.

O `$ltrim` operador no Amazon DocumentDB é usado para remover caracteres iniciais de uma string. Por padrão, ele remove os caracteres de espaço em branco iniciais, mas você também pode especificar um conjunto de caracteres a serem removidos passando o argumento chars.

Parâmetros

- `input`: a string de entrada da qual remover os caracteres de espaço em branco iniciais.
- `chars`: (opcional) Para remover caracteres específicos.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso de `$ltrim` para remover caracteres especificados (” *”) do início de uma string.

Crie documentos de amostra

```
db.collection.insertMany([
  { name: " *John Doe", age: 30 },
  { name: "Jane Doe*", age: 25 },
  { name: "  Bob Smith  ", age: 35 }
]);
```

Exemplo de consulta

```
db.collection.aggregate([
  {
    $project: {
      _id: 0,
      name: {
        $ltrim: { input: "$name", chars: " *" }
      },
      age: 1
    }
  }
]);
```

Saída

```
[
  { "name": "John Doe", "age": 30 },
  { "name": "Jane Doe ", "age": 25 },
  { "name": "Bob Smith  ", "age": 35 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$ltrim` comando, escolha a guia do idioma que você deseja 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

O `$map` operador no Amazon DocumentDB permite que você aplique uma expressão específica a cada elemento em uma matriz e retorne uma nova matriz com os elementos transformados. Esse operador é particularmente útil para manipular e transformar dados em matrizes, o que pode ajudar a simplificar o código do aplicativo e melhorar o desempenho das consultas, elevando o processamento da matriz ao nível do banco de dados.

Parâmetros

- `input`: A matriz a ser transformada.
- `as`: (opcional) O nome da variável a ser usada na expressão in para representar o elemento atual que está sendo processado.
- `in`: a expressão a ser aplicada a cada elemento na matriz de entrada.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o operador $map para transformar uma matriz de números, dobrando cada valor.

Crie documentos de amostra

```
db.collection.insertMany([
  { _id: 1, numbers: [1, 2, 3, 4, 5] },
  { _id: 2, numbers: [10, 20, 30, 40, 50] }
])
```

Exemplo de consulta

```
db.collection.aggregate([
  {
    $project: {
      doubledNumbers: { $map: { input: "$numbers", as: "num", in: { $multiply: ["$
$num", 2] } } }
    }
  }
])
```

Saída

```
[
  { _id: 1, doubledNumbers: [2, 4, 6, 8, 10] },
  { _id: 2, doubledNumbers: [20, 40, 60, 80, 100] }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$map` comando, escolha a guia do idioma que você deseja 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

O estágio de `$match` pipeline no Amazon DocumentDB é usado para filtrar os documentos de entrada somente para aqueles que correspondem aos critérios de consulta especificados. É um dos estágios de pipeline mais usados em operações de agregação. O `$match` estágio é aplicado antes de qualquer outro estágio do pipeline, permitindo que você reduza com eficiência o número de documentos que precisam ser processados nos estágios subsequentes.

Parâmetros

- `query`: um documento que expressa os critérios de seleção para a operação. O documento de consulta usa a mesma sintaxe do `find()` método.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$match` estágio para filtrar documentos com base em um valor de campo específico.

Crie documentos de amostra

```
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" }
]);
```

Exemplo de consulta

```
db.collection.aggregate([
  { $match: { age: { $gt: 30 } } },
  { $project: { _id: 1, name: 1, city: 1 } }
]);
```

Saída

```
[
  { "_id": 3, "name": "Bob", "city": "Chicago" },
  { "_id": 4, "name": "Alice", "city": "Miami" }
]
```

O `$match` estágio filtra os documentos para incluir somente aqueles em que o `age` campo é maior que 30.

Exemplos de código

Para ver um exemplo de código para usar o `$match` comando, escolha a guia do idioma que você deseja 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

O operador `$max` de agregação retorna o valor máximo de um campo especificado em um conjunto de documentos. Esse operador é útil para encontrar o valor mais alto em um conjunto de documentos.

Parâmetros

- `expression`: A expressão a ser usada para calcular o valor máximo.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$max` operador para encontrar a pontuação máxima em uma coleção de documentos estudantis. O `$group` estágio agrupa todos os documentos e o `$max` operador é usado para calcular o valor máximo do `score` campo em todos os documentos.

Crie documentos de amostra

```
db.students.insertMany([
  { name: "John", score: 85 },
  { name: "Jane", score: 92 },
  { name: "Bob", score: 78 },
  { name: "Alice", score: 90 }
])
```

Exemplo de consulta

```
db.students.aggregate([
  { $group: { _id: null, maxScore: { $max: "$score" } } },
  { $project: { _id: 0, maxScore: 1 } }
])
```

Saída

```
[ { maxScore: 92 } ]
```

Exemplo de uso do operador de janela (MongoDB Shell)

Novo na versão 8.0.2.

O `$max` operador também pode ser usado como operador de janela no `$setWindowFields` palco. Nesse contexto, ele retorna o valor máximo dos documentos em cada janela. Você especifica o operador no `output` campo e, opcionalmente, define os limites da janela com um `window` documento.

> **Note**
>
> Quando usado como operador de janela em`$setWindowFields`, `$max` é limitado a 100 MB de dados intermediários. Uma operação que excede esse limite retorna um erro.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

O exemplo a seguir particiona os documentos por`ticker`, classifica cada partição por `hour` e retorna o preço mais alto visto desde o início da partição até o documento atual.

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
      sortBy: { hour: 1 },
      output: {
        highestSoFar: {
          $max: "$price",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Saída

```
[
  { "_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 é aumentado com`highestSoFar`, o máximo de execução `price` dentro de sua partição até e incluindo o documento atual.

Exemplos de código

Para ver um exemplo de código para usar o `$max` operador, escolha a guia do idioma que você deseja usar. Os exemplos a seguir mostram o uso do acumulador (em`$group`) e o uso do operador de janela (em`$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()
```

$ máximo

Novo na versão 8.0.1.

O `$maxN` operador no Amazon DocumentDB retorna os N maiores valores. Quando usado como acumulador em um `$group` estágio, ele retorna uma matriz dos N valores máximos em cada grupo. Quando usado como um operador de expressão de matriz, ele retorna o máximo de N elementos de uma matriz.

Parâmetros

- `input`: uma expressão que resolve para a matriz ou campo do qual retornar os n valores máximos.
- n: uma expressão que resolve para um número inteiro positivo especificando quantos valores máximos devem ser retornados.

Comportamento

Comportamento do operador de expressão de matriz

- Você não pode especificar um valor n menor que`1`.
- `$maxN`filtra `null` os valores encontrados na `input` matriz.
- Se o especificado n for maior ou igual ao número de elementos na `input` matriz, `$maxN` retornará todos os elementos na `input` matriz.
- Se for `input` resolvido para um valor que não seja de matriz, a operação de agregação falhará.
- Se `input` contiver elementos numéricos e de string, os elementos de string serão classificados antes dos elementos numéricos de acordo com a ordem de comparação BSON.

Comportamento do acumulador

- Quando usado como acumulador, n deve ser uma expressão integral positiva que seja constante ou dependa do `_id` valor de`$group`.
- `$maxN`filtra valores nulos e ausentes.
- Se o grupo contiver menos de n elementos, `$maxN` retornará todos os elementos do grupo.
- `$maxN`compara os dados de entrada seguindo a ordem de comparação BSON para determinar o tipo de saída apropriado. Quando os dados de entrada contêm vários tipos de dados, o tipo de `$maxN` saída é o mais alto na ordem de comparação.

Ordenação de saída

`$maxN`retorna valores em nenhuma ordem de classificação específica. Se garantir uma ordem de classificação específica for um requisito, use `$topN` em vez disso ou envolva o resultado com`$sortArray`.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o $maxN acumulador para recuperar as duas pontuações mais altas de cada assunto.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.scores.aggregate([
  { $group: { _id: "$subject", highestTwo: { $maxN: { input: "$score", n: 2 } } } }
]);
```

Saída

```
[
  { "_id": "math", "highestTwo": [85, 93] },
  { "_id": "science", "highestTwo": [88, 90] }
]
```

Exemplo de uso de expressão (MongoDB Shell)

O $maxN operador também pode ser usado como uma expressão em um $project estágio para retornar os N maiores elementos de um campo de matriz.

Crie documentos de amostra

```
db.readings.insertMany([
  { _id: 1, sensor: "A", values: [45, 12, 78, 3, 56] },
  { _id: 2, sensor: "B", values: [90, 23, 67, 11, 44] }
]);
```

Exemplo de consulta

```
db.readings.aggregate([
  { $project: {
      highestThree: { $maxN: { input: "$values", n: 3 } }
    }}
]);
```

Saída

```
[
  { "_id": 1, "highestThree": [45, 56, 78] },
  { "_id": 2, "highestThree": [44, 90, 67] }
]
```

Exemplo de uso do operador de janela (MongoDB Shell)

Novo na versão 8.0.2.

O `$maxN` operador também pode ser usado como operador de janela no `$setWindowFields` palco. Nesse contexto, ele retorna uma matriz de até os n maiores valores para os documentos em cada janela. Você especifica o operador no `output` campo e, opcionalmente, define os limites da janela com um `window` documento.

> **Note**
>
> Quando usado como operador de janela em`$setWindowFields`, `$maxN` é limitado a 100 MB de dados intermediários. Uma operação que excede esse limite retorna um erro.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

O exemplo a seguir particiona os documentos por`ticker`, classifica cada partição por `hour` e retorna os dois preços mais altos vistos desde o início da partição até o documento atual.

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
      sortBy: { hour: 1 },
      output: {
        highestTwoSoFar: {
          $maxN: { input: "$price", n: 2 },
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Saída

```
[
  { "_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 é aumentado com `highestTwoSoFar` os dois maiores `price` valores em sua janela. Conforme observado na seção Ordem de saída, a ordem dos valores na matriz retornada não é garantida.

Exemplos de código

Para ver um exemplo de código para usar o `$maxN` operador, escolha a guia do idioma que você deseja usar. Os exemplos a seguir mostram o uso do acumulador (in`$group`), o uso da expressão (in`$project`) e o uso do operador de janela (in`$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

Novo na versão 8.0.1.

O `$median` operador no Amazon DocumentDB calcula o valor médio dos dados numéricos. Como acumulador, ele calcula a mediana dos valores numéricos em documentos dentro de um grupo no `$group` estágio de um pipeline de agregação. Como expressão, ele calcula a mediana de uma matriz de números.

Parâmetros

- `input`: uma expressão que resolve para um valor numérico ou uma matriz de valores numéricos.
- `method`: uma string especificando o método de cálculo. Atualmente, só `"approximate"` é suportado, que usa o algoritmo t-digest.

Comportamento

O `"approximate"` método usa o algoritmo t-digest para calcular uma mediana aproximada. O resultado é um valor existente no conjunto de dados, em vez de uma interpolação entre valores. A precisão melhora à medida que o número de pontos de dados aumenta.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$median` operador para calcular a pontuação média do teste por aula.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.students.aggregate([
  { $group: {
      _id: "$class",
      medianScore: { $median: { input: "$score", method: "approximate" } }
    }}
]);
```

Saída

```
[
  { "_id": "A", "medianScore": 85 },
  { "_id": "B", "medianScore": 80 }
]
```

Exemplo de uso de expressão (MongoDB Shell)

O `$median` operador também pode ser usado como uma expressão dentro de um `$project` estágio para calcular a mediana de um campo de matriz.

Crie documentos de amostra

```
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] }
]);
```

Exemplo de consulta

```
db.surveys.aggregate([
  { $project: {
      medianRating: { $median: { input: "$ratings", method: "approximate" } }
    }}
]);
```

Saída

```
[
  { "_id": 1, "medianRating": 5 },
  { "_id": 2, "medianRating": 30 },
  { "_id": 3, "medianRating": 2 }
]
```

Exemplo de uso do operador de janela (MongoDB Shell)

Novo na versão 8.0.2.

O `$median` operador também pode ser usado como operador de janela no `$setWindowFields` palco. Nesse contexto, ele calcula a mediana dos valores numéricos dos documentos em cada janela. Você especifica o operador no `output` campo e, opcionalmente, define os limites da janela com um `window` documento.

> **Note**
>
> Quando usado como operador de janela em`$setWindowFields`, `$median` é limitado a 100 MB de dados intermediários. Uma operação que excede esse limite retorna um erro.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

O exemplo a seguir particiona os documentos `city` e calcula a mediana `reading` em todos os documentos em cada partição.

```
db.temperatures.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$city",
      output: {
        medianReading: {
          $median: { input: "$reading", method: "approximate" },
          window: { documents: ["unbounded", "unbounded"] }
        }
      }
    }
  }
]);
```

Saída

```
[
  { "_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 é aumentado com `medianReading` a mediana aproximada de todos os documentos `reading` em sua partição.

Exemplos de código

Para ver um exemplo de código para usar o `$median` operador, escolha a guia do idioma que você deseja usar. Os exemplos a seguir mostram o uso do acumulador (in`$group`), o uso da expressão (in`$project`) e o uso do operador de janela (in`$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

O operador `$meta` de agregação acessa os metadados associados aos documentos em um pipeline de agregação. É comumente usado para recuperar pontuações de pesquisa de texto e classificar os resultados por relevância.

Parâmetros

- `textScore`: recupera a pontuação da pesquisa de texto indicando a relevância do documento para a consulta de pesquisa.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$meta` operador em um pipeline de agregação para recuperar e classificar por pontuações de pesquisa de texto.

Crie documentos de amostra

```
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." }
]);
```

Exemplo de consulta

```
db.articles.aggregate([
  { $match: { $text: { $search: "Python" } } },
  { $addFields: { score: { $meta: "textScore" } } },
  { $sort: { score: -1 } }
]);
```

Saída

```
[
```

```
  {
    _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
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o operador de `$meta` agregação, escolha a guia do idioma que você deseja 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

Introduzido na versão 8.0

O estágio `$merge` de agregação no Amazon DocumentDB é usado para mesclar os resultados do estágio anterior do pipeline em uma coleção de destino. Isso é útil para atualizar ou inserir documentos em uma coleção de destino com base nos dados dos documentos de entrada.

O `$merge` estágio permite que você execute várias ações com base na condição de correspondência entre os documentos de entrada e a coleção de destino, como:

```
- Insert new documents
- Update existing documents
- Delete documents
- Fail the operation if there are any conflicts
```

Parâmetros

- `into`: (obrigatório) O nome da coleção de destino na qual mesclar os documentos de entrada.

- on: (obrigatório) Os campos a serem usados como condição de correspondência entre os documentos de entrada e a coleção de destino.
- `whenMatched`: (opcional) A ação a ser executada quando o documento de entrada corresponde a um documento existente na coleção de destino. Os valores compatíveis são: `"merge"`, `"replace"`, `"keepExisting"` e `"fail"`. O Amazon DocumentDB não suporta a especificação de um pipeline de agregação como valor. `whenMatched`
- `whenNotMatched`: (opcional) A ação a ser executada quando o documento de entrada não corresponde a nenhum documento na coleção de destino. Os valores suportados são: `"insert"` `"fail"` e.
- `bypassDocumentValidation`: (opcional) Quando definido como`true`, os documentos gravados na coleção de destino pelo `$merge` estágio ignoram quaisquer regras de validação de esquema (`$jsonSchema`) definidas nessa coleção. O padrão é`false`, que impõe as regras de validação da coleção de destino. Isso é compatível com o Amazon DocumentDB 8.0.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$merge` estágio para atualizar uma `users` coleção com novos dados de um pipeline de entrada.

Crie documentos de amostra

```
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" }
]);
```

Exemplo de consulta

```
db.inputData.aggregate([
  {
    $merge: {
      into: "users",
      on: "_id",
```

```
      whenMatched: "merge",
      whenNotMatched: "insert"
    }
  }
])
```

Saída

Depois de executar o `$merge` pipeline, a `users` coleção conterá os seguintes 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" }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$merge` comando, escolha a guia do idioma que você deseja usar:

Node.js

Aqui está um exemplo do uso do operador $merge em um Node.js aplicativo:

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.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

Aqui está um exemplo do uso do operador $merge em um aplicativo Python:

```
from pymongo import MongoClient

def 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

O `$mergeObjects` operador no Amazon DocumentDB é usado para combinar vários documentos ou objetos em um único documento. Esse operador é particularmente útil quando você precisa mesclar o conteúdo de dois ou mais documentos ou objetos, potencialmente substituindo valores de um objeto pelos de outro.

Parâmetros

- `expression1`: o primeiro objeto a ser mesclado.
- `expression2`: (opcional) O segundo objeto a ser mesclado.
- `expression3`: (opcional) Objetos adicionais a serem mesclados.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$mergeObjects` operador para combinar dois objetos.

Crie documentos de amostra

```
db.collection.insertMany([
  { "_id": 1, "name": "John", "address": { "city": "New York", "state": "NY" } },
  { "_id": 2, "name": "Jane", "address": { "city": "Los Angeles", "state": "CA" } }
]);
```

Exemplo de consulta

```
db.collection.aggregate([
  {
    $project: {
      "combinedAddress": {
        $mergeObjects: ["$address", { "country": "USA" }]
      }
    }
  }
])
```

Saída

```
[
  {
    "_id": 1,
```

```
    "combinedAddress": {
      "city": "New York",
      "state": "NY",
      "country": "USA"
    }
  },
  {
    "_id": 2,
    "combinedAddress": {
      "city": "Los Angeles",
      "state": "CA",
      "country": "USA"
    }
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$mergeObjects` comando, escolha a guia do idioma que você deseja 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

O `$millisecond` operador no Amazon DocumentDB é usado para extrair a parte de milissegundos de um valor de data.

Parâmetros

Nenhum

Exemplo (MongoDB Shell)

Este exemplo demonstra como usar o `$millisecond` operador para extrair a parte de milissegundos de um valor de data.

Crie documentos de amostra

```
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")
  }
])
```

Exemplo de consulta

```
db.events.aggregate([
  {
    $project: {
      name: 1,
      milliseconds: { $millisecond: "$timestamp" }
    }
  }
])
```

Saída

```
[
  {
    "_id": ObjectId("644332a42054ed1b0d15f0c1"),
    "name": "Event 1",
    "milliseconds": 123
  },
  {
```

```
    "_id": ObjectId("644332a42054ed1b0d15f0c2"),
    "name": "Event 2",
    "milliseconds": 456
  },
  {
    "_id": ObjectId("644332a42054ed1b0d15f0c3"),
    "name": "Event 3",
    "milliseconds": 789
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$millisecond` comando, escolha a guia do idioma que você deseja 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

O operador `$min` de agregação retorna o valor mínimo de um campo especificado em um conjunto de documentos. Esse operador é útil para encontrar o menor valor em um conjunto de documentos.

Parâmetros

- `expression`: A expressão a ser avaliada. Isso pode ser um caminho de campo, uma variável ou qualquer expressão que se resolva em um valor.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$min` operador para encontrar o valor mínimo do `age` campo em vários documentos.

Crie documentos de amostra

```
db.users.insertMany([
```

```
  { name: "John", age: 35 },
  { name: "Jane", age: 28 },
  { name: "Bob", age: 42 },
  { name: "Alice", age: 31 }
]);
```

Exemplo de consulta

```
db.users.aggregate([
  { $group: { _id: null, minAge: { $min: "$age" } } },
  { $project: { _id: 0, minAge: 1 } }
])
```

Saída

```
[ { minAge: 28 } ]
```

Exemplo de uso do operador de janela (MongoDB Shell)

Novo na versão 8.0.2.

O `$min` operador também pode ser usado como operador de janela no `$setWindowFields` palco. Nesse contexto, ele retorna o valor mínimo para os documentos em cada janela. Você especifica o operador no `output` campo e, opcionalmente, define os limites da janela com um `window` documento.

> **Note**
>
> Quando usado como operador de janela em`$setWindowFields`, `$min` é limitado a 100 MB de dados intermediários. Uma operação que excede esse limite retorna um erro.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

O exemplo a seguir particiona os documentos por`ticker`, classifica cada partição por `hour` e retorna o menor preço visto desde o início da partição até o documento atual.

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
      sortBy: { hour: 1 },
      output: {
        lowestSoFar: {
          $min: "$price",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Saída

```
[
  { "_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 é aumentado com`lowestSoFar`, o mínimo de execução `price` dentro de sua partição, incluindo o documento atual.

Exemplos de código

Para ver um exemplo de código para usar o `$min` operador, escolha a guia do idioma que você deseja usar. Os exemplos a seguir mostram o uso do acumulador (em`$group`) e o uso do operador de janela (em`$setWindowFields`):

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');

  // Accumulator usage: minimum value across a group
  const users = db.collection('users');
  await users.insertMany([
    { name: "John", age: 35 },
    { name: "Jane", age: 28 },
    { name: "Bob", age: 42 },
    { name: "Alice", age: 31 }
  ]);
  const accumulatorResult = await users.aggregate([
    { $group: { _id: null, minAge: { $min: "$age" } } }
  ]).toArray();
  console.log('Accumulator result:', accumulatorResult);

  // Window operator usage: running minimum within each partition
  const stockPrices = db.collection('stockPrices');
  await stockPrices.insertMany([
    { _id: 1, ticker: "ABC", hour: 1, price: 50 },
    { _id: 2, ticker: "ABC", hour: 2, price: 45 },
    { _id: 3, ticker: "ABC", hour: 3, price: 60 },
    { _id: 4, ticker: "ABC", hour: 4, price: 40 },
    { _id: 5, ticker: "XYZ", hour: 1, price: 30 },
    { _id: 6, ticker: "XYZ", hour: 2, price: 35 },
    { _id: 7, ticker: "XYZ", hour: 3, price: 25 }
  ]);
  const windowResult = await stockPrices.aggregate([
    {
      $setWindowFields: {
```

```
        partitionBy: "$ticker",
        sortBy: { hour: 1 },
        output: {
          lowestSoFar: {
            $min: "$price",
            window: { documents: ["unbounded", "current"] }
          }
        }
      }
    }
  ]).toArray();
  console.log('Window result:', windowResult);

  client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']

    # Accumulator usage: minimum value across a group
    users = db.users
    users.insert_many([
        { "name": "John", "age": 35 },
        { "name": "Jane", "age": 28 },
        { "name": "Bob", "age": 42 },
        { "name": "Alice", "age": 31 }
    ])
    accumulator_result = list(users.aggregate([
        { "$group": { "_id": None, "minAge": { "$min": "$age" } } }
    ]))
    print('Accumulator result:', accumulator_result)

    # Window operator usage: running minimum within each partition
    stock_prices = db.stockPrices
```

```
    stock_prices.insert_many([
        { "_id": 1, "ticker": "ABC", "hour": 1, "price": 50 },
        { "_id": 2, "ticker": "ABC", "hour": 2, "price": 45 },
        { "_id": 3, "ticker": "ABC", "hour": 3, "price": 60 },
        { "_id": 4, "ticker": "ABC", "hour": 4, "price": 40 },
        { "_id": 5, "ticker": "XYZ", "hour": 1, "price": 30 },
        { "_id": 6, "ticker": "XYZ", "hour": 2, "price": 35 },
        { "_id": 7, "ticker": "XYZ", "hour": 3, "price": 25 }
    ])
    window_result = list(stock_prices.aggregate([
        {
            "$setWindowFields": {
                "partitionBy": "$ticker",
                "sortBy": { "hour": 1 },
                "output": {
                    "lowestSoFar": {
                        "$min": "$price",
                        "window": { "documents": ["unbounded", "current"] }
                    }
                }
            }
        }
    ]))
    print('Window result:', window_result)

    client.close()

example()
```

$ min

Novo na versão 8.0.1.

O `$minN` operador no Amazon DocumentDB retorna os N menores valores. Quando usado como acumulador em um `$group` estágio, ele retorna uma matriz dos N valores mínimos em cada grupo. Quando usado como um operador de expressão de matriz, ele retorna os N elementos mínimos de uma matriz.

Parâmetros

- `input`: uma expressão que resolve para a matriz ou campo do qual retornar os n valores mínimos.

- n: uma expressão que resolve para um número inteiro positivo especificando quantos valores mínimos devem ser retornados.

Comportamento

Comportamento do operador de expressão de matriz

- Você não pode especificar um valor n menor que`1`.
- `$minN`filtra `null` os valores encontrados na `input` matriz.
- Se o especificado n for maior ou igual ao número de elementos na `input` matriz, `$minN` retornará todos os elementos na `input` matriz.
- Se for `input` resolvido para um valor que não seja de matriz, a operação de agregação falhará.
- Se `input` contiver elementos numéricos e de string, os elementos numéricos serão classificados antes dos elementos de string de acordo com a ordem de comparação BSON.

Comportamento do acumulador

- Quando usado como acumulador, n deve ser uma expressão integral positiva que seja constante ou dependa do `_id` valor de`$group`.
- `$minN`filtra valores nulos e ausentes.
- Se o grupo contiver menos de n elementos, `$minN` retornará todos os elementos do grupo.
- `$minN`compara os dados de entrada seguindo a ordem de comparação BSON para determinar o tipo de saída apropriado. Quando os dados de entrada contêm vários tipos de dados, o tipo de `$minN` saída é o mais baixo na ordem de comparação.

Ordenação de saída

`$minN`retorna valores em nenhuma ordem de classificação específica. Se garantir uma ordem de classificação específica for um requisito, use `$bottomN` em vez disso ou envolva o resultado com`$sortArray`.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$minN` acumulador para recuperar as duas pontuações mais baixas de cada assunto.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.scores.aggregate([
  { $group: { _id: "$subject", lowestTwo: { $minN: { input: "$score", n: 2 } } } }
]);
```

Saída

```
[
  { "_id": "math", "lowestTwo": [72, 68] },
  { "_id": "science", "lowestTwo": [78, 65] }
]
```

Exemplo de uso de expressão (MongoDB Shell)

O `$minN` operador também pode ser usado como uma expressão em um `$project` estágio para retornar os N menores elementos de um campo de matriz.

Crie documentos de amostra

```
db.readings.insertMany([
  { _id: 1, sensor: "A", values: [45, 12, 78, 3, 56] },
  { _id: 2, sensor: "B", values: [90, 23, 67, 11, 44] }
]);
```

Exemplo de consulta

```
db.readings.aggregate([
  { $project: {
      lowestThree: { $minN: { input: "$values", n: 3 } }
```

```
    }}
]);
```

Saída

```
[
  { "_id": 1, "lowestThree": [45, 12, 3] },
  { "_id": 2, "lowestThree": [44, 23, 11] }
]
```

Exemplo de uso do operador de janela (MongoDB Shell)

Novo na versão 8.0.2.

O `$minN` operador também pode ser usado como operador de janela no `$setWindowFields` palco. Nesse contexto, ele retorna uma matriz de até os n menores valores para os documentos em cada janela. Você especifica o operador no `output` campo e, opcionalmente, define os limites da janela com um `window` documento.

> **Note**
>
> Quando usado como operador de janela em`$setWindowFields`, `$minN` é limitado a 100 MB de dados intermediários. Uma operação que excede esse limite retorna um erro.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

O exemplo a seguir particiona os documentos por`ticker`, classifica cada partição por `hour` e retorna os dois preços mais baixos vistos desde o início da partição até o documento atual.

```
db.stockPrices.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$ticker",
      sortBy: { hour: 1 },
      output: {
        lowestTwoSoFar: {
          $minN: { input: "$price", n: 2 },
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Saída

```
[
  { "_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 é aumentado com `lowestTwoSoFar` os dois menores `price` valores em sua janela. Conforme observado na seção Ordem de saída, a ordem dos valores na matriz retornada não é garantida.

Exemplos de código

Para ver um exemplo de código para usar o `$minN` operador, escolha a guia do idioma que você deseja usar. Os exemplos a seguir mostram o uso do acumulador (in`$group`), o uso da expressão (in`$project`) e o uso do operador de janela (in`$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

O estágio do pipeline `$minute` de agregação no Amazon DocumentDB extrai o valor do minuto de um campo de data ou timestamp.

Esse operador é útil quando você precisa realizar cálculos com base em data e hora ou agrupar em seu pipeline de agregação.

Parâmetros

- `expression`: o campo de data ou carimbo de data/hora do qual extrair o valor do minuto.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$minute` operador para agrupar os documentos pelo valor do minuto extraído do campo de carimbo de data/hora e contar o número de documentos em cada grupo.

Crie documentos de amostra

```
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") }
]);
```

Exemplo de consulta

```
db.events.aggregate([
  {
    $group: {
      _id: {
        minute: { $minute: "$timestamp" }
      },
      count: { $count: {} }
    }
  },
  { $sort: { "_id.minute": 1 } }
]);
```

Saída

```
[
  { "_id": { "minute": 30 }, "count": 2 },
  { "_id": { "minute": 31 }, "count": 2 },
  { "_id": { "minute": 32 }, "count": 1 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$minute` comando, escolha a guia do idioma que você deseja 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

O $mod operador é um operador aritmético que executa uma operação de módulo em um número. Ele retorna o restante da divisão de um número por outro. Esse operador é comumente usado para determinar se um número é par ou ímpar, ou para distribuir itens em um número finito de grupos.

Parâmetros

- `expression1`: A expressão do dividendo.
- `expression2`: A expressão divisora.

Exemplo (MongoDB Shell)

Este exemplo demonstra como usar o $mod operador para determinar o número de widgets restantes ao enviar em pacotes de 100.

Crie documentos de amostra

```
db.widgets.insertMany([
  { "_id" : 1, "widget" : "A", "count" : 80372 },
  { "_id" : 2, "widget" : "B", "count" : 409282 },
  { "_id" : 3, "widget" : "C", "count" : 60739 }
])
```

Exemplo de consulta

```
db.widgets.aggregate([
  { $addFields: { leftOver: { $mod: [ "$count", 100 ] } } }
])
```

Saída

```
[
  { "_id" : 1, "widget" : "A", "count" : 80372, "leftOver" : 72 },
  { "_id" : 2, "widget" : "B", "count" : 409282, "leftOver" : 82 },
  { "_id" : 3, "widget" : "C", "count" : 60739, "leftOver" : 39 }
]
```

A saída mostra o restante do `count` dividido por 100 para cada documento, o que representa o número de widgets restantes ao serem enviados em pacotes de 100.

Exemplos de código

Para ver um exemplo de código para usar o $mod comando, escolha a guia do idioma que você deseja 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

O `$month` operador no Amazon DocumentDB retorna o mês de uma data como um número entre 1 e 12. Esse operador é útil para extrair o componente mês de um campo de data e realizar agregações e análises baseadas em data.

Parâmetros

- `date_expression`: essa é a expressão ou o campo que contém a data ou o timestamp do qual você deseja extrair o mês.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$month` operador para extrair o mês de um campo de data e agrupar os dados por mês.

Crie documentos de amostra

```
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") }
]);
```

Exemplo de consulta

```
db.sales.aggregate([
  { $group: {
      _id: { month: { $month: "$date" } },
      totalSales: { $sum: "$price" }
    }},
  { $sort: { "_id.month": 1 } }
]);
```

Saída

```
[
  { _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 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$month` comando, escolha a guia do idioma que você deseja 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

O `$multiply` operador no Amazon DocumentDB é usado para multiplicar os valores de dois ou mais campos ou expressões. Esse operador é particularmente útil para realizar operações aritméticas em campos numéricos em documentos. Ele pode ser empregado em vários estágios do pipeline de agregação, como `$project` e`$addFields`, para criar novos campos ou modificar os existentes.

Parâmetros

- `expression1`: a primeira expressão numérica a ser multiplicada.
- `expression2`: a segunda expressão numérica a ser multiplicada.
- `[expression3, ...]`: (opcional) Expressões numéricas adicionais a serem multiplicadas.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso `$multiply` para calcular `bonus_miles` multiplicando `base_miles` e `bonus_rate` para clientes que usaram um cartão de crédito para a viagem.

Documentos de amostra

```
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 }
]);
```

Exemplo 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"]
      }
    }
  }
]);
```

Saída

```
[
  { _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 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$multiply` comando, escolha a guia do idioma que você deseja 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

O `$natural` operador no Amazon DocumentDB é usado para classificar os documentos em sua ordem natural, que é a ordem em que foram inseridos na coleção. Isso contrasta com o comportamento de classificação padrão, que consiste em classificar os documentos com base nos valores dos campos especificados.

Parâmetros

Nenhum

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$natural` operador para classificar os documentos em uma coleção em sua ordem natural.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.people.find({}, { "_id": 1, "name": 1 }).sort({ "$natural": 1 });
```

Saída

```
[
  { "_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" }
]
```

A consulta classifica os documentos na coleção em sua ordem natural, que é a ordem em que foram inseridos.

Exemplos de código

Para ver um exemplo de código para usar o `$natural` comando, escolha a guia do idioma que você deseja 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

O operador `$ne` de agregação compara dois valores e retorna `true` se eles não forem iguais, caso contrário, retorna. `false`

Parâmetros

- `expression1`: o primeiro valor a ser comparado.
- `expression2`: O segundo valor a ser comparado.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$ne` operador para identificar pedidos com alterações de status.

Crie documentos de amostra

```
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" }
]);
```

Exemplo de consulta

```
db.orders.aggregate([
  {
    $project: {
      orderId: 1,
      status: 1,
      expectedStatus: 1,
      needsAttention: { $ne: ["$status", "$expectedStatus"] }
    }
  }
]);
```

Saída

```
[
  { _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 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o operador de $ne agregação, escolha a guia do idioma que você deseja 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

O operador `$not` de agregação executa uma operação lógica NOT em uma expressão. Ele retorna `true` se a expressão for avaliada como e `false` se a `false` expressão for avaliada como. `true`

Parâmetros

- `expression`: A expressão a ser negada.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$not` operador para inverter valores booleanos.

Crie documentos de amostra

```
db.users.insertMany([
  { _id: 1, name: "Alice", active: true },
  { _id: 2, name: "Bob", active: false },
  { _id: 3, name: "Charlie", active: true }
]);
```

Exemplo de consulta

```
db.users.aggregate([
  {
    $project: {
      name: 1,
      active: 1,
      inactive: { $not: ["$active"] }
    }
  }
]);
```

Saída

```
[
  { _id: 1, name: 'Alice', active: true, inactive: false },
  { _id: 2, name: 'Bob', active: false, inactive: true },
  { _id: 3, name: 'Charlie', active: true, inactive: false }
]
```

Exemplos de código

Para ver um exemplo de código para usar o operador de `$not` agregação, escolha a guia do idioma que você deseja 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

O operador `$objectToArray` de agregação no Amazon DocumentDB converte um objeto (ou documento) em uma matriz. A entrada para o operador é um documento e a saída consiste em um elemento de matriz para cada par de valores de campo no documento de entrada. Esse operador é útil quando você precisa trabalhar com os campos individuais de um documento como uma matriz, como quando você deseja encontrar o documento com o valor máximo ou mínimo para um campo específico.

Parâmetros

- `expression`: a expressão do documento a ser convertida em uma matriz.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$objectToArray` operador para encontrar o documento com o estoque máximo de uma rede de locadoras de vídeo.

Crie documentos de amostra

```
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
    }
  }
]);
```

Exemplo de consulta

```
db.videos.aggregate([
  {
    $project: {
      name: 1,
      videos: {
        $objectToArray: "$inventory"
      }
    }
  },
  {
    $unwind: "$videos"
  },
  {
    $group: {
      _id: "$name",
      maxInventory: {
        $max: "$videos.v"
      }
    }
  }
]);
```

Saída

```
[
  {
    "_id": "Bravemind",
```

```
    "maxInventory": 1000
  },
  {
    "_id": "Live Soft",
    "maxInventory": 1000
  },
  {
    "_id": "Romancing the Rock",
    "maxInventory": 500
  },
  {
    "_id": "Top Pilot",
    "maxInventory": 1000
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$objectToArray` comando, escolha a guia do idioma que você deseja 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

O operador `$or` de agregação avalia várias expressões e retorna `true` se pelo menos uma expressão for avaliada como. `true` Ele retornará `false` somente se todas as expressões forem`false`.

Parâmetros

- `expressions`: uma matriz de expressões para avaliar.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$or` operador para verificar se os produtos atendem a algum dos vários critérios.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.items.aggregate([
  {
    $project: {
```

```
      name: 1,
      price: 1,
      onSale: 1,
      goodDeal: {
        $or: [
          { $lt: ["$price", 50] },
          { $eq: ["$onSale", true] }
        ]
      }
    }
  }
]);
```

Saída

```
[
  { _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 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o operador de `$or` agregação, escolha a guia do idioma que você deseja 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

O `$out` operador no Amazon DocumentDB é usado para gravar o resultado de um pipeline de agregação em uma coleção especificada.

`$out`deve ser a última etapa do pipeline.

Parâmetros

- `output_collection`: o nome da coleção de saída na qual gravar os resultados da agregação.

Nota: Se a coleção já existir, ela será substituída pelos resultados do estágio de agregação.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$out` operador no Amazon DocumentDB para gravar os resultados de um pipeline de agregação em uma nova coleção.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.products.aggregate([
  { $group: { _id: "$category", totalPrice: { $sum: "$price" } } },
  { $out: "product_categories" }
])
```

Saída

Nenhum (os resultados são gravados na coleção de saída).

O pipeline de agregação agrupa os produtos por categoria e calcula o preço total dos itens para cada categoria. O `$out` operador grava os resultados em uma nova coleção chamada “product_categories”.

Para visualizar os resultados na coleção de saída:

```
db.product_categories.find()
[
{ "_id" : "Books", "totalPrice" : 125 },
{ "_id" : "Electronics", "totalPrice" : 300 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$out` comando, escolha a guia do idioma que você deseja 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

Novo na versão 8.0.1.

O `$percentile` operador no Amazon DocumentDB calcula valores percentuais especificados para dados numéricos. Como acumulador, ele calcula valores percentuais em documentos dentro de um grupo no `$group` estágio de um pipeline de agregação. Como expressão, ele calcula os percentis para uma matriz de números.

Parâmetros

- `input`: uma expressão que se resolve em um valor numérico ou em uma matriz de valores numéricos.
- p: uma matriz de valores percentuais entre 0 e 1, em que cada valor representa um percentil a ser calculado. Por exemplo, `[0.25, 0.5, 0.75]` calcula os percentis 25, 50 e 75.
- `method`: uma string especificando o método de cálculo. No momento, somente `"approximate"` é compatível.

Comportamento

O `"approximate"` método usa o algoritmo t-digest para calcular métricas aproximadas baseadas em percentis. Com pequenos conjuntos de dados, valores percentuais distintos podem resultar no mesmo resultado. Por exemplo, com apenas 5 valores por grupo, p90 e p99 podem retornar o valor máximo no grupo. A precisão melhora à medida que o número de pontos de dados aumenta.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$percentile` operador para calcular o 25º e o 75º percentil de pontuações por classe.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.students.aggregate([
  { $group: {
      _id: "$class",
      percentiles: { $percentile: { input: "$score", p: [0.25, 0.75], method:
 "approximate" } }
```

```
    }}
]);
```

Saída

```
[
  { "_id": "A", "percentiles": [72, 90] },
  { "_id": "B", "percentiles": [75, 88] }
]
```

Exemplo de uso de expressão (MongoDB Shell)

O `$percentile` operador também pode ser usado como uma expressão dentro de um `$project` estágio para calcular os percentis de um campo de matriz.

Crie documentos de amostra

```
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] }
]);
```

Exemplo de consulta

```
db.surveys.aggregate([
  { $project: {
      quartiles: { $percentile: { input: "$responses", p: [0.25, 0.5, 0.75], method:
 "approximate" } }
    }}
]);
```

Saída

```
[
  { "_id": 1, "quartiles": [6, 10, 16] },
  { "_id": 2, "quartiles": [5, 9, 15] }
]
```

Exemplo de uso do operador de janela (MongoDB Shell)

Novo na versão 8.0.2.

O `$percentile` operador também pode ser usado como operador de janela no `$setWindowFields` palco. Nesse contexto, ele calcula os valores percentuais especificados para os documentos em cada janela. Você especifica o operador no `output` campo e, opcionalmente, define os limites da janela com um `window` documento.

> **Note**
>
> Quando usado como operador de janela em`$setWindowFields`, `$percentile` é limitado a 100 MB de dados intermediários. Uma operação que excede esse limite retorna um erro.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

O exemplo a seguir particiona os documentos `sensor` e calcula os percentis 25 e 75 de todos os documentos `reading` em cada partição.

```
db.sensorData.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$sensor",
      output: {
        readingPercentiles: {
          $percentile: { input: "$reading", p: [0.25, 0.75], method: "approximate" },
          window: { documents: ["unbounded", "unbounded"] }
        }
      }
    }
```

```
  }
]);
```

Saída

```
[
  { "_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 é aumentado com`readingPercentiles`, aproximadamente, os percentis 25 e 75 de todos os documentos `reading` em sua partição.

Exemplos de código

Para ver um exemplo de código para usar o `$percentile` operador, escolha a guia do idioma que você deseja usar. Os exemplos a seguir mostram o uso do acumulador (in`$group`), o uso da expressão (in`$project`) e o uso do operador de janela (in`$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

Introduzido na versão 8.0

O $pow operador no Amazon DocumentDB permite que você eleve um número a uma potência. Isso pode ser útil para realizar cálculos exponenciais em seu pipeline de agregação.

Parâmetros

- <number>(obrigatório): O número a ser elevado a uma potência.
- <exponent>(obrigatório): A potência à qual o número deve ser aumentado.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o $pow operador para calcular o quadrado de um número.

Crie documentos de amostra

```
db.numbers.insertMany([
  { "_id": 1, "value": 2 },
  { "_id": 2, "value": 3 },
  { "_id": 3, "value": 4 }
]);
```

Exemplo de consulta

```
db.numbers.aggregate([
  { $addFields: { "square": { $pow: ["$value", 2] } } }
])
```

Saída

```
[
  { "_id": 1, "value": 2, "square": 4 },
  { "_id": 2, "value": 3, "square": 9 },
  { "_id": 3, "value": 4, "square": 16 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o $pow comando, escolha a guia do idioma que você deseja usar:

Node.js

Aqui está um exemplo de uso do operador $pow em um Node.js aplicativo:

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Aqui está um exemplo do uso do operador $pow em um aplicativo Python:

```
from pymongo import MongoClient

def 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

O operador `$push` de agregação retorna uma matriz de todos os valores de uma expressão especificada para cada grupo. Normalmente é usado dentro do `$group` estágio para acumular valores em uma matriz.

Parâmetros

- `expression`: A expressão a ser avaliada para cada documento no grupo.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$push` operador para coletar todos os nomes de produtos de cada categoria.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.sales.aggregate([
  {
    $group: {
      _id: "$category",
      products: { $push: "$product" }
    }
  }
]);
```

Saída

```
[
  { _id: 'Furniture', products: [ 'Desk', 'Chair' ] },
  { _id: 'Electronics', products: [ 'Laptop', 'Mouse', 'Keyboard' ] }
]
```

Exemplo de uso do operador de janela (MongoDB Shell)

Novo na versão 8.0.2.

O `$push` operador também pode ser usado como operador de janela no `$setWindowFields` palco. Nesse contexto, ele retorna uma matriz de todos os valores da expressão especificada para os documentos em cada janela, em ordem de classificação. Você especifica o operador no `output` campo e, opcionalmente, define os limites da janela com um `window` documento.

> **Note**
>
> Quando usado como operador de janela em`$setWindowFields`, `$push` é limitado a 100 MB de dados intermediários. Uma operação que excede esse limite retorna um erro.

Crie documentos de amostra

```
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" }
]);
```

Exemplo de consulta

O exemplo a seguir particiona os documentos por`category`, classifica cada partição por `day` e retorna uma lista contínua de `product` valores desde o início da partição até o documento atual.

```
db.purchases.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$category",
      sortBy: { day: 1 },
      output: {
        productsSoFar: {
          $push: "$product",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Saída

```
[
  { "_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 é aumentado com a matriz de todos os `product` valores em sua partição`productsSoFar`, desde o início até o documento atual, em ordem de classificação.

Exemplos de código

Para ver um exemplo de código para usar o `$push` operador, escolha a guia do idioma que você deseja usar. Os exemplos a seguir mostram o uso do acumulador (em`$group`) e o uso do operador de janela (em`$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

O `$project` operador no Amazon DocumentDB permite que você inclua ou exclua seletivamente campos dos documentos de saída, passe valores para a próxima etapa do pipeline e calcule novos campos a partir dos valores do documento de entrada.

Parâmetros

- `field`: o campo a ser incluído ou excluído dos documentos de saída, pode ser um caminho de campo (por exemplo, “a.b.c”).
- `1`ou`true`: Inclui o campo na saída.
- `0`ou`false`: Exclui o campo da saída.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$project` operador na coleção de estudantes

Crie documentos de amostra

```
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" }
  ]);
```

Essa consulta inclui somente os `math` campos `name` e na saída. O `_id` campo é incluído por padrão, a menos que seja explicitamente excluído.

```
db.students.aggregate([
  { $project: { "name": 1, "math": 1 } }
])
```

Saída

```
{ _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 }
```

Essa consulta exclui os `_id` campos `grade` e da saída, mostrando todos os outros campos (`name`,`math`,`science`).

```
db.students.aggregate([
  { $project: { "grade": 0, "_id": 0 } }
])
```

Saída

```
{ 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 }
```

Exemplos de código

Para ver um exemplo de código para usar o `$project` comando, escolha a guia do idioma que você deseja 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()
```

$ radianos ToDegrees

Novo na versão 8.0.1.

O `$radiansToDegrees` operador no Amazon DocumentDB converte um valor de radianos em graus.

Parâmetros

- `expression`: uma expressão que resolve para um valor numérico em radianos.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$radiansToDegrees` operador para converter valores de radianos em graus.

Crie documentos de amostra

```
db.angles.insertMany([
  { "_id": 1, "radians": 0 },
  { "_id": 2, "radians": 1.5707963267948966 },
  { "_id": 3, "radians": 3.141592653589793 },
  { "_id": 4, "radians": 6.283185307179586 }
]);
```

Exemplo de consulta

```
db.angles.aggregate([
  { $project: {
    "degrees": { $radiansToDegrees: "$radians" }
  }}
]);
```

Saída

```
[
  { "_id": 1, "degrees": 0 },
  { "_id": 2, "degrees": 90 },
  { "_id": 3, "degrees": 180 },
  { "_id": 4, "degrees": 360 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$radiansToDegrees` operador, escolha a guia do idioma que você deseja 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

Novo a partir da versão 8.0

O `$rand` operador no Amazon DocumentDB é usado para gerar um número aleatório entre 0 e 1.

Parâmetros

Nenhum

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$rand` operador para selecionar aleatoriamente dois documentos da `temp` coleção.

Crie documentos de amostra

```
db.items.insertMany([
  { "name": "pencil", "quantity": 110 },
  { "name": "pen", "quantity": 159 }
])
```

Exemplo de consulta

```
db.items.aggregate([
  {
    $project: {
      randomValue: { $rand: {} }
    }
  }
])
```

Saída

```
[
  {
    _id: ObjectId('6924a5edd66dcae121d29517'),
    randomValue: 0.8615243955294392
  },
  {
    _id: ObjectId('6924a5edd66dcae121d29518'),
    randomValue: 0.22815483022099903
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$rand` comando, escolha a guia do idioma que você deseja 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

O operador `$range` de agregação no Amazon DocumentDB é usado para criar uma matriz de números consecutivos dentro de um intervalo especificado. Esse operador é particularmente útil para gerar sequências de números, como marcadores de milhas para postos de socorro em uma corrida, conforme demonstrado nos exemplos a seguir.

Parâmetros

- `start`: O valor inicial do intervalo.
- `end`: O valor final do intervalo.
- `step`: (opcional) O valor da etapa a ser usado ao gerar o intervalo. Se não for fornecido, o valor padrão da etapa será 1.

Exemplo (MongoDB Shell)

Neste exemplo, usaremos o `$range` operador para gerar os marcadores de milhas para estações de água em uma corrida de bicicleta.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.races.aggregate([
```

```
  {
    $project: {
      race: 1,
      "waterStations": { $range: [20, "$distance", 20] }
    }
  }
]);
```

Saída

```
[
  {
    _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 ] }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$range` comando, escolha a guia do idioma que você deseja 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

Novo na versão 8.0.2.

O operador de `$rank` janela no Amazon DocumentDB retorna a posição de cada documento em sua partição, de acordo com a `sortBy` ordem das `$setWindowFields` etapas. Documentos com o mesmo `sortBy` valor recebem a mesma classificação, e o próximo valor distinto ignora as posições intermediárias, deixando lacunas (por exemplo,`1, 2, 2, 4`). É especificado no `output` campo do `$setWindowFields` estágio.

É um operador de janela usado somente no `$setWindowFields` palco.

Parâmetros

- `$rank`usa um documento vazio `{}` como valor.
- Ele não aceita uma `window` moldura.
- Isso requer exatamente uma `sortBy` chave no `$setWindowFields` estágio.

Exemplo (MongoDB Shell)

O exemplo a seguir divide os documentos porgame, classifica cada partição `score` em ordem decrescente e atribui uma classificação a cada jogador.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.players.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$game",
      sortBy: { score: -1 },
      output: {
        rank: { $rank: {} }
      }
    }
  }
]);
```

Saída

```
[
  { "_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` partição, os jogadores são classificados por ordem decrescente`score`. Bob e Carol empatam `85` e dividem a classificação`2`, então a próxima classificação passa `4` para Dave.

Exemplos de código

Para ver um exemplo de código para usar o `$rank` operador no `$setWindowFields` estágio, escolha a guia do idioma que você deseja 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 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

O estágio `$redact` de agregação no Amazon DocumentDB é usado para incluir ou excluir seletivamente conteúdo dos documentos de saída com base nos valores dos campos especificados. Isso é particularmente útil em cenários em que você precisa controlar a visibilidade de dados confidenciais com base nos níveis de acesso ou nas permissões do usuário.

Parâmetros

- `$cond`: uma expressão que é avaliada como `$$KEEP`$$PRUNE, ou `$$DESCEND` para cada campo no documento.
- `$$KEEP`: mantém o campo atual no documento de saída.
- `$$PRUNE`: remove o campo atual do documento de saída.
- `$$DESCEND`: aplica recursivamente o `$redact` estágio ao campo atual, que é um objeto ou matriz.

Exemplo (MongoDB Shell)

Neste exemplo, usaremos o `$redact` estágio para filtrar pedidos com base em seu status, mostrando somente pedidos com valores de status específicos.

Crie documentos de amostra

```
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" }
])
```

Exemplo de consulta

```
db.orders.aggregate([
```

```
  {
    $redact: {
      $cond: {
        if: { $in: ["$status", ["shipped", "processing"]] },
        then: "$$KEEP",
        else: "$$PRUNE"
      }
    }
  }
])
```

Saída

```
[
  { _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' }
]
```

Neste exemplo, o `$redact` estágio verifica o valor do `status` campo em cada documento. Se `status` for “enviado” ou “processado”, o documento será mantido (`$$KEEP`). Caso contrário, o documento será removido (`$$PRUNE`).

Exemplos de código

Para ver um exemplo de código para usar o `$redact` comando, escolha a guia do idioma que você deseja 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

O operador `$reduce` de agregação no Amazon DocumentDB é usado para aplicar uma função de dois argumentos cumulativamente aos elementos de uma matriz para reduzir a matriz a um único valor. Esse operador é particularmente útil para realizar cálculos ou transformações complexas em dados de matriz dentro do pipeline de agregação.

Parâmetros

- `input`: A matriz a ser reduzida.
- `initialValue`: O valor inicial a ser usado na operação de redução.
- `in`: a expressão a ser avaliada em cada elemento da `input` matriz. Essa expressão deve retornar um valor que será usado na próxima iteração da redução.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$reduce` operador para calcular a soma de todos os elementos em uma matriz.

Crie documentos de amostra

```
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] }
])
```

Exemplo de consulta

```
db.orders.aggregate([
  {
    $project: {
      total: {
        $reduce: {
          input: "$items",
          initialValue: 0,
          in: { $add: ["$$value", "$$this"] }
        }
      }
```

```
    }
  }
])
```

Saída

```
[
  { "_id": 1, "total": 15 },
  { "_id": 2, "total": 60 },
  { "_id": 3, "total": 80 },
  { "_id": 4, "total": 300 }
]
```

O `$reduce` operador itera sobre a `items` matriz, adicionando cada elemento ao `initialValue` de 0. O resultado é a soma de todos os elementos na matriz.

Exemplos de código

Para ver um exemplo de código para usar o `$reduce` comando, escolha a guia do idioma que você deseja usar:

Node.js

Aqui está um exemplo de uso do `$reduce` operador em um Node.js aplicativo:

```
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

Aqui está um exemplo de uso do `$reduce` operador em um aplicativo 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

Novo a partir da versão 5.0.

Não é compatível com o cluster elástico.

O `$regexFind` operador no Amazon DocumentDB é usado para realizar a correspondência de expressões regulares em campos de string em documentos. Ele permite pesquisar e extrair substrings específicas que correspondam a um determinado padrão de expressão regular.

Parâmetros

- `input`: O campo de string ou expressão a ser pesquisado.
- `regex`: O padrão de expressão regular a ser correspondente.
- `options`: (opcional) Um objeto que especifica parâmetros opcionais para a expressão regular, como distinção entre maiúsculas e minúsculas e correspondência de várias linhas. As opções suportadas são `i` (sem distinção entre maiúsculas e minúsculas) e `m` (várias linhas).

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$regexFind` operador para pesquisar documentos em que o `name` campo corresponda a um padrão específico de expressão regular.

Crie documentos de amostra

```
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" }
]);
```

Exemplo de consulta

```
db.users.aggregate([
  {
    $project: {
      names: {
        $regexFind: { input: '$name', regex: 'j', options: 'i' }
```

```
      }
    }
  },
  { $match: {names: {$ne: null}}}
])
```

Essa consulta retornará todos os documentos em que o `name` campo contém a letra “j” (sem distinção entre maiúsculas e minúsculas).

Saída

```
[
  { _id: 1, names: { match: 'J', idx: 0, captures: [] } }
]
```

Nota: Se sua consulta estiver usando o planejador Amazon DocumentDB versão 1, você deve usar uma dica para utilizar um índice. Sem uma dica, a consulta pode realizar uma varredura de coleção. Para verificar sua versão do planejador e saber mais sobre como usar dicas, consulte. Planejador de consultas v2

Exemplos de código

Para ver um exemplo de código para usar o `$regexFind` comando, escolha a guia do idioma que você deseja 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()
```

$ regex FindAll

Introduzido na versão 5.0

O `$regexFindAll` operador no Amazon DocumentDB é usado para realizar a correspondência de expressões regulares em campos de string em documentos. Ele permite pesquisar e extrair substrings específicas que correspondam a um determinado padrão de expressão regular, retornando todas as correspondências da expressão regular.

Parâmetros

- `input`: O campo de string ou expressão a ser pesquisado.

- `regex`: O padrão de expressão regular a ser correspondente.
- `options`: (opcional) Um objeto que especifica parâmetros opcionais para a expressão regular, como distinção entre maiúsculas e minúsculas e correspondência de várias linhas. As opções suportadas são `i` (sem distinção entre maiúsculas e minúsculas) e `m` (várias linhas).

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$regexFindAll` operador para extrair todas as sequências de letras do `email` campo.

Crie documentos de amostra

```
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" }

]);
```

Exemplo de consulta

```
db.users.aggregate([
  {
    $project: {
      name: 1,
      emailMatches: {
        $regexFindAll: { input: '$email', regex: '[a-z]+', options: 'i' }
      }
    }
  }
])
```

Saída

```
[
  {
    _id: 1,
    name: 'John Doe',
    emailMatches: [
```

```
      { match: 'john', idx: 0, captures: [] },
      { match: 'example', idx: 5, captures: [] },
      { match: 'com', idx: 13, captures: [] }
    ]
  },
  {
    _id: 2,
    name: 'Jane Roe',
    emailMatches: [
      { match: 'jane', idx: 0, captures: [] },
      { match: 'example', idx: 5, captures: [] },
      { match: 'com', idx: 13, captures: [] }
    ]
  },
  {
    _id: 3,
    name: 'Carlos Salazar',
    emailMatches: [
      { match: 'carlos', idx: 0, captures: [] },
      { match: 'example', idx: 7, captures: [] },
      { match: 'com', idx: 15, captures: [] }
    ]
  },
  {
    _id: 4,
    name: 'Saanvi Sarkar',
    emailMatches: [
      { match: 'saanvi', idx: 0, captures: [] },
      { match: 'example', idx: 7, captures: [] },
      { match: 'com', idx: 15, captures: [] }
    ]
  }
]
```

Nota: Se sua consulta estiver usando o planejador Amazon DocumentDB versão 1, você deve usar uma dica para utilizar um índice. Sem uma dica, a consulta pode realizar uma varredura de coleção. Para verificar sua versão do planejador e saber mais sobre como usar dicas, consulte. Planejador de consultas v2

Exemplos de código

Para ver um exemplo de código para usar o `$regexFindAll` comando, escolha a guia do idioma que você deseja usar:

Node.js

Aqui está um exemplo de uso do `$regexFind` operador em um Node.js aplicativo:

```
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

Aqui está um exemplo de uso do `$regexFind` operador em um aplicativo 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

Novo a partir da versão 5.0. Não é compatível com o cluster elástico.

O `$regexMatch` operador no Amazon DocumentDB é usado para realizar a correspondência de expressões regulares em campos de string. Ele retorna um valor booleano (`trueoufalse`) indicando se a string de entrada corresponde ao padrão especificado.

Parâmetros

- `input`: a string a ser testada em relação à expressão regular.
- `regex`: O padrão de expressão regular a ser correspondente.
- `options`: (Opcional) Sinalizadores para modificar o comportamento da expressão regular, como correspondência sem distinção entre maiúsculas e minúsculas (`i`) ou correspondência de várias linhas (). `m`

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$regexMatch` operador para verificar se os nomes começam com a letra 'M'. O operador retorna `true` ou `false` para cada documento.

Crie documentos de amostra

```
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" },

]);
```

Exemplo de consulta

```
db.users.aggregate([
  {
    $project: {
      name: 1,
      startsWithM: {
        $regexMatch: {
          input: "$name",
          regex: "^M",
          options: "i"
        }
      }
    }
  }
]);
```

Saída

```
{ _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 }
```

Exemplos de código

Para ver um exemplo de código para usar o `$regexMatch` comando, escolha a guia do idioma que você deseja 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

Introduzido na versão 5.0

O `$replaceAll` operador no Amazon DocumentDB é usado para substituir todas as ocorrências de um padrão de sequência de caracteres especificado em um campo por uma nova cadeia de caracteres. Esse operador pode ser útil para tarefas como normalização de dados, limpeza de texto e manipulação de cadeias de caracteres.

Parâmetros

- `input`: o campo ou a expressão que contém a string a ser substituída.
- `find`: O padrão de string a ser pesquisado e substituído.
- `replacement`: a string pela qual substituir as ocorrências correspondentes.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$replaceAll` operador em um pipeline de agregação para substituir todas as ocorrências da string “Chocolatier” por “Chocolate Co.” no campo “BrandName” de uma coleção de “produtos”.

Crie documentos de amostra

```
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
  }
]);
```

Exemplo de consulta

```
db.products.aggregate([
  {
    $addFields: {
      "brandName": {
        $replaceAll: {
          input: "$brandName",
          find: "Chocolatier",
          replacement: "Chocolate Co."
        }
      }
    }
  }
```

```
])
```

Saída

```
[
  {
    _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
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$replaceAll` comando, escolha a guia do idioma que você deseja 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

Introduzido na versão 5.0

O `$replaceOne` operador no Amazon DocumentDB é um operador de expressão de string usado em pipelines de agregação para substituir a primeira ocorrência de uma substring especificada em uma string por uma string de substituição. Esse operador diferencia maiúsculas de minúsculas e substitui somente a primeira correspondência encontrada.

Parâmetros

- `input`: A string (campo) na qual realizar a busca.
- `find`: a string a ser pesquisada na entrada.
- `replacement`: a string para substituir a primeira ocorrência da descoberta na entrada (campo).

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$replaceOne` operador em um pipeline de agregação para substituir substrings em nomes de produtos.

Crie documentos de amostra

```
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 }
```

```
]);
```

Exemplo de agregação

```
db.products.aggregate([
  {
    $addFields: {
      standardizedName: {
        $replaceOne: {
          input: "$name",
          find: "Pack",
          replacement: "Package"
        }
      }
    }
  }
]);
```

Saída

A saída mostra que somente a primeira ocorrência de “Pack” em cada nome de produto foi substituída 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"
}
```

Exemplos de código

Para ver um exemplo de código para usar o `$replaceOne` comando, escolha a guia do idioma que você deseja 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

O `$replaceRoot` operador é usado para substituir o documento raiz pelo documento incorporado especificado. Isso é útil quando você deseja promover um documento aninhado ao nível superior ou reestruturar sua saída de dados.

Parâmetros

- `newRoot`: o novo documento raiz que substituirá o documento raiz existente.

Exemplo (MongoDB Shell)

Este exemplo mostra como extrair informações de endereço de entrega de pedidos de clientes, o que é útil para gerar etiquetas de frete ou listas de endereços.

Crie documentos de amostra

```
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 } ])
```

Exemplo de consulta

```
db.orders.aggregate([
  {
    $replaceRoot: {
      newRoot: "$shippingAddress"
    }
  }
])
```

Saída

```
{
    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'
  }
```

Exemplos de código

Para ver um exemplo de código para usar o `$replaceRoot` comando, escolha a guia do idioma que você deseja 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()
```

$ substituir por

Novo a partir da versão 8.0

Não é compatível com o cluster elástico.

O estágio `$replaceWith` de agregação no Amazon DocumentDB é usado para substituir o documento de entrada por um novo documento. Todos os campos existentes no documento de entrada, incluindo o campo _id, são substituídos pelo novo documento. `$replaceWith`é comumente usado para nivelar documentos ou promover um documento incorporado ao nível superior.

Parâmetros

- `<replacement>`(obrigatório): o novo documento que substituirá o documento existente.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$replaceWith` operador para substituir um documento existente em uma coleção do Amazon DocumentDB.

Crie documentos de amostra

```
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 ]
  }
]);
```

Exemplo de consulta

```
db.restaurants.aggregate([
  { $replaceWith: {
      name: "$name",
      cuisine: "$cuisine",
      rating: { $avg: "$ratings" }
    }
  }
]);
```

Saída

```
[
  {
    name: 'Biryani Adda',
    cuisine: 'Indian',
    rating: 3.4
  },
  {
    name: 'The Burger Spot',
    cuisine: 'American',
    rating: 2.7777777777777777
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$replaceWith` comando, escolha a guia do idioma que você deseja 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

O `$reverseArray` operador no Amazon DocumentDB é usado para reverter os elementos de uma matriz na ordem especificada. Esse operador é útil quando você precisa reordenar os elementos de uma matriz na direção inversa.

Parâmetros

- `expression`: A expressão de matriz a ser revertida.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$reverseArray` operador para reverter a ordem dos elementos em uma matriz.

Crie documentos de amostra

```
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]}
]);
```

Exemplo de consulta

```
db.miles.aggregate([
  {
    $project: {
      _id: 1,
      member_since: 1,
      credit_card: 1,
      reversed_flight_miles: { $reverseArray: "$flight_miles" }
    }
  }
]);
```

Saída

```
{ "_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 ] }
```

Neste exemplo, o `$reverseArray` operador é usado para reverter a ordem da `flight_miles` matriz. O `reversed_flight_miles` campo resultante na saída mostra os elementos da matriz na ordem inversa.

Exemplos de código

Para ver um exemplo de código para usar o `$reverseArray` comando, escolha a guia do idioma que você deseja usar:

Node.js

Aqui está um exemplo de uso do `$reverseArray` operador em um Node.js aplicativo:

```
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

Aqui está um exemplo de uso do `$reverseArray` operador em um aplicativo 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

Novo na versão 8.0.1.

O `$round` operador no Amazon DocumentDB arredonda um número para uma casa decimal especificada. Quando um valor fica exatamente na metade do caminho entre os dois valores mais próximos naquele local, `$round` arredonda para o dígito par mais próximo. Por exemplo, 2,5 rodadas para 2 e 3,5 rodadas para 4.

Parâmetros

- `number`: uma expressão que se resolve em um número.
- `place`: opcional. Uma expressão inteira entre -20 e 100 que especifica o número de casas decimais para as quais arredondar. Um valor negativo é arredondado para a esquerda do decimal. O padrão é 0.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$round` operador para arredondar valores numéricos para uma casa decimal.

Arredondamento com um valor posicional positivo

```
db.measurements.insertMany([
  {_id: 1, value: 3.456},
  {_id: 2, value: 7.891},
  {_id: 3, value: 12.345}
]);
```

Exemplo de consulta

```
db.measurements.aggregate([
  { $project: { rounded: { $round: ["$value", 1] } } }
]);
```

Saída

```
[
  {_id: 1, rounded: 3.5},
  {_id: 2, rounded: 7.9},
  {_id: 3, rounded: 12.3}
]
```

Arredondando valores intermediários para o dígito par mais próximo

Quando um valor fica exatamente na metade do caminho entre os dois valores mais próximos no local de destino, `$round` arredonda para o dígito par mais próximo, em vez de sempre arredondar para cima. O exemplo a seguir arredonda valores intermediários para números inteiros.

```
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] } } }
]);
```

Saída

```
[
  {_id: 1, rounded: 10},
  {_id: 2, rounded: 12},
  {_id: 3, rounded: 12},
  {_id: 4, rounded: 14}
]
```

Porque `10.5` está exatamente na metade do caminho entre `10` e`11`, ele arredonda para o valor par mais próximo. `10` Da mesma forma, `11.5` e `12.5` ambos arredondam para o valor `12` par mais próximo e `13.5` arredondam para`14`.

Arredondamento com um valor posicional negativo

Quando `place` for negativo, `$round` arredonda para a esquerda do decimal. O exemplo a seguir arredonda valores usando o primeiro dígito à esquerda do 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] } } }
]);
```

Saída

```
[
  {_id: 1, rounded: 20},
  {_id: 2, rounded: 30},
  {_id: 3, rounded: 30}
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$round` operador, escolha a guia do idioma que você deseja 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('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

Novo a partir da versão 4.0.

Não é compatível com o cluster elástico.

O `$rtrim` operador no Amazon DocumentDB é usado para remover caracteres finais de uma string. Por padrão, ele remove os caracteres de espaço em branco à direita, mas você também pode especificar um conjunto de caracteres a serem removidos passando o argumento chars.

Parâmetros

- `input`: a string de entrada da qual remover os caracteres de espaço em branco à direita.
- `chars`: (opcional) Para remover caracteres específicos.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$rtrim` operador para remover caracteres especificados à direita (” *”) de uma string.

Crie documentos de amostra

```
db.collection.insert([
  { "name": "John Doe*  ", "age": 30 },
  { "name": "Jane Smith  ", "age": 25 },
  { "name": "Bob Johnson", "age": 35 }
]);
```

Exemplo de consulta

```
db.collection.aggregate([
  {
    $project: {
      _id: 0,
      name: { $rtrim: { input: "$name", chars: " *" } },
      age: 1
    }
  }
]);
```

Saída

```
[
  { age: 30, name: 'John Doe' },
  { age: 25, name: 'Jane Smith' },
  { age: 35, name: 'Bob Johnson' }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$rtrim` comando, escolha a guia do idioma que você deseja 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

O estágio `$sample` de agregação no Amazon DocumentDB é usado para selecionar aleatoriamente um número específico de documentos de uma coleção. Isso é útil para tarefas como análise de dados, testes e geração de amostras para processamento posterior.

Parâmetros

- `size`: o número de documentos a serem selecionados aleatoriamente.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$sample` palco para selecionar aleatoriamente dois documentos da `temp` coleção.

Crie documentos de amostra

```
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") }
]);
```

Exemplo de consulta

```
db.temp.aggregate([
   { $sample: { size: 2 } }
])
```

Saída

```
{ "_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 mostram os resultados, 2 dos 10 documentos foram amostrados aleatoriamente. Agora você pode usar esses documentos para determinar uma média ou realizar min/max cálculos.

Exemplos de código

Para ver um exemplo de código para usar o `$sample` comando, escolha a guia do idioma que você deseja 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

Novo a partir da versão 5.0.

O `$search` operador no Amazon DocumentDB é usado para fornecer recursos de pesquisa de texto.

Parâmetros

Nenhum

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$search` operador para realizar uma consulta de pesquisa de texto.

Crie documentos de amostra

```
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." }
]);
```

Exemplo de consulta

```
db.textcollection.find(
  { $text: { $search: "John" } }
);
```

Saída

```
[
  {
    _id: 1,
    name: 'John Doe',
    description: 'This is a sample document about John Doe.'
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$search` comando, escolha a guia do idioma que você deseja 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

O `$second` operador no Amazon DocumentDB extrai o componente de segundos de uma data ou timestamp. Ele é usado para recuperar o valor de segundos de um campo de data ou carimbo de data/hora.

Parâmetros

- `expression`: o campo de data ou timestamp do qual extrair o valor dos segundos. Essa expressão pode ser um caminho de campo ou qualquer expressão válida que seja resolvida como data ou timestamp.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$second` operador para extrair o componente de segundos de um campo de data.

Crie documentos de amostra

```
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") }
]);
```

Exemplo de consulta

```
db.users.aggregate([{ $project: { name: 1, dobSeconds: { $second: "$dob" } } }])
```

Saída

```
[
  { "_id" : ObjectId("6089a9c306a829d1f8b456a1"), "name" : "John", "dobSeconds" : 45 },
  { "_id" : ObjectId("6089a9c306a829d1f8b456a2"), "name" : "Jane", "dobSeconds" : 59 },
  { "_id" : ObjectId("6089a9c306a829d1f8b456a3"), "name" : "Bob", "dobSeconds" : 0 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$second` comando, escolha a guia do idioma que você deseja 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

Novo da versão 8.0

Não é suportado pelo cluster elástico.

O estágio `$set` de agregação no Amazon DocumentDB permite que você adicione novos campos ou atualize os valores dos campos existentes na documentação durante um pipeline de agregação.

Parâmetros

- `expression`: A expressão a ser avaliada. Pode ser qualquer expressão de agregação válida, incluindo referências de campo e operações aritméticas.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do estágio de `$set` agregação para calcular totais multiplicando o `quantity` campo pelo campo. `price`

Crie documentos de amostra

```
db.inventory.insertMany([
  { item: "pencil", quantity: 100, price: 0.24},
  { item: "pen", quantity: 204, price: 1.78 }
]);
```

Exemplo de agregação

```
db.inventory.aggregate([
  {
    $set: {
```

```
      total: { $multiply: ["$quantity", "$price"] }
    }
  }
])
```

Saída

```
[
  {
    _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
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$set` comando, escolha a guia do idioma que você deseja 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

Novo a partir da versão 4.0.

O `$setDifference` operador no Amazon DocumentDB é usado para comparar dois conjuntos e retornar os elementos que estão no primeiro conjunto, mas não no segundo. Esse operador é útil para encontrar os elementos exclusivos entre dois conjuntos.

Parâmetros

- `firstSet`: O primeiro conjunto a ser comparado.
- `secondSet`: O segundo conjunto para comparar.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$setDifference` operador para encontrar os elementos exclusivos entre dois conjuntos.

Crie documentos de amostra

```
db.collection.insertMany([
  { _id: 1, fruits: ["apple", "banana", "cherry", "date"] },
  { _id: 2, fruits: ["banana", "cherry", "date", "elderberry"] }
]);
```

Exemplo de consulta

```
db.collection.aggregate([
  {
    $project: {
      uniqueFruits: { $setDifference: ["$fruits", ["banana", "cherry", "date"]] }
    }
  }
]);
```

Saída

```
[
  { "_id": 1, "uniqueFruits": ["apple"] },
  { "_id": 2, "uniqueFruits": ["elderberry"] }
]
```

A consulta executa as seguintes etapas:

1. Ele usa o `$project` palco para criar um novo campo `uniqueFruits` para cada documento.

2. O `$setDifference` operador compara a `fruits` matriz com a matriz `["banana", "cherry", "date"]` e retorna os elementos exclusivos na `fruits` matriz.

Exemplos de código

Para ver um exemplo de código para usar o `$setDifference` comando, escolha a guia do idioma que você deseja usar:

Node.js

Aqui está um exemplo de como usar o `$setDifference` operador em um Node.js aplicativo:

```
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

Aqui está um exemplo de como usar o `$setDifference` operador em um aplicativo 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

O `$setEquals` operador no Amazon DocumentDB é usado para determinar se dois conjuntos são iguais. Ele compara duas matrizes e retorna `true` se elas contêm os mesmos elementos distintos, independentemente da ordem.

Parâmetros

- `expression1`: a primeira matriz a ser comparada.
- `expression2`: a segunda matriz a ser comparada.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$setEquals` operador para comparar dois conjuntos de valores.

Crie documentos de amostra

```
db.collection.insertMany([
  { _id: 1, fruits: ["apple", "banana", "cherry"] },
  { _id: 2, fruits: ["banana", "apple", "cherry"] },
  { _id: 3, fruits: ["apple", "banana", "orange"] }
])
```

Exemplo de consulta

```
db.collection.find({
  $expr: {
    $setEquals: ["$fruits", ["apple", "banana", "cherry"]]
  }
})
```

Saída

```
{ "_id" : 1, "fruits" : [ "apple", "banana", "cherry" ] }
{ "_id" : 2, "fruits" : [ "banana", "apple", "cherry" ] }
```

A consulta usa o `$setEquals` operador para comparar o `fruits` campo de cada documento com a matriz`["apple", "banana", "cherry"]`. Os documentos em que o `fruits` campo é igual à matriz de comparação são retornados.

Exemplos de código

Para ver um exemplo de código para usar o `$setEquals` comando, escolha a guia do idioma que você deseja 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

O `$setIntersection` operador no Amazon DocumentDB é usado para retornar os elementos comuns entre duas ou mais matrizes. Esse operador é particularmente útil ao trabalhar com conjuntos de dados, permitindo que você encontre a interseção de vários conjuntos.

Parâmetros

- `array1`: a primeira matriz a se cruzar.
- `array2`: a segunda matriz a ser cruzada.
- `arrayN`: (opcional) Matrizes adicionais a serem cruzadas.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$setIntersection` operador para encontrar os elementos comuns entre duas matrizes.

Crie documentos de amostra

```
db.collection.insertMany([
  { _id: 1, colors: ["red", "blue", "green"] },
  { _id: 2, colors: ["blue", "yellow", "orange"] },
  { _id: 3, colors: ["red", "green", "purple"] }
])
```

Exemplo de consulta

```
db.collection.aggregate([
  { $project: {
      _id: 1,
      commonColors: { $setIntersection: ["$colors", ["red", "blue", "green"]] }
    }
  }
])
```

Saída

```
[
  { "_id": 1, "commonColors": ["red", "blue", "green"] },
  { "_id": 2, "commonColors": ["blue"] },
  { "_id": 3, "commonColors": ["red", "green"] }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$setIntersection` comando, escolha a guia do idioma que você deseja 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()
```

$ set IsSubset

O `$setIsSubset` operador no Amazon DocumentDB é usado para determinar se um conjunto de valores é um subconjunto de outro conjunto. É útil para realizar comparações e operações baseadas em conjuntos em campos de matriz.

Parâmetros

- `field`: o campo ao qual aplicar o `$setIsSubset` operador.
- `set`: O conjunto com o qual comparar o campo.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$setIsSubset` operador para verificar se o `tags` campo é um subconjunto do conjunto especificado.

Crie documentos de amostra

```
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"] }
]);
```

Exemplo de consulta

```
db.products.find({
  $expr: { $setIsSubset: [["tag1", "tag2"], "$tags"] }
})
```

*Nota: * `$setIsSubset` é um operador de agregação e não pode ser usado diretamente nas consultas find (). Neste exemplo, `$expr` é usado com `find()` para preencher a lacuna entre operadores de consulta e expressões de agregação.

Saída

```
[
  { "_id" : 1, "name" : "Product A", "tags" : [ "tag1", "tag2", "tag3" ] },
  { "_id" : 2, "name" : "Product B", "tags" : [ "tag1", "tag2" ] }
]
```

A consulta retorna os documentos em que o `tags` campo é um subconjunto do conjunto`["tag1", "tag2"]`.

Exemplos de código

Para ver um exemplo de código para usar o `$setIsSubset` comando, escolha a guia do idioma que você deseja 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

O operador `$setUnion` de agregação no Amazon DocumentDB é usado para combinar dois ou mais conjuntos de valores e retornar um conjunto que contém todos os elementos exclusivos dos conjuntos de entrada. Esse operador é útil quando você precisa realizar operações baseadas em conjuntos em campos de matriz em seus documentos.

Parâmetros

- `expression1`: uma expressão que se resolve em uma matriz.
- `expression2`: uma expressão que se resolve em uma matriz.
- `expressionN`: expressões adicionais que se resolvem em matrizes (opcional).

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$setUnion` operador para combinar os elementos exclusivos de dois campos de matriz em uma coleção.

Crie documentos de amostra

```
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"] }
]);
```

Exemplo de consulta

```
db.users.aggregate([
  {
    $project: {
      name: 1,
      allInterests: { $setUnion: ["$hobbies", "$skills"] }
    }
  }
]);
```

Saída

```
[
  { "_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" ] }
]
```

Neste exemplo, o `$setUnion` operador é usado para combinar os elementos exclusivos dos campos `hobbies` e da `skills` matriz para cada documento do usuário. O `allInterests` campo resultante contém a união de todos os hobbies e habilidades exclusivos de cada usuário.

Exemplos de código

Para ver um exemplo de código para usar o `$setUnion` comando, escolha a guia do idioma que você deseja 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()
```

$set WindowFields

Novo na versão 8.0.2.

O `$setWindowFields` estágio no Amazon DocumentDB executa operações em uma extensão específica de documentos (uma janela) em uma partição e retorna os resultados com base no operador de janela escolhido. O estágio agrupa os documentos de entrada em partições, opcionalmente os classifica e, para cada documento, calcula um ou mais campos de saída em uma janela de documentos ao redor.

Sintaxe

```
{
  $setWindowFields: {
    partitionBy: <expression>,
    sortBy: { <sortField1>: <sortOrder>, ... },
    output: {
      <outputField1>: {
        <windowOperator>: <expression>,
        window: {
          documents: [ <lowerBound>, <upperBound> ],
          range: [ <lowerBound>, <upperBound> ],
          unit: <time unit>
        }
      },
      ...
    }
  }
}
```

Parâmetros

- `partitionBy`: opcional. Uma única expressão usada para agrupar os documentos em partições. Isso pode ser um caminho de campo (por exemplo`"$state"`) ou outra expressão. Se omitidos, todos os documentos de entrada pertencerão a uma única partição.
- `sortBy`: opcional. Um documento que especifica o campo ou os campos pelos quais classificar os documentos em cada partição, no formato `{ <field>: 1 }` de ordem crescente ou `{ <field>: -1 }` decrescente. A `sortBy` é necessário quando você usa limites de janela baseados em documentos ou em intervalos e quando você usa um dos operadores de classificação.
- `output`: Obrigatório. Um documento que especifica os campos a serem anexados a cada documento. Você pode especificar vários campos de saída, mas cada campo de saída usa exatamente um operador de janela, seu argumento e um `window` documento opcional que define os limites da janela:`{ <field>: { <windowOperator>: <expression>, window: { ... } } }`.

Limites da janela

Em cada `output` campo, você pode definir opcionalmente os limites da janela com um `window` documento. Uma janela pode ser definida pela posição do documento ou por um intervalo de valores.

- `documents`: especifica a janela como um número de documentos antes e depois do documento atual, no formulário`[ <lowerBound>, <upperBound> ]`. Cada limite é um deslocamento inteiro (por exemplo, `-1` ou`2`) ou uma das palavras-chave e. `"unbounded"` `"current"` `"unbounded"`estende a janela até o início da partição quando usada como limite inferior ou até o final da partição quando usada como limite superior. `"current"`significa o documento atual. Uma `documents` janela requer um, `sortBy` a menos que ambos os limites sejam`"unbounded"`.
- `range`: especifica a janela como um intervalo de valores de `sortBy` campo em relação ao documento atual, no formulário`[ <lowerBound>, <upperBound> ]`. Cada limite é um número adicionado ao `sortBy` valor do documento atual ou à `"unbounded"` palavra-chave. Uma `range` janela exige exatamente um `sortBy` campo ascendente e não aceita a `"current"` palavra-chave.
- `unit`: opcional. Usado com `range` para interpretar os limites numéricos da janela como um intervalo de tempo em que o `sortBy` campo contém valores de data. O Amazon DocumentDB oferece suporte às seguintes unidades: `"year"` `"quarter"` `"month"``"week"`,`"day"`,`"hour"`,`"minute"`,`"second"`, e. `"millisecond"` Quando `unit` especificado, os `range` limites devem ser inteiros.

Quando o `window` documento é omitido, a janela padrão é`documents: ["unbounded", "unbounded"]`, que abrange toda a partição.

> **Note**
>
> Quando você usa uma `range` janela sobre valores de data, recomendamos uma duração fixa, `unit` como `"day"` (ou menor), em vez de, ou `"month"` `"quarter"``"year"`, cujos comprimentos dependentes do calendário possam produzir limites de janela ambíguos.

Operadores de janela compatíveis

O Amazon DocumentDB oferece suporte aos seguintes 26 operadores de janela no `output` campo do `$setWindowFields` palco:

- `$sum`
- `$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**
>
> O `$rand` operador não é suportado em um operador de janela no `$setWindowFields` palco.

Janelas vazias

Quando uma janela não contém documentos (por exemplo, uma `range` janela sem valores no intervalo), o resultado depende do operador: `$count` e `$sum` retorna`0`;`$push`,,,, `$addToSet` `$firstN` `$lastN` `$minN` `$maxN``$topN`, e retorna uma matriz vazia; `$bottomN` `$percentile` retorna uma matriz de `null` valores (um por percentil solicitado); todos os outros operadores retornam. `null`

Exemplo (MongoDB Shell)

O exemplo a seguir particiona os documentos por`region`, classifica cada partição por `day` e calcula um total em execução `revenue` desde o início da partição até o documento atual.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.sales.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$region",
      sortBy: { day: 1 },
      output: {
        runningRevenue: {
          $sum: "$revenue",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Saída

```
[
  { "_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 é aumentado com `runningRevenue` a soma cumulativa de `revenue` dentro de sua partição até e incluindo o documento atual.

Exemplos de código

Para ver um exemplo de código para usar o `$setWindowFields` estágio, escolha a guia do idioma que você deseja 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 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()
```

$ sigmóide

Novo na versão 8.0.1.

O `$sigmoid` operador no Amazon DocumentDB aplica a função sigmóide logística,`1 / (1 + e^(-x))`, a uma entrada numérica. A função mapeia qualquer número real para um valor entre 0 e 1, produzindo uma S-shaped curva. Isso é útil quando você precisa normalizar valores na faixa de 0 a 1, como converter pontuações brutas em probabilidades ou pontuações de relevância.

Parâmetros

- `expression`: uma expressão que se resolve em um valor numérico.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$sigmoid` operador para converter valores logit em probabilidades.

Crie documentos de amostra

```
db.predictions.insertMany([
  {_id: 1, logit: -2},
  {_id: 2, logit: 0},
  {_id: 3, logit: 2}
]);
```

Exemplo de consulta

```
db.predictions.aggregate([
  { $project: { probability: { $sigmoid: "$logit" } } }
]);
```

Saída

```
[
  {_id: 1, probability: 0.11920292202211755},
  {_id: 2, probability: 0.5},
  {_id: 3, probability: 0.8807970779778823}
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$sigmoid` operador, escolha a guia do idioma que você deseja 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

Novo na versão 8.0.1.

O `$sin` operador no Amazon DocumentDB retorna o seno de um valor que é medido em radianos. Use-o no pipeline de agregação para realizar cálculos trigonométricos em campos numéricos.

Parâmetros

- `expression`: uma expressão que se resolve em um número em radianos.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$sin` operador para calcular o seno dos valores angulares em radianos.

Crie documentos de amostra

```
db.angles.insertMany([
  { "_id": 1, "angle": 0 },
  { "_id": 2, "angle": 0.5235987755982988 },
```

```
  { "_id": 3, "angle": 1.5707963267948966 }
]);
```

Exemplo de consulta

```
db.angles.aggregate([
  { $project: {
    "sine": { $sin: "$angle" }
  }}
]);
```

Saída

```
[
  { "_id": 1, "sine": 0 },
  { "_id": 2, "sine": 0.49999999999999994 },
  { "_id": 3, "sine": 1 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$sin` operador, escolha a guia do idioma que você deseja 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

Novo na versão 8.0.1.

O `$sinh` operador no Amazon DocumentDB retorna o seno hiperbólico de um valor. Use-o no pipeline de agregação para realizar cálculos trigonométricos hiperbólicos em campos numéricos.

Parâmetros

- `expression`: uma expressão que se resolve em um número.

O tipo de retorno é `double` por padrão. Se a entrada for decimal de 128 bits, a saída também será decimal de 128 bits.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$sinh` operador para calcular o seno hiperbólico dos valores numéricos.

Crie documentos de amostra

```
db.values.insertMany([
  { "_id": 1, "value": 0 },
  { "_id": 2, "value": 1 },
  { "_id": 3, "value": -1 }
]);
```

Exemplo de consulta

```
db.values.aggregate([
  { $project: {
    "result": { $sinh: "$value" }
  }}
]);
```

Saída

```
[
  { "_id": 1, "result": 0 },
  { "_id": 2, "result": 1.1752011936438014 },
  { "_id": 3, "result": -1.1752011936438014 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$sinh` operador, escolha a guia do idioma que você deseja 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

No Amazon DocumentDB, o `$skip` operador é usado para compensar o ponto de partida dos resultados da consulta, permitindo que você recupere um subconjunto específico dos documentos

correspondentes. Isso é particularmente útil em cenários de paginação, nos quais você deseja recuperar páginas subsequentes de resultados.

Parâmetros

- `skip`: o número de documentos a serem ignorados antes de devolver os documentos restantes.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$skip` operador para recuperar a segunda página de resultados (documentos 11-20) de uma coleção.

Crie documentos de amostra

```
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" }
]);
```

Exemplo de consulta

```
db.collection.find({}, { "name": 1 })
             .skip(10)
             .limit(10);
```

Saída

```
[
  { "_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" }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$skip` comando, escolha a guia do idioma que você deseja 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

O operador `$slice` de agregação permite que você retorne um subconjunto de uma matriz percorrendo a matriz do início ou do final da matriz. Isso é usado para exibir um número limitado de itens de um campo de matriz, como os N itens superiores ou inferiores.

Parâmetros

- `array`: O campo da matriz a ser fatiado.
- n: um número inteiro que especifica o número de elementos a serem retornados. Um valor positivo começa no início da matriz, enquanto um valor negativo começa no final da matriz.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar `$slice` para devolver os dois primeiros doces favoritos de cada chef.

Crie documentos de amostra

```
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" ] }
]);
```

Exemplo de consulta

```
db.sweets.aggregate([
  { $project: { _id: 0, name: 1, topTwoFavorites: { $slice: [ "$favorites", 2 ] } } }
]);
```

Saída

```
[
  { name: 'Alvin', topTwoFavorites: [ 'chocolate', 'cake' ] },
  { name: 'Tom', topTwoFavorites: [ 'donuts', 'pudding' ] },
  { name: 'Jessica', topTwoFavorites: [ 'fudge', 'smores' ] },
  { name: 'Rachel', topTwoFavorites: [ 'ice cream' ] }
]
```

Neste exemplo, o `$slice` operador é usado para extrair os dois primeiros elementos da `favorites` matriz para cada documento.

Exemplos de código

Para ver um exemplo de código para usar o `$slice` comando, escolha a guia do idioma que você deseja 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

O `$size` operador é usado para retornar a contagem de itens em um campo de matriz. Isso pode ser usado para determinar o número de elementos em uma matriz armazenada em um documento.

Parâmetros

- `field`: o caminho do campo cujo tamanho de matriz você deseja retornar.

Exemplo (MongoDB Shell)

Este exemplo mostra como usar o `$size` operador para retornar a contagem de equipes que cada usuário está seguindo.

Crie documentos de amostra

```
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"] }
]);
```

Exemplo de consulta

```
db.profiles.aggregate([
  {
    $project: {
      _id: 0,
      name: 1,
      "numberOfTeams": { $size: "$teams" }
    }
  }
])
```

Saída

```
[
  { name: 'John Doe', numberOfTeams: 3 },
  { name: 'Jane Smith', numberOfTeams: 2 },
  { name: 'Bob Johnson', numberOfTeams: 3 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$size` comando, escolha a guia do idioma que você deseja 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

O estágio `$sort` de agregação ordena documentos no pipeline com base nos valores de campo especificados. Os documentos são organizados em ordem crescente ou decrescente de acordo com os critérios de classificação fornecidos.

Parâmetros

- `field`: O nome do campo pelo qual classificar.
- `order`: Use `1` para ordem crescente ou `-1` para ordem decrescente.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$sort` estágio para ordenar produtos por preço em ordem decrescente.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.products.aggregate([
  { $sort: { price: -1 } }
]);
```

Saída

```
[
  { _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 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o estágio de `$sort` agregação, escolha a guia do idioma que você deseja 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()
```

Classificação incremental

A classificação incremental é uma otimização na qual o Amazon DocumentDB usa um índice que já ordena os principais campos de uma classificação de vários campos. Em seguida, o Amazon DocumentDB classifica somente os campos restantes em cada grupo de documentos que compartilham os mesmos valores principais. Em vez de classificar todo o conjunto de resultados na memória como um bloco (uma classificação completa), o Amazon DocumentDB lê documentos em ordem de indexação. Em seguida, ele executa vários tipos pequenos e independentes, um por grupo. Os resultados são idênticos a uma classificação completa. Somente o plano de consulta e a quantidade de memória e trabalho que ele usa mudam.

Benefícios

A classificação incremental oferece os seguintes benefícios:

- Menor uso de memória e menos vazamentos de classificação baseados em disco. O Amazon DocumentDB solicita um pequeno grupo por vez, em vez de todo o conjunto de resultados de uma só vez.
- Resultados mais rápidos com**`limit`**. O Amazon DocumentDB pode emitir documentos classificados antes de ler toda a entrada. Uma classificação seguida por um `limit` (por exemplo, uma consulta no topo N ou paginada) pode ser interrompida assim que produzir documentos suficientes.

Quando o Amazon DocumentDB escolhe uma classificação incremental

Por exemplo, com uma classificação ativada `{ customerId: 1, orderDate: -1 }` e um índice ativada somente`{ customerId: 1 }`, o índice já retorna documentos ordenados por`customerId`.

O Amazon DocumentDB só precisa solicitar cada grupo de documentos que compartilham o mesmo `customerId` por`orderDate`. O campo inicial já ordenado (`customerId`) é a chave pré-classificada. Dependendo da classificação que um índice já fornece, o Amazon DocumentDB escolhe um dos três comportamentos:

- O índice ordena cada chave de classificação. Nenhum tipo é necessário. O Amazon DocumentDB lê documentos diretamente na ordem solicitada.
- O índice ordena um prefixo inicial das chaves de classificação. O Amazon DocumentDB usa uma classificação incremental, ordenando somente as chaves finais dentro de cada grupo pré-classificado.
- Nenhum índice ordena qualquer chave de classificação principal. O Amazon DocumentDB realiza uma classificação completa de toda a entrada.

Requisitos e limitações

A classificação incremental requer o Amazon DocumentDB 8.0.2 ou posterior, executando o planejador versão 3.0 (o planejador de consultas padrão a partir do Amazon DocumentDB 8.0.2). Para obter mais informações sobre as versões do planejador, consultePlanejador de consultas v3.

O Amazon DocumentDB não pode usar um índice parcial ou um índice multichave (matriz) para fornecer o prefixo pré-classificado para uma classificação incremental. Nesses casos, o Amazon DocumentDB volta a ser um tipo completo. Isso nunca afeta a exatidão dos resultados.

Verificando uma classificação incremental com **`explain()`**

Use `explain()` para confirmar que uma consulta usa uma classificação incremental. No plano vencedor, uma classificação incremental aparece como um `SORT` estágio que inclui uma `presortedKeys` matriz listando os principais campos de classificação que o índice já ordenou. Uma classificação completa aparece como um `SORT` estágio sem uma `presortedKeys` matriz. O `presortedKeys` campo é uma DocumentDB-specific adição da Amazon à `explain()` saída e não aparece nos planos de explicação do 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

Novo a partir da versão 8.0

Não é compatível com o cluster elástico.

O `$sortArray` operador no Amazon DocumentDB classifica os elementos de uma matriz. Para matrizes de valores escalares, a matriz é classificada pelo valor do elemento. Para matrizes de documentos, a matriz é classificada pelo campo ou campos especificados. O operador retorna a matriz classificada em ordem crescente (1) ou decrescente (`-1`).

> **Note**
>
> `$sortArray`é suportado no `$project` estágio do pipeline de agregação com o planejador de consultas versão 3.

Sintaxe

```
{ $sortArray: { input: <array>, sortBy: <sort specification> } }
```

Parâmetros

- `input`: A matriz a ser classificada.
- `sortBy`: A ordem de classificação. Para uma matriz de valores escalares, especifique 1 para ascendente ou `-1` decrescente. Para uma matriz de documentos, especifique um documento do formulário `{ field: 1 }` ou `{ field: -1 }` de cada campo pelo qual classificar.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$sortArray` operador para classificar os elementos de uma matriz em ordem crescente.

Crie documentos de amostra

```
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]}
]);
```

Exemplo de consulta

```
db.miles.aggregate([
  {
    $project: {
      _id: 1,
      member_since: 1,
      credit_card: 1,
      sorted_flight_miles: { $sortArray: { input: "$flight_miles", sortBy: 1 } }
    }
  }
]);
```

Saída

```
{ "_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 ] }
```

Neste exemplo, o `$sortArray` operador classifica a `flight_miles` matriz em ordem crescente. O `sorted_flight_miles` campo resultante na saída mostra os elementos da matriz em ordem ordenada.

Exemplos de código

Para ver um exemplo de código para usar o `$sortArray` comando, escolha a guia do idioma que você deseja usar:

Node.js

Aqui está um exemplo de uso do `$sortArray` operador em um Node.js aplicativo:

```
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

Aqui está um exemplo de uso do `$sortArray` operador em um aplicativo 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

Novo na versão 8.0.1.

O estágio de `$sortByCount` agregação no Amazon DocumentDB agrupa documentos recebidos com base no valor de uma expressão especificada, depois calcula a contagem de documentos em cada grupo distinto e classifica os resultados por contagem em ordem decrescente. É equivalente a um `$group` estágio com um `$sum` acumulador seguido por um `$sort` estágio no campo de contagem.

Parâmetros

- `expression`: A expressão pela qual agrupar. Isso pode ser um caminho de campo (prefixado com$) ou qualquer expressão de agregação válida.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$sortByCount` estágio para contar e classificar documentos por um campo de categoria.

Crie documentos de amostra

```
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" }
]);
```

Exemplo de consulta

```
db.products.aggregate([
  { $sortByCount: "$category" }
]);
```

Saída

```
[
  { "_id": "electronics", "count": 3 },
  { "_id": "food", "count": 2 },
  { "_id": "tools", "count": 1 }
]
```

Os resultados são classificados automaticamente `count` em ordem decrescente.

Exemplos de código

Para ver um exemplo de código para usar o `$sortByCount` estágio, escolha a guia do idioma que você deseja 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('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

O operador `$split` de agregação no Amazon DocumentDB é usado para dividir uma string em uma matriz de substrings, com base em um delimitador especificado. Isso pode ser útil para analisar campos de cadeia de caracteres complexos e extrair componentes individuais para processamento adicional.

Parâmetros

- `string`: A string a ser dividida.
- `delimiter`: o caractere ou cadeia de caracteres usado para dividir a cadeia de caracteres de entrada.

Exemplo (MongoDB Shell)

Neste exemplo, usamos `$split` para separar os componentes de um campo “Mesa” em uma matriz, facilitando o processamento dos dados.

Crie documentos de amostra

```
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" }
]);
```

Exemplo de consulta

```
db.people.aggregate([
  { $project: { parts: { $split: ["$Desk", "-"] } } }
]);
```

Saída

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

A saída de `$split` cria uma matriz que pode ser usada no aplicativo para exibir as informações para os funcionários.

Exemplos de código

Para ver um exemplo de código para usar o `$split` comando, escolha a guia do idioma que você deseja 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

Novo a partir da versão 4.0.

O `$sqrt` operador no Amazon DocumentDB é usado para calcular a raiz quadrada de um número.

Parâmetros

- `expression`: o argumento pode ser qualquer expressão válida, desde que seja resolvido em um número não negativo.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$sqrt` operador para calcular a raiz quadrada de um número.

Crie documentos de amostra

```
db.numbers.insertMany([
  { "_id": 1, "number": 16 },
  { "_id": 2, "number": 36 },
  { "_id": 3, "number": 64 }
]);
```

Exemplo de consulta

```
db.numbers.aggregate([
  { $project: {
    "_id": 1,
    "square_root": { $sqrt: "$number" }
  }}
]);
```

Saída

```
[
  { _id: 1, square_root: 4 },
  { _id: 2, square_root: 6 },
  { _id: 3, square_root: 8 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$sqrt` comando, escolha a guia do idioma que você deseja 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()
```

$ std DevPop

Novo na versão 8.0.1.

O `$stdDevPop` operador no Amazon DocumentDB calcula o desvio padrão populacional dos valores numéricos. Como acumulador, ele calcula o desvio padrão da população em documentos dentro de um grupo no `$group` estágio de um pipeline de agregação. Como expressão, ele calcula o

desvio padrão da população de uma matriz de números. O desvio padrão da população usa N como divisor (não N-1). Non-numeric os valores são ignorados. Se não houver valores numéricos, ele retornará`null`. Se houver somente um valor numérico, ele retornará`0`.

Parâmetros

- `expression`: uma expressão que resolve para um valor numérico ou uma matriz de valores numéricos.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$stdDevPop` operador para calcular o desvio padrão da população das pontuações por assunto.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.scores.aggregate([
  { $group: {
      _id: "$subject",
      stdDev: { $stdDevPop: "$score" }
    }}
]);
```

Saída

```
[
```

```
  { "_id": "math", "stdDev": 10.75174404457249 },
  { "_id": "science", "stdDev": 12.441864811996632 }
]
```

Exemplo de uso de expressão (MongoDB Shell)

O `$stdDevPop` operador também pode ser usado como uma expressão dentro de um `$project` estágio para calcular o desvio padrão da população de um campo de matriz.

Crie documentos de amostra

```
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] }
]);
```

Exemplo de consulta

```
db.experiments.aggregate([
  { $project: {
      stdDev: { $stdDevPop: "$measurements" }
    }}
]);
```

Saída

```
[
  { "_id": 1, "stdDev": 2.8284271247461903 },
  { "_id": 2, "stdDev": 0 },
  { "_id": 3, "stdDev": 2.8284271247461903 }
]
```

Exemplo de uso do operador de janela (MongoDB Shell)

Novo na versão 8.0.2.

O `$stdDevPop` operador também pode ser usado como operador de janela no `$setWindowFields` palco. Nesse contexto, ele retorna o desvio padrão da população da expressão especificada para os documentos em cada janela. Você especifica o operador no `output` campo e, opcionalmente, define os limites da janela com um `window` documento.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

O exemplo a seguir particiona os documentos por`sensor`, classifica cada partição por `time` e retorna o desvio padrão da população em execução `reading` desde o início da partição até o documento atual.

```
db.sensorData.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$sensor",
      sortBy: { time: 1 },
      output: {
        runningStdDev: {
          $stdDevPop: "$reading",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Saída

```
[
  { "_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 é aumentado com `runningStdDev` o desvio padrão da população `reading` dentro de sua partição até e incluindo o documento atual. Quando a janela contém um único valor, `$stdDevPop` retorna`0`.

Exemplos de código

Para ver um exemplo de código para usar o `$stdDevPop` operador, escolha a guia do idioma que você deseja usar. Os exemplos a seguir mostram o uso do acumulador (in`$group`), o uso da expressão (in`$project`) e o uso do operador de janela (in`$setWindowFields`):

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const uri = 'mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false';
  const client = new MongoClient(uri);

  try {
    await client.connect();
    const db = client.db('test');

    // Accumulator usage: stdDevPop across grouped documents
    const scores = db.collection('scores');
    await scores.insertMany([
      { subject: "math", score: 60 },
      { subject: "math", score: 75 },
      { subject: "math", score: 85 },
      { subject: "math", score: 92 },
      { subject: "math", score: 78 },
      { subject: "science", score: 55 },
      { subject: "science", score: 70 },
      { subject: "science", score: 82 },
      { subject: "science", score: 91 },
      { subject: "science", score: 67 }
    ]);
    const accumulatorResult = await scores.aggregate([
      { $group: {
          _id: "$subject",
          stdDev: { $stdDevPop: "$score" }
        }}
```

```
    ]).toArray();
    console.log('Accumulator result:', accumulatorResult);

    // Expression usage: stdDevPop of an array field
    const experiments = db.collection('experiments');
    await experiments.insertMany([
      { _id: 1, measurements: [10, 12, 14, 16, 18] },
      { _id: 2, measurements: [5, 5, 5, 5, 5] },
      { _id: 3, measurements: [2, 4, 6, 8, 10] }
    ]);
    const expressionResult = await experiments.aggregate([
      { $project: {
          stdDev: { $stdDevPop: "$measurements" }
        }}
    ]).toArray();
    console.log('Expression result:', expressionResult);

    // Window operator usage: running stdDevPop within each partition
    const sensorData = db.collection('sensorData');
    await sensorData.insertMany([
      { _id: 1, sensor: "A", time: 1, reading: 4 },
      { _id: 2, sensor: "A", time: 2, reading: 10 },
      { _id: 3, sensor: "A", time: 3, reading: 10 },
      { _id: 4, sensor: "B", time: 1, reading: 10 },
      { _id: 5, sensor: "B", time: 2, reading: 20 }
    ]);
    const windowResult = await sensorData.aggregate([
      {
        $setWindowFields: {
          partitionBy: "$sensor",
          sortBy: { time: 1 },
          output: {
            runningStdDev: {
              $stdDevPop: "$reading",
              window: { documents: ["unbounded", "current"] }
            }
          }
        }
      }
    ]).toArray();
    console.log('Window result:', windowResult);

  } finally {
    await client.close();
```

```
  }
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    try:
        db = client['test']

        # Accumulator usage: stdDevPop across grouped documents
        scores = db['scores']
        scores.insert_many([
            { 'subject': 'math', 'score': 60 },
            { 'subject': 'math', 'score': 75 },
            { 'subject': 'math', 'score': 85 },
            { 'subject': 'math', 'score': 92 },
            { 'subject': 'math', 'score': 78 },
            { 'subject': 'science', 'score': 55 },
            { 'subject': 'science', 'score': 70 },
            { 'subject': 'science', 'score': 82 },
            { 'subject': 'science', 'score': 91 },
            { 'subject': 'science', 'score': 67 }
        ])
        accumulator_result = list(scores.aggregate([
            { '$group': {
                '_id': '$subject',
                'stdDev': { '$stdDevPop': '$score' }
            }}
        ]))
        print('Accumulator result:', accumulator_result)

        # Expression usage: stdDevPop of an array field
        experiments = db['experiments']
        experiments.insert_many([
            { '_id': 1, 'measurements': [10, 12, 14, 16, 18] },
```

```
            { '_id': 2, 'measurements': [5, 5, 5, 5, 5] },
            { '_id': 3, 'measurements': [2, 4, 6, 8, 10] }
        ])
        expression_result = list(experiments.aggregate([
            { '$project': {
                'stdDev': { '$stdDevPop': '$measurements' }
            }}
        ]))
        print('Expression result:', expression_result)

        # Window operator usage: running stdDevPop within each partition
        sensor_data = db['sensorData']
        sensor_data.insert_many([
            { '_id': 1, 'sensor': 'A', 'time': 1, 'reading': 4 },
            { '_id': 2, 'sensor': 'A', 'time': 2, 'reading': 10 },
            { '_id': 3, 'sensor': 'A', 'time': 3, 'reading': 10 },
            { '_id': 4, 'sensor': 'B', 'time': 1, 'reading': 10 },
            { '_id': 5, 'sensor': 'B', 'time': 2, 'reading': 20 }
        ])
        window_result = list(sensor_data.aggregate([
            {
                '$setWindowFields': {
                    'partitionBy': '$sensor',
                    'sortBy': { 'time': 1 },
                    'output': {
                        'runningStdDev': {
                            '$stdDevPop': '$reading',
                            'window': { 'documents': ['unbounded', 'current'] }
                        }
                    }
                }
            }
        ]))
        print('Window result:', window_result)

    finally:
        client.close()

example()
```

$ std DevSamp

Novo na versão 8.0.1.

O `$stdDevSamp` operador no Amazon DocumentDB calcula o desvio padrão da amostra de valores numéricos. Como acumulador, ele calcula o desvio padrão da amostra entre documentos dentro de um grupo no `$group` estágio de um pipeline de agregação. Como expressão, ele calcula o desvio padrão da amostra de uma matriz de números. O desvio padrão da amostra é usado N-1 como divisor (correção de Bessel). Non-numeric os valores são ignorados. Se houver menos de dois valores numéricos, ele retornará`null`.

Parâmetros

- `expression`: uma expressão que se resolve em um valor numérico ou em uma matriz de valores numéricos.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$stdDevSamp` operador para calcular o desvio padrão da amostra das pontuações por assunto.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.scores.aggregate([
  { $group: {
      _id: "$subject",
      stdDev: { $stdDevSamp: "$score" }
    }}
```

```
]);
```

Saída

```
[
  { "_id": "math", "stdDev": 6.454972243679028 },
  { "_id": "science", "stdDev": 6.454972243679028 }
]
```

Exemplo de uso de expressão (MongoDB Shell)

O `$stdDevSamp` operador também pode ser usado como uma expressão dentro de um `$project` estágio para calcular o desvio padrão da amostra de um campo de matriz.

Crie documentos de amostra

```
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] }
]);
```

Exemplo de consulta

```
db.experiments.aggregate([
  { $project: {
      stdDev: { $stdDevSamp: "$measurements" }
    }}
]);
```

Saída

```
[
  { "_id": 1, "stdDev": 3.1622776601683795 },
  { "_id": 2, "stdDev": 0 },
  { "_id": 3, "stdDev": 3.1622776601683795 }
]
```

Exemplo de uso do operador de janela (MongoDB Shell)

Novo na versão 8.0.2.

O `$stdDevSamp` operador também pode ser usado como operador de janela no `$setWindowFields` palco. Nesse contexto, ele retorna o desvio padrão da amostra da expressão especificada para os documentos em cada janela. Você especifica o operador no `output` campo e, opcionalmente, define os limites da janela com um `window` documento.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

O exemplo a seguir particiona os documentos por`sensor`, classifica cada partição por `time` e retorna o desvio padrão da amostra em execução `value` desde o início da partição até o documento atual.

```
db.sensorReadings.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$sensor",
      sortBy: { time: 1 },
      output: {
        runningStdDev: {
          $stdDevSamp: "$value",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Saída

```
[
  { "_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 é aumentado com `runningStdDev` o desvio padrão da amostra de `value` dentro de sua partição até e incluindo o documento atual. Quando a janela contém menos de dois valores (o primeiro documento em cada partição), `$stdDevSamp` retorna`null`.

Exemplos de código

Para ver um exemplo de código para usar o `$stdDevSamp` operador, escolha a guia do idioma que você deseja usar. Os exemplos a seguir mostram o uso do acumulador (in`$group`), o uso da expressão (in`$project`) e o uso do operador de janela (in`$setWindowFields`):

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const uri = 'mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false';
  const client = new MongoClient(uri);

  try {
    await client.connect();
    const db = client.db('test');

    // Accumulator usage: stdDevSamp across grouped documents
    const scores = db.collection('scores');
    await scores.insertMany([
      { subject: "math", score: 80 },
      { subject: "math", score: 90 },
      { subject: "math", score: 85 },
      { subject: "math", score: 95 },
      { subject: "science", score: 70 },
      { subject: "science", score: 75 },
      { subject: "science", score: 80 },
      { subject: "science", score: 85 }
    ]);
    const accumulatorResult = await scores.aggregate([
```

```
  { $group: {
      _id: "$subject",
      stdDev: { $stdDevSamp: "$score" }
    }}
]).toArray();
console.log('Accumulator result:', accumulatorResult);

// Expression usage: stdDevSamp of an array field
const experiments = db.collection('experiments');
await experiments.insertMany([
  { _id: 1, measurements: [10, 12, 14, 16, 18] },
  { _id: 2, measurements: [5, 5, 5, 5, 5] },
  { _id: 3, measurements: [2, 4, 6, 8, 10] }
]);
const expressionResult = await experiments.aggregate([
  { $project: {
      stdDev: { $stdDevSamp: "$measurements" }
    }}
]).toArray();
console.log('Expression result:', expressionResult);

// Window operator usage: running stdDevSamp within each partition
const sensorReadings = db.collection('sensorReadings');
await sensorReadings.insertMany([
  { _id: 1, sensor: "A", time: 1, value: 2 },
  { _id: 2, sensor: "A", time: 2, value: 4 },
  { _id: 3, sensor: "A", time: 3, value: 6 },
  { _id: 4, sensor: "B", time: 1, value: 10 },
  { _id: 5, sensor: "B", time: 2, value: 12 }
]);
const windowResult = await sensorReadings.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$sensor",
      sortBy: { time: 1 },
      output: {
        runningStdDev: {
          $stdDevSamp: "$value",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]).toArray();
```

```
    console.log('Window result:', windowResult);

  } finally {
    await client.close();
  }
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')

    try:
        db = client['test']

        # Accumulator usage: stdDevSamp across grouped documents
        scores = db['scores']
        scores.insert_many([
            { 'subject': 'math', 'score': 80 },
            { 'subject': 'math', 'score': 90 },
            { 'subject': 'math', 'score': 85 },
            { 'subject': 'math', 'score': 95 },
            { 'subject': 'science', 'score': 70 },
            { 'subject': 'science', 'score': 75 },
            { 'subject': 'science', 'score': 80 },
            { 'subject': 'science', 'score': 85 }
        ])
        accumulator_result = list(scores.aggregate([
            { '$group': {
                '_id': '$subject',
                'stdDev': { '$stdDevSamp': '$score' }
            }}
        ]))
        print('Accumulator result:', accumulator_result)

        # Expression usage: stdDevSamp of an array field
        experiments = db['experiments']
```

```
        experiments.insert_many([
            { '_id': 1, 'measurements': [10, 12, 14, 16, 18] },
            { '_id': 2, 'measurements': [5, 5, 5, 5, 5] },
            { '_id': 3, 'measurements': [2, 4, 6, 8, 10] }
        ])
        expression_result = list(experiments.aggregate([
            { '$project': {
                'stdDev': { '$stdDevSamp': '$measurements' }
            }}
        ]))
        print('Expression result:', expression_result)

        # Window operator usage: running stdDevSamp within each partition
        sensor_readings = db['sensorReadings']
        sensor_readings.insert_many([
            { '_id': 1, 'sensor': 'A', 'time': 1, 'value': 2 },
            { '_id': 2, 'sensor': 'A', 'time': 2, 'value': 4 },
            { '_id': 3, 'sensor': 'A', 'time': 3, 'value': 6 },
            { '_id': 4, 'sensor': 'B', 'time': 1, 'value': 10 },
            { '_id': 5, 'sensor': 'B', 'time': 2, 'value': 12 }
        ])
        window_result = list(sensor_readings.aggregate([
            {
                '$setWindowFields': {
                    'partitionBy': '$sensor',
                    'sortBy': { 'time': 1 },
                    'output': {
                        'runningStdDev': {
                            '$stdDevSamp': '$value',
                            'window': { 'documents': ['unbounded', 'current'] }
                        }
                    }
                }
            }
        ]))
        print('Window result:', window_result)

    finally:
        client.close()

example()
```

$ str LenBytes

O `$strLenBytes` operador no Amazon DocumentDB é usado para determinar o tamanho de uma string em bytes. Isso é útil quando você precisa entender o tamanho de armazenamento de um campo de string, especialmente ao lidar com caracteres Unicode que podem usar mais de um byte por caractere.

Parâmetros

- `expression`: a expressão em cadeia de caracteres para calcular o comprimento de.

Exemplo (MongoDB Shell)

Este exemplo demonstra como usar o `$strLenBytes` operador para calcular o comprimento dos campos de string em bytes.

Crie documentos de amostra

```
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" }
]);
```

Exemplo de consulta

```
db.people.aggregate([
  {
    $project: {
      "Desk": 1,
      "length": { $strLenBytes: "$Desk" }
    }
  }
])
```

Saída

```
{ "_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 }
```

Observe que o comprimento da string “Düsseldorf-BVV-021” é de 19 bytes, o que é diferente do número de pontos de código (18) devido ao caractere Unicode “Ü” ocupar 2 bytes.

Exemplos de código

Para ver um exemplo de código para usar o `$strLenBytes` comando, escolha a guia do idioma que você deseja 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

O `$strLenCP` operador no Amazon DocumentDB é usado para determinar o tamanho de uma expressão de string em pontos de código (caracteres Unicode). Isso é útil quando você precisa saber o número de caracteres em uma string, em vez do número de bytes.

Parâmetros

- `expression`: a expressão de string para a qual retornar o comprimento em pontos de código.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$strLenCP` operador para determinar o comprimento das cadeias de caracteres com caracteres Unicode.

Crie documentos de amostra

```
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" }
])
```

Exemplo de consulta

```
db.people.aggregate([
  {
    $project: {
      "Desk": 1,
      "length": { $strLenCP: "$Desk" }
    }
  }
])
```

Saída

```
{ "_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 a diferença na medida do comprimento da string “Düsseldorf-BVV-021”, que contém um caractere Unicode (Ü). O `$strLenCP` operador conta corretamente o número de caracteres Unicode, enquanto o `$strLenBytes` operador conta o número de bytes.

Exemplos de código

Para ver um exemplo de código para usar o `$strLenCP` comando, escolha a guia do idioma que você deseja usar:

Node.js

Aqui está um exemplo de uso do `$strLenCP` operador em um Node.js aplicativo com o driver MongoDB:

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
```

```
  const db = client.db('test');
  const collection = db.collection('people');

  const result = await collection.aggregate([
    {
      $project: {
        "Desk": 1,
        "length": { $strLenCP: "$Desk" }
      }
    }
  ]).toArray();

  console.log(result);
  await client.close();
}

example();
```

Python

Aqui está um exemplo de uso do `$strLenCP` operador em um aplicativo Python com o PyMongo driver:

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.people

    result = list(collection.aggregate([
        {
            '$project': {
                "Desk": 1,
                "length": { "$strLenCP": "$Desk" }
            }
        }
    ]))

    print(result)
    client.close()
```

```
example()
```

$strcasecmp

O `$strcasecmp` operador no Amazon DocumentDB realiza uma comparação sem distinção entre maiúsculas e minúsculas entre duas cadeias de caracteres. Ele retorna um valor inteiro indicando a comparação lexicográfica das duas cadeias de caracteres de entrada, ignorando as diferenças entre maiúsculas e minúsculas.

Parâmetros

- `string1`: a primeira string a ser comparada.
- `string2`: a segunda string a ser comparada.

Exemplo (MongoDB Shell)

Este exemplo demonstra como usar o `$strcasecmp` operador para comparar cadeias de caracteres de localização de mesa em uma `people` coleção, ignorando as diferenças entre maiúsculas e minúsculas.

Crie documentos de amostra

```
db.people.insertMany([
  { "_id": 1, "Desk": "mke233-wi" },
  { "_id": 2, "Desk": "MKE233-WI" },
  { "_id": 3, "Desk": "mke233-wi" }
]);
```

Exemplo de consulta

```
db.people.aggregate([
  {
    $project: {
      item: 1,
      compare: { $strcasecmp: ["$Desk", "mke233-wi"] }
    }
  }
]);
```

Saída

```
{ "_id" : 1, "compare" : 0 }
{ "_id" : 2, "compare" : 0 }
{ "_id" : 3, "compare" : 0 }
```

A saída mostra que a comparação entre o "Desk" campo e a string "mke233-wi" retorna 0 para todos os três documentos, indicando que as strings são iguais quando as maiúsculas e minúsculas são ignoradas.

Exemplos de código

Para ver um exemplo de código para usar o $strcasecmp comando, escolha a guia do idioma que você deseja 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

O `$substr` operador no Amazon DocumentDB é usado para extrair uma substring de uma determinada string. É particularmente útil quando você precisa definir substrings com base em uma faixa de caracteres, em vez de uma faixa de bytes. Isso é especialmente importante ao lidar com cadeias de caracteres Unicode, nas quais o número de bytes usados para representar um caractere pode variar.

Parâmetros

- `string`: a string de entrada da qual extrair a substring.
- `start`: A posição inicial (com base em zero) da substring a ser extraída. Pode ser uma expressão inteira não negativa.

- `length`: o número de caracteres na substring extraída. Pode ser uma expressão inteira não negativa.

Exemplo (MongoDB Shell)

Neste exemplo, demonstraremos o uso de `$substr` para extrair a abreviatura estadual da mesa de um funcionário.

Crie documentos de amostra

```
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" }
]);
```

Exemplo de consulta

```
db.people.aggregate([
  {
    $project: {
      "state": { $substr: ["$Desk", 12, 3] }
    }
  }
])
```

Saída

```
{ "_id": 1, "state": "NRW" },
{ "_id": 2, "state": "HBB" },
{ "_id": 3, "state": "SHH" },
{ "_id": 4, "state": "BBB" }
```

Exemplos de código

Para ver um exemplo de código para usar o `$substr` comando, escolha a guia do idioma que você deseja 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

O `$substrBytes` operador no Amazon DocumentDB é usado para extrair uma substring de uma string com base em um intervalo de bytes especificado. Esse operador é útil quando você precisa extrair uma substring de uma string e o número de bytes necessários para representar cada caractere na string é importante.

Diferentemente`$substrCP`, que opera com base no número de pontos de código Unicode, `$substrBytes` opera no número de bytes necessários para representar os caracteres na string. Isso pode ser particularmente útil ao trabalhar com cadeias de caracteres que contêm caracteres não ASCII, pois esses caracteres podem exigir mais de um byte para serem representados.

*Nota: * está obsoleto `$substr` desde a versão 3.4. `$substr`agora é um alias para`$substrBytes`.

Parâmetros

- `string`: a string de entrada da qual a substring será extraída.
- `startByte`: A posição do byte inicial com base em zero da substring a ser extraída. Um valor negativo pode ser usado para especificar uma posição a partir do final da string.
- `length`: o número de bytes na substring a ser extraída.

Exemplo (MongoDB Shell)

Neste exemplo, usaremos `$substrBytes` para extrair uma substring de uma string que contém caracteres não ASCII.

Crie documentos de amostra

```
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" }
```

```
]);
```

Exemplo de consulta

```
db.people.aggregate([
  {
    $project: {
      "state": { $substrBytes: [ "$Desk", 12, 3] }
    }
  }
])
```

Saída

```
{ "_id": 1, "state": "NRW" },
{ "_id": 2, "state": "HBB" },
{ "_id": 3, "state": "SHH" },
{ "_id": 4, "state": "BBB" }
```

Neste exemplo, usamos `$substrBytes` para extrair uma substring de 3 bytes a partir do 12º byte do campo. `Desk` Isso nos permite extrair a abreviatura de estado de 2 caracteres, mesmo que a string possa conter caracteres não ASCII.

Exemplos de código

Para ver um exemplo de código para usar o `$substrBytes` comando, escolha a guia do idioma que você deseja 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

O `$substrCP` operador no Amazon DocumentDB é usado para extrair uma substring de uma string, onde a substring é especificada como um intervalo de pontos de UTF-8 código (CP). Esse operador é particularmente útil ao trabalhar com cadeias de caracteres Unicode, pois permite extrair substrings sem precisar se preocupar com a representação de bytes subjacente dos caracteres.

Ao contrário do `$substrBytes` operador, que opera em posições de bytes, o `$substrCP` operador trabalha com posições de pontos de código. Isso facilita o trabalho com cadeias de caracteres que contêm caracteres não ASCII, pois o número de pontos de código pode não corresponder ao número de bytes ou caracteres.

Parâmetros

- `string`: a string de entrada da qual extrair a substring.
- `start`: a posição inicial do ponto de código (com base em zero) da qual extrair a substring.
- `length`: o número de pontos de código a serem extraídos.

Exemplo (MongoDB Shell)

Neste exemplo, usaremos o `$substrCP` operador para extrair a abreviatura do estado de uma string contendo a localização da mesa do funcionário.

Crie documentos de amostra

```
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" }
]);
```

Exemplo de consulta

```
db.people.aggregate([
  {
    $project: {
      "state": { $substrCP: ["$Desk", 25, 2] }
    }
  }
]);
```

Saída

```
{ "_id" : 1, "state" : "MN" }
```

```
{ "_id" : 2, "state" : "LA" }
{ "_id" : 3, "state" : "CA" }
```

Neste exemplo, sabemos que a abreviatura do estado começa no 25º ponto de código no `Desk` campo e tem 2 pontos de código. Ao usar o `$substrCP` operador, podemos extrair a abreviatura do estado sem precisar nos preocupar com a representação de bytes subjacente da string.

Exemplos de código

Para ver um exemplo de código para usar o `$substrCP` comando, escolha a guia do idioma que você deseja 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()
```

Nos exemplos do Python Node.js e do Python, usamos o `$substrCP` operador para extrair a abreviatura do estado do `Desk` campo, semelhante ao exemplo do MongoDB Shell.

$subtract

O `$subtract` operador no Amazon DocumentDB é usado para subtrair valores. Ele pode ser usado para subtrair datas, números ou uma combinação de ambos. Esse operador é útil para calcular a diferença entre duas datas ou subtrair um valor de um número.

Parâmetros

- `expression1`: o primeiro valor a ser subtraído.
- `expression2`: o segundo valor a ser subtraído. `<expression1>`

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$subtract` operador para calcular a diferença entre duas datas.

Crie um documento de amostra

```
db.dates.insert([
  {
  "_id": 1,
  "startDate": ISODate("2023-01-01T00:00:00Z"),
  "endDate": ISODate("2023-01-05T12:00:00Z")
  }
```

```
]);
```

Exemplo de consulta

```
db.dates.aggregate([
  {
    $project: {
      _id: 1,
      durationDays: {
        $divide: [
          { $subtract: ["$endDate", "$startDate"] },
          1000 * 60 * 60 * 24  // milliseconds in a day
        ]
      }
    }
  }
]);
```

Saída

```
[ { _id: 1, durationDays: 4.5 } ]
```

Neste exemplo, o `$subtract` operador é usado para calcular a diferença entre `$endDate` e `$startDate` em dias.

Exemplos de código

Para ver um exemplo de código para usar o `$subtract` comando, escolha a guia do idioma que você deseja 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

O `$sum` operador no Amazon DocumentDB retorna a soma da expressão especificada para cada documento em um grupo. É um operador de acumulador de grupo que normalmente é usado no estágio $group de um pipeline de agregação para realizar cálculos de soma.

Parâmetros

- `expression`: A expressão numérica a ser somada. Isso pode ser um caminho de campo, uma expressão ou uma constante.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$sum` operador para calcular o total de vendas de cada produto.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.sales.aggregate([
  { $group: {
      _id: "$product",
      totalSales: { $sum: { $multiply: [ "$price", "$quantity" ] } }
    }}
]);
```

Saída

```
[
  { "_id": "abc", "totalSales": 50 },
  { "_id": "xyz", "totalSales": 120 }
]
```

Exemplo de uso do operador de janela (MongoDB Shell)

Novo na versão 8.0.2.

O `$sum` operador também pode ser usado como operador de janela no `$setWindowFields` palco. Nesse contexto, ele retorna a soma da expressão especificada para os documentos em cada janela. Você especifica o operador no `output` campo e, opcionalmente, define os limites da janela com um `window` documento.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

O exemplo a seguir particiona os documentos por`product`, classifica cada partição por `day` e retorna um total contínuo de `amount` desde o início da partição até o documento atual.

```
db.dailySales.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$product",
      sortBy: { day: 1 },
      output: {
        runningTotal: {
          $sum: "$amount",
          window: { documents: ["unbounded", "current"] }
        }
      }
    }
  }
]);
```

Saída

```
[
  { "_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 é aumentado com `runningTotal` a soma cumulativa de `amount` dentro de sua partição até e incluindo o documento atual.

Exemplos de código

Para ver um exemplo de código para usar o `$sum` operador, escolha a guia do idioma que você deseja usar. Os exemplos a seguir mostram o uso do acumulador (em`$group`) e o uso do operador de janela (em`$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

Novo a partir da versão 4.0.

Não é compatível com o cluster elástico.

O `$switch` operador é um operador de expressão condicional no Amazon DocumentDB que permite avaliar uma lista de expressões de caso e retornar o valor do primeiro caso avaliado como verdadeiro ou um valor padrão se nenhuma expressão de caso for verdadeira.

Parâmetros

- `branches`: uma matriz de documentos, cada um com um campo de caso que contém a expressão booleana a ser avaliada e um campo then que contém o valor a ser retornado se a expressão de maiúsculas e minúsculas for verdadeira.
- `default`: (opcional) O valor a ser retornado se nenhuma das expressões de maiúsculas e minúsculas for verdadeira.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$switch` operador para determinar o custo de envio de um pedido com base no total do pedido.

Crie documentos de amostra

```
db.orders.insertMany([
  { _id: 1, total: 50 },
  { _id: 2, total: 150 },
  { _id: 3, total: 250 }
]);
```

Exemplo 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
        }
      }
    }
  }
])
```

Saída

```
[
  {
    "_id": 1,
    "total": 50,
    "shippingCost": 5
  },
  {
    "_id": 2,
    "total": 150,
    "shippingCost": 10
  },
  {
    "_id": 3,
    "total": 250,
    "shippingCost": 15
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$switch` comando, escolha a guia do idioma que você deseja 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()
```

$ bronzeado

Novo na versão 8.0.1.

O `$tan` operador no Amazon DocumentDB retorna a tangente de um valor que é medido em radianos.

A expressão de entrada deve ser resolvida com um valor numérico. Se seu valor estiver em graus, aplique `$degreesToRadians` antes de passá-lo para`$tan`.

O tipo de retorno é `double` por padrão. Se a entrada for decimal de 128 bits, a saída também será decimal de 128 bits.

Parâmetros

- `expression`: uma expressão que se resolve em um número em radianos.

Comportamento

null, NaN e +/- Infinito

Exemplo	Resultados
`{ $tan: NaN }`	`NaN`
`{ $tan: null }`	`null`
`{ $tan: Infinity }` ou `{ $tan: -Infinity }`	Lança um erro.

Quando a entrada está `null` ou o campo referenciado está ausente, `$tan` retorna`null`. Uma entrada de `NaN` produtos`NaN`. O infinito positivo ou negativo causa um erro porque a tangente é indefinida nesses valores.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$tan` operador para calcular a tangente de um ângulo em radianos.

Crie documentos de amostra

```
db.angles.insertMany([
  { "_id": 1, "angle": 0 },
  { "_id": 2, "angle": 0.7853981633974483 },
  { "_id": 3, "angle": -0.7853981633974483 }
]);
```

Exemplo de consulta

```
db.angles.aggregate([
  { $project: {
    "tangent": { $tan: "$angle" }
  }}
]);
```

Saída

```
[
```

```
  { "_id": 1, "tangent": 0 },
  { "_id": 2, "tangent": 0.9999999999999999 },
  { "_id": 3, "tangent": -0.9999999999999999 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$tan` operador, escolha a guia do idioma que você deseja 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

Novo na versão 8.0.1.

O `$tanh` operador no Amazon DocumentDB retorna a tangente hiperbólica de um valor.

A expressão de entrada deve ser resolvida com um valor numérico.

O tipo de retorno é `double` por padrão. Se a entrada for decimal de 128 bits, a saída também será decimal de 128 bits.

Parâmetros

- `expression`: uma expressão que se resolve em um número.

Comportamento

null, NaN e +/- Infinito

Exemplo	Resultados
`{ $tanh: NaN }`	`NaN`
`{ $tanh: null }`	`null`
`{ $tanh: -Infinity }`	`-1`
`{ $tanh: Infinity }`	`1`

Quando a entrada está `null` ou o campo referenciado está ausente, `$tanh` retorna`null`. Uma entrada de `NaN` produtos`NaN`. Para infinito positivo, o resultado é`1`; para infinito negativo, o resultado é. `-1`

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$tanh` operador para calcular a tangente hiperbólica de um valor.

Crie documentos de amostra

```
db.values.insertMany([
  { "_id": 1, "value": 0 },
  { "_id": 2, "value": 1 },
  { "_id": 3, "value": -1 }
]);
```

Exemplo de consulta

```
db.values.aggregate([
  { $project: {
    "result": { $tanh: "$value" }
  }}
]);
```

Saída

```
[
  { "_id": 1, "result": 0 },
  { "_id": 2, "result": 0.7615941559557649 },
  { "_id": 3, "result": -0.7615941559557649 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$tanh` operador, escolha a guia do idioma que você deseja 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()
```

$top

Novo na versão 8.0.1.

Use o `$top` acumulador no `$group` estágio para retornar o documento com a classificação mais alta por grupo, de acordo com uma ordem de classificação especificada.

Parâmetros

- `sortBy`: um documento especificando a ordem de classificação. Use `1` para ascender ou `-1` descer.
- `output`: uma expressão que especifica os campos a serem retornados do documento superior.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$top` acumulador para encontrar a maior venda (maior quantidade) por item em uma coleção de vendas.

Crie documentos de amostra

```
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 }
])
```

Exemplo de consulta

```
db.sales.aggregate([
  { $group: { _id: "$item", topSale: { $top: { sortBy: { quantity: -1 }, output:
 { quantity: "$quantity", price: "$price" } } } } }
])
```

Saída

```
[
  { "_id": "xyz", "topSale": { "quantity": 15, "price": 3 } },
  { "_id": "abc", "topSale": { "quantity": 10, "price": 5 } }
]
```

Exemplo de uso do operador de janela (MongoDB Shell)

Novo na versão 8.0.2.

O `$top` operador também pode ser usado como operador de janela no `$setWindowFields` palco. Nesse contexto, ele retorna o `output` do documento mais bem classificado (de acordo com o próprio operador`sortBy`) entre os documentos em cada janela. Você especifica o operador no `output` campo e, opcionalmente, define os limites da janela com um `window` documento.

> **Note**
>
> Quando usado como operador de janela em`$setWindowFields`, `$top` é limitado a 100 MB de dados intermediários. Uma operação que excede esse limite retorna um erro.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

O exemplo a seguir particiona os documentos por`player`, classifica cada partição por `round` e retorna a rodada de maior pontuação vista desde o início da partição até o documento atual.

```
db.matchScores.aggregate([
  {
    $setWindowFields: {
      partitionBy: "$player",
      sortBy: { round: 1 },
      output: {
        bestRoundSoFar: {
          $top: { sortBy: { score: -1 }, output: { round: "$round", score:
 "$score" } },
          window: { documents: ["unbounded", "current"] }
        }
      }
```

```
    }
  }
]);
```

Saída

```
[
  { "_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 é aumentado com `bestRoundSoFar` a rodada de maior pontuação em sua partição, incluindo o documento atual.

Exemplos de código

Para ver um exemplo de código para usar o `$top` operador, escolha a guia do idioma que você deseja usar. Os exemplos a seguir mostram o uso do acumulador (em`$group`) e o uso do operador de janela (em`$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()
```

$ para PN

Novo na versão 8.0.1.

Use o `$topN` acumulador no `$group` estágio para retornar os N principais elementos dentro de um grupo de acordo com uma ordem de classificação especificada. Se um grupo contiver menos de N elementos, `$topN` retornará todos os elementos do grupo.

Parâmetros

- n: um número inteiro positivo ou uma expressão que se resolve em um, especificando quantos resultados principais devem ser retornados por grupo.
- `sortBy`: um documento especificando a ordem de classificação. Use 1 para ascender ou `-1` descer.

- `output`: uma expressão que especifica os campos a serem retornados de cada um dos N principais documentos.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$topN` acumulador para encontrar as duas principais vendas (maior quantidade) por item em uma coleção de vendas.

Crie documentos de amostra

```
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 }
])
```

Exemplo de consulta

```
db.sales.aggregate([
  { $group: { _id: "$item", topTwoSales: { $topN: { n: 2, sortBy: { quantity: -1 },
 output: { quantity: "$quantity", price: "$price" } } } } }
])
```

Saída

```
[
  { "_id": "xyz", "topTwoSales": [{ "quantity": 15, "price": 3 }, { "quantity": 9,
 "price": 6 }] },
  { "_id": "abc", "topTwoSales": [{ "quantity": 10, "price": 5 }, { "quantity": 7,
 "price": 8 }] }
]
```

Exemplo de uso do operador de janela (MongoDB Shell)

Novo na versão 8.0.2.

O `$topN` operador também pode ser usado como operador de janela no `$setWindowFields` palco. Nesse contexto, ele retorna uma matriz `output` dos N documentos superiores (de acordo com

o próprio operador`sortBy`) entre os documentos em cada janela. Você especifica o operador no `output` campo e, opcionalmente, define os limites da janela com um `window` documento.

> **Note**
>
> Quando usado como operador de janela em`$setWindowFields`, `$topN` é limitado a 100 MB de dados intermediários. Uma operação que excede esse limite retorna um erro.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

O exemplo a seguir particiona os documentos por`player`, classifica cada partição por `round` e retorna as duas rodadas de maior pontuação vistas desde o início da partição até o documento atual.

```
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"] }
        }
      }
    }
  }
]);
```

Saída

```
[
  { "_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 é aumentado com `topTwoRoundsSoFar` as duas rodadas de maior pontuação dentro de sua partição, incluindo o documento atual. Se a janela contiver menos de dois documentos, todos eles serão retornados.

Exemplos de código

Para ver um exemplo de código para usar o `$topN` operador, escolha a guia do idioma que você deseja usar. Os exemplos a seguir mostram o uso do acumulador (em`$group`) e o uso do operador de janela (em`$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

Novo na versão 8.0.1.

O `$trunc` operador no Amazon DocumentDB trunca um número em uma casa decimal especificada, removendo dígitos sem arredondar.

Parâmetros

- `number`: uma expressão que se resolve em um número.
- `place`: opcional. Uma expressão inteira entre -20 e 100 que especifica o número de casas decimais para truncar. Um valor negativo é truncado à esquerda do decimal. O padrão é 0.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$trunc` operador para truncar valores numéricos em uma casa decimal.

Truncando com um valor posicional positivo

```
db.measurements.insertMany([
  {_id: 1, value: 3.456},
  {_id: 2, value: 7.891},
  {_id: 3, value: 12.345}
]);
```

Exemplo de consulta

```
db.measurements.aggregate([
  { $project: { truncated: { $trunc: ["$value", 1] } } }
]);
```

Saída

```
[
  {_id: 1, truncated: 3.4},
  {_id: 2, truncated: 7.8},
  {_id: 3, truncated: 12.3}
]
```

Truncando com um valor posicional negativo

Quando `place` é negativo, `$trunc` substitui os dígitos à esquerda do decimal por zeros. O exemplo a seguir trunca valores usando o primeiro dígito à esquerda do 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] } } }
]);
```

Saída

```
[
  {_id: 1, truncated: 10},
  {_id: 2, truncated: 20},
  {_id: 3, truncated: 30}
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$trunc` operador, escolha a guia do idioma que você deseja 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('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

Novo na versão 8.0.1.

O `$tsIncrement` operador no Amazon DocumentDB retorna o ordinal incremental de um valor de timestamp como um inteiro longo.

Parâmetros

- `expression`: uma expressão que se transforma em um carimbo de data/hora.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$tsIncrement` operador para extrair o incremento ordinal dos valores do Timestamp.

Crie documentos de amostra

```
db.events.insertMany([
  {_id: 1, ts: Timestamp(1678900000, 1)},
  {_id: 2, ts: Timestamp(1678900000, 2)},
  {_id: 3, ts: Timestamp(1678900001, 1)}
]);
```

Exemplo de consulta

```
db.events.aggregate([
  { $project: { increment: { $tsIncrement: "$ts" } } }
]);
```

Saída

```
[
  {_id: 1, increment: Long("1")},
  {_id: 2, increment: Long("2")},
  {_id: 3, increment: Long("1")}
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$tsIncrement` operador, escolha a guia do idioma que você deseja 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()
```

$ tSecond

Novo na versão 8.0.1.

O `$tsSecond` operador no Amazon DocumentDB retorna a parte dos segundos de um valor de timestamp como um inteiro longo (os segundos da época do Unix).

Parâmetros

- `expression`: uma expressão que se transforma em um carimbo de data/hora.

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$tsSecond` operador para extrair a parte dos segundos dos valores do Timestamp.

Crie documentos de amostra

```
db.events.insertMany([
  {_id: 1, ts: Timestamp(1678900000, 1)},
  {_id: 2, ts: Timestamp(1678900000, 2)},
  {_id: 3, ts: Timestamp(1678900001, 1)}
]);
```

Exemplo de consulta

```
db.events.aggregate([
  { $project: { seconds: { $tsSecond: "$ts" } } }
]);
```

Saída

```
[
  {_id: 1, seconds: Long("1678900000")},
  {_id: 2, seconds: Long("1678900000")},
  {_id: 3, seconds: Long("1678900001")}
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$tsSecond` operador, escolha a guia do idioma que você deseja 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()
```

$toBool

O `$toBool` operador no Amazon DocumentDB converte uma expressão em um valor booleano.

Parâmetros

- `expression`: a expressão a ser convertida em um valor booleano.

Nota: Qualquer string é convertida em. `true`

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$toBool` operador para normalizar os valores de estado do dispositivo de diferentes tipos de dados.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.deviceStates.aggregate([
  {
    $project: {
      _id: 1,
```

```
      deviceId: 1,
      isActive: { $toBool: "$status" }
    }
  }
]);
```

Saída

```
[
  { "_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 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$toBool` comando, escolha a guia do idioma que você deseja 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

Novo da versão 4.0

O operador `$toDate` de agregação no Amazon DocumentDB é usado para converter uma string de data ou data e hora em um tipo de data BSON. Essa é a operação inversa do `$dateToString` operador.

Parâmetros

- `dateString`: uma representação em cadeia de caracteres de uma data ou data e hora a serem convertidas em um tipo de data BSON.
- `format`: (opcional) Uma string que especifica o formato do`dateString`. Se não for fornecido, o operador tentará analisá-los `dateString` em vários formatos padrão de data e hora.
- `timezone`: (opcional) Uma string representando o fuso horário a ser usado para a conversão. Se não for fornecido, o fuso horário local será usado.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$toDate` operador para converter uma string de data em um tipo de data BSON.

Crie documentos de amostra

```
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" }
]);
```

Exemplo de consulta

```
db.events.aggregate([
  {
    $project: {
      eventName: 1,
      eventTimeDate: { $toDate: "$eventTime" }
    }
  }
]);
```

Saída

```
[
  {
    "_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")
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$toDate` comando, escolha a guia do idioma que você deseja 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()
```

$toDecimal

Novo da versão 4.0

O `$toDecimal` operador no Amazon DocumentDB é usado para converter um valor no tipo de dados Decimal128. Isso é útil quando você precisa realizar aritmética decimal precisa ou lidar com valores decimais grandes que não podem ser representados com precisão usando o tipo de dados Double.

Parâmetros

- `expression`: a expressão a ser convertida para o tipo de dados Decimal128.

Exemplo (MongoDB Shell)

Este exemplo demonstra como usar o `$toDecimal` operador para converter um valor de string em um tipo de dados Decimal128.

Crie documentos de amostra

```
db.numbers.insertOne({ _id: 1, value: "3.14" });
db.numbers.insertOne({ _id: 2, value: "2.71" });
```

Exemplo de consulta

```
db.numbers.aggregate([
  { $project: {
    _id: 1,
    decimalValue: { $toDecimal: "$value" }
  }}
])
```

Saída

```
[
  { "_id" : 1, "decimalValue" : Decimal128("3.14") },
  { "_id" : 2, "decimalValue" : Decimal128("2.71") }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$toDecimal` comando, escolha a guia do idioma que você deseja 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()
```

$toDouble

Novo a partir da versão 4.0

O `$toDouble` operador no Amazon DocumentDB é usado para converter um valor em um número de ponto flutuante de 64 bits de precisão dupla. Isso pode ser útil quando você precisa realizar operações aritméticas em valores que não estão originalmente em um formato numérico.

Parâmetros

`<expression>`: a expressão a ser convertida em um valor duplo. Isso pode ser qualquer expressão válida que seja resolvida em um valor numérico, de string ou booleano.

Exemplo (MongoDB Shell)

Este exemplo demonstra como usar o `$toDouble` operador para converter um valor de string em um valor numérico com a finalidade de realizar um cálculo matemático.

Crie documentos de amostra

```
db.numbers.insertMany([
  { _id: 1, value: "10.5" },
  { _id: 2, value: "20.25" },
  { _id: 3, value: "7" }
])
```

Exemplo de consulta

```
db.numbers.aggregate([
  {
    $project: {
      _id: 1,
      value: 1,
      double_value: { $toDouble: "$value" },
      double_plus_five: { $add: [{ $toDouble: "$value" }, 5] }
    }
  }
])
```

Saída

```
[
  { "_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 }
```

```
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$toDouble` comando, escolha a guia do idioma que você deseja 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

Novo da versão 4.0

O `$toInt` operador no Amazon DocumentDB é usado para converter um valor de entrada em um tipo de dados inteiro. Esse operador é útil quando você precisa garantir que um campo ou expressão seja representado como um número inteiro, o que pode ser importante para determinadas operações ou tarefas de processamento de dados.

Parâmetros

- `expression`: a expressão a ser convertida em um número inteiro.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$toInt` operador para converter um valor de string em um número inteiro.

Crie documentos de amostra

```
db.numbers.insertMany([
  { "name": "one", "value": "1" },
```

```
  { "name": "hundred", "value": "100" }
]);
```

Exemplo de consulta

```
db.numbers.aggregate([
  { $project: {
    "_id": 0,
    "name": 1,
    "intValue": { $toInt: "$value" }
  }}
]);
```

Saída

```
{ "name": "one", "intValue": 1 }
{ "name": "hundred", "intValue": 100 }
```

Exemplos de código

Para ver um exemplo de código para usar o `$toInt` comando, escolha a guia do idioma que você deseja 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()
```

$toLong

Novo a partir da versão 4.0

O `$toLong` operador no Amazon DocumentDB é usado para converter um valor em um tipo de dados inteiro (longo) de 64 bits. Isso pode ser útil quando você precisa realizar operações aritméticas ou comparações em valores numéricos que podem ser armazenados como cadeias de caracteres ou outros tipos de dados.

Parâmetros

- `expression`: a expressão a ser convertida em um número inteiro de 64 bits.

Exemplo (MongoDB Shell)

Este exemplo demonstra como usar o `$toLong` operador para converter um valor de string em um inteiro de 64 bits.

Crie documentos de amostra

```
db.numbers.insertMany([
  { _id: 1, value: "42" },
  { _id: 3, value: "92233720368547758O7" }
]);
```

Exemplo de consulta

```
db.numbers.aggregate([
  {
    $project: {
      _id: 1,
      longValue: { $toLong: "$value" }
    }
  }
])
```

Saída

```
[
  { "_id" : 1, "longValue" : 42 },
  { "_id" : 3, "longValue" : 9223372036854775807 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$toLong` comando, escolha a guia do idioma que você deseja 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

O `$toLower` operador no Amazon DocumentDB é usado para converter uma string em minúsculas.

Parâmetros

- `expression`: a expressão de string a ser convertida em minúsculas.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$toLower` operador para converter o `Desk` campo em minúsculas.

Crie documentos de amostra

```
db.locations.insertMany([
  { "_id": 1, "Desk": "Düsseldorf-BVV-021" },
  { "_id": 2, "Desk": "Munich-HGG-32a" }
]);
```

Exemplo de consulta

```
db.locations.aggregate([
  { $project: { item: { $toLower: "$Desk" } } }
]);
```

Saída

```
{ "_id" : 1, "item" : "düsseldorf-bvv-021" }
{ "_id" : 2, "item" : "munich-hgg-32a" }
```

Exemplos de código

Para ver um exemplo de código para usar o `$toLower` comando, escolha a guia do idioma que você deseja 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()
```

$ para ObjectId

Novo da versão 4.0

O `$toObjectId` operador no Amazon DocumentDB é usado para converter uma representação em cadeia de caracteres de um em um ObjectId tipo de ObjectId dados real. Isso pode ser útil ao trabalhar com dados que foram armazenados como representações de seqüências de caracteres ObjectIds, pois permite realizar operações que exigem o tipo de ObjectId dados.

Parâmetros

- `expression`: uma expressão de string representando um válido ObjectId.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$toObjectId` operador para converter uma representação em cadeia de caracteres de an ObjectId para o tipo de ObjectId dados.

Crie documentos de amostra

```
db.employees.insertMany([
  { _id: 1, empId:"64e5f8886218c620cf0e8f8a", name: "Carol Smith", employeeId:
 "c720a" },
  { _id: 2, empId:"64e5f94e6218c620cf0e8f8c", name: "Bill Taylor", employeeId:
 "c721a" }
]);
```

Exemplo de consulta

```
db.employees.aggregate([
  { $project: {
    "empIdAsObjectId": {$toObjectId: "$empId"}}
  }
]);
```

Saída

```
[
  { _id: 1, empIdAsObjectId: ObjectId('64e5f8886218c620cf0e8f8a') },
  { _id: 2, empIdAsObjectId: ObjectId('64e5f94e6218c620cf0e8f8c') }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$toObjectId` comando, escolha a guia do idioma que você deseja 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

Novo da versão 4.0

O `$toString` operador no Amazon DocumentDB é usado para converter um valor de qualquer tipo (exceto null) em uma representação de string. Isso pode ser útil quando você precisa realizar operações de string em valores que não estão originalmente no formato de string.

Parâmetros

- `expression`: a expressão a ser convertida em uma string.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$toString` operador para converter valores numéricos em cadeias de caracteres.

Crie documentos de amostra

```
db.numbers.insertMany([
  { "_id": 1, "value": 42 },
  { "_id": 2, "value": 3.14 }
]);
```

Exemplo de consulta

```
db.numbers.aggregate([
  { $project: {
    _id: 1,
    valueAsString: { $toString: "$value" }
  }}
]);
```

Saída

```
{ "_id": 1, "valueAsString": "42" }
{ "_id": 2, "valueAsString": "3.14" }
```

Exemplos de código

Para ver um exemplo de código para usar o `$toString` comando, escolha a guia do idioma que você deseja 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

O `$toUpper` operador no Amazon DocumentDB é usado para converter uma string em maiúsculas.

Parâmetros

- `expression`: a expressão de seqüência de caracteres a ser convertida em maiúsculas.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$toUpper` operador para converter o `Desk` campo em maiúsculas.

Crie documentos de amostra

```
db.locations.insertMany([
  { "_id": 1, "Desk": "düsseldorf-bvv-021" },
  { "_id": 2, "Desk": "munich-hgg-32a" }
]);
```

Exemplo de consulta

```
db.locations.aggregate([
  { $project: { item: { $toUpper: "$Desk" } } }
]);
```

Saída

```
{ "_id" : 1, "item" : "DüSSELDORF-BVV-021" }
{ "_id" : 2, "item" : "MUNICH-HGG-32A" }
```

Exemplos de código

Para ver um exemplo de código para usar o `$toUpper` comando, escolha a guia do idioma que você deseja 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()
```

$ para UUID

Novo na versão 8.0.1.

O `$toUUID` operador no Amazon DocumentDB converte um valor de string em um UUID (subtipo binário 4).

Parâmetros

- `expression`: uma expressão que resolve para uma string no formato UUID (por exemplo, “12345678-1234-1234-1234-123456789abc”).

Exemplo (MongoDB Shell)

O exemplo a seguir mostra como usar o `$toUUID` operador para converter valores de string no formato UUID.

Crie documentos de amostra

```
db.records.insertMany([
  {_id: 1, uuidStr: "550e8400-e29b-41d4-a716-446655440000"},
  {_id: 2, uuidStr: "6ba7b810-9dad-11d1-80b4-00c04fd430c8"}
]);
```

Exemplo de consulta

```
db.records.aggregate([
  { $project: { uuid: { $toUUID: "$uuidStr" } } }
]);
```

Saída

```
[
  {_id: 1, uuid: UUID("550e8400-e29b-41d4-a716-446655440000")},
  {_id: 2, uuid: UUID("6ba7b810-9dad-11d1-80b4-00c04fd430c8")}
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$toUUID` operador, escolha a guia do idioma que você deseja 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

Novo da versão 4.0

O `$trim` operador no Amazon DocumentDB é usado para remover os caracteres iniciais de espaço em branco à and/or direita de uma string.

Parâmetros

- `input`: A expressão de string a ser cortada.
- `chars`: (opcional) Especifica os caracteres a serem cortados do início e do final da entrada, o padrão é espaço em branco.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$trim` operador para remover espaços em branco à esquerda e à direita de uma string.

Crie documentos de amostra

```
db.people.insertMany([
  { "name": "   John Doe   " },
  { "name": "   Bob Johnson   " }
])
```

Exemplo de consulta

```
db.people.aggregate([
  { $project: {
```

```
    "name": { $trim: {input: "$name"}}
  }}
])
```

Saída

```
[
  { "name": "John Doe" },
  { "name": "Bob Johnson" }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$trim` comando, escolha a guia do idioma que você deseja 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

O operador `$type` de agregação retorna o tipo de dados BSON de um campo especificado. Isso é útil para identificar o tipo de dados dos valores de campo durante as operações de agregação.

Parâmetros

- `expression`: o campo ou expressão cujo tipo deve ser retornado.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$type` operador para identificar o tipo de dados do campo de preço de cada produto.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.inventory.aggregate([
  {
    $project: {
      item: 1,
      price: 1,
      priceType: { $type: "$price" }
    }
  }
]);
```

Saída

```
[
  { _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' }
]
```

Exemplos de código

Para ver um exemplo de código para usar o operador de `$type` agregação, escolha a guia do idioma que você deseja 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

Novo a partir da versão 8.0

Não é compatível com o cluster elástico.

O estágio `$unset` de agregação no Amazon DocumentDB permite que você remova campos dos documentos.

Parâmetros

- `expression`: nome do campo ou lista de vários nomes de campo.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do estágio de `$unset` agregação para remover o `price` campo.

Crie documentos de amostra

```
db.inventory.insertMany([
  { item: "pencil", quantity: 100, price: 0.24},
  { item: "pen", quantity: 204, price: 1.78 }
]);
```

Exemplo de agregação

```
db.inventory.aggregate([
  {
    $unset: "price"
  }
])
```

Saída

```
[
  {
    _id: ObjectId('69248951d66dcae121d2950d'),
    item: 'pencil',
    quantity: 100
  },
  {
    _id: ObjectId('69248951d66dcae121d2950e'),
    item: 'pen',
    quantity: 204
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$unset` comando, escolha a guia do idioma que você deseja 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

O `$unwind` operador é usado para desconstruir um campo de matriz dos documentos de entrada para gerar um documento para cada elemento. Isso pode ser útil quando você deseja realizar operações nos elementos individuais de uma matriz, como filtrar, classificar ou transformar os dados.

Parâmetros

- `path`: o caminho para o campo de matriz a ser desenrolado.
- `includeArrayIndex`: (opcional) Especifica o nome do novo campo para conter o índice do elemento da matriz.
- `preserveNullAndEmptyArrays`: (opcional) Determina se a operação mantém o documento original quando o campo de matriz é nulo ou uma matriz vazia.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$unwind` operador para desconstruir um campo de matriz e realizar outras operações nos elementos individuais.

Crie documentos de amostra

```
db.people.insertMany([
  { _id: 1, name: "jon", hobbies: ["painting", "dancing", "singing"] },
  { _id: 2, name: "jane", hobbies: ["reading", "swimming"] },
  { _id: 3, name: "jack", hobbies: [] }
])
```

Exemplo de consulta

```
db.people.aggregate([
  { $unwind: "$hobbies" }
```

```
])
```

Saída

```
[
  { _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' }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$unwind` comando, escolha a guia do idioma que você deseja 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

Novo da versão 8.0

Não é compatível com o cluster elástico.

O `$vectorSearch` operador no Amazon DocumentDB permite que você realize a pesquisa vetorial, um método usado no aprendizado de máquina para encontrar pontos de dados semelhantes comparando suas representações vetoriais usando métricas de distância ou similaridade. Esse recurso combina a flexibilidade e a rica consulta de um banco de dados de JSON-based documentos com o poder da pesquisa vetorial, permitindo que você crie casos de uso de aprendizado de máquina e IA generativa, como pesquisa semântica, recomendação de produtos e muito mais.

Parâmetros

- `<exact>`(opcional): Sinalizador que especifica se a pesquisa Exact Nearest Neighbor (ENN) ou Approximate Nearest Neighbor (ANN) deve ser executada. O valor pode ser um dos seguintes:
- false - para executar uma pesquisa ANN
- true - para executar a pesquisa ENN

Se omitido ou definido como falso, `numCandidates` é obrigatório.

```
- `<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.
```

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$vectorSearch` operador para encontrar descrições de produtos semelhantes com base em suas representações vetoriais.

Crie documentos de amostra

```
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]
  }
]);
```

Criar índice de pesquisa vetorial

```
db.runCommand(
    {
        createIndexes: "products",
        indexes: [{
            key: {
                "description_vector": "vector"
```

```
            },
            vectorOptions: {
                type: "hnsw",
                dimensions: 3,
                similarity: "cosine",
                m: 16,
                efConstruction: 64
            },
            name: "description_index"
        }]
    }
);
```

Exemplo de consulta

```
db.products.aggregate([
  { $vectorSearch: {
      index: "description_index",
      limit: 2,
      numCandidates: 10,
      path: "description_vector",
      queryVector: [0.1, 0.2, 0.3]
    }
  }
]);
```

Saída

```
[
  {
    "_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 ]
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$vectorSearch` comando, escolha a guia do idioma que você deseja 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

O `$week` operador no Amazon DocumentDB retorna o número da semana de uma data (0-53) com base no padrão ISO 8601. O número da semana é calculado com base no ano e no dia da semana, com segunda-feira como o primeiro dia da semana.

Parâmetros

Nenhum

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$week` operador para recuperar o número da semana de uma determinada data.

Crie documentos de amostra

```
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") }
]);
```

Exemplo de consulta

```
db.events.aggregate([
  { $project: {
    _id: 1,
    week: { $week: "$date" }
  }}
]);
```

Saída

```
[
  { "_id": 1, "week": 1 },
  { "_id": 2, "week": 2 },
  { "_id": 3, "week": 53 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$week` comando, escolha a guia do idioma que você deseja 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

O `$year` operador no Amazon DocumentDB extrai o componente ano de uma data ou timestamp.

Parâmetros

- `expression`: a expressão de data ou carimbo de data/hora da qual extrair o componente do ano.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$year` operador para extrair o componente ano de um campo de data.

Crie documentos de amostra

```
db.events.insertMany([
  { "_id": 1, "date": ISODate("2023-04-15T00:00:00Z") },
  { "_id": 3, "date": ISODate("2021-12-31T00:00:00Z") }
]);
```

Exemplo de consulta

```
db.events.aggregate([
  { $project: { year: { $year: "$date" } } }
]);
```

Saída

```
[
  { "_id": 1, "year": 2023 },
  { "_id": 3, "year": 2021 }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$year` comando, escolha a guia do idioma que você deseja 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

O `$zip` operador no Amazon DocumentDB permite combinar várias matrizes em uma única matriz de tuplas (pares de valores-chave). Isso é útil quando você precisa criar novos documentos ou objetos combinando dados de diferentes fontes ou matrizes em um documento.

Parâmetros

- `inputs`: uma matriz de expressões que se resolvem em matrizes. Essas matrizes serão combinadas em uma única matriz de tuplas.
- `useLongestLength`: (opcional) `Setrue`, a matriz de saída tiver o comprimento da matriz de entrada mais longa, preenchendo matrizes mais curtas com valores. `null` `Sefalse`, a matriz de saída terá o comprimento da matriz de entrada mais curta.
- `defaults`: (opcional) Uma matriz de valores padrão a serem usados para as tuplas se a matriz de entrada correspondente for menor que a matriz de entrada mais longa e `useLongestLength` for. `true`

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$zip` operador para combinar duas matrizes em uma única matriz de tuplas.

Crie documentos de amostra

```
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"]
  }
])
```

Exemplo de consulta

```
db.grades.aggregate([
  {
    $project: {
      "name": 1,
      "scoredClasses": {
        $zip: {
          inputs: ["$scores", "$classes"],
          useLongestLength: true,
          defaults: [null, null]
        }
      }
    }
  }
])
```

Saída

```
[
  {
```

```
    "_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"]
    ]
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$zip` comando, escolha a guia do idioma que você deseja 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()
```

Geoespacial

Esta seção fornece informações detalhadas sobre operadores geoespaciais suportados pelo Amazon DocumentDB.

Tópicos

- $geometry
- $geoIntersects
- $geoWithin
- $maxDistance
- $minDistance
- $near
- $nearSphere

$geometry

O `$geometry` operador no Amazon DocumentDB é usado para especificar um objeto geométrico GeoJSON como parte de uma consulta geoespacial. Esse operador é usado em conjunto com outros operadores de consulta geoespacial, como `$geoWithin` e `$geoIntersects` para realizar consultas espaciais em seus dados.

No Amazon DocumentDB, o `$geometry` operador oferece suporte aos seguintes tipos de geometria GeoJSON:

- Point
- LineString
- Polígono
- MultiPoint
- MultiLineString
- MultiPolygon

- GeometryCollection

Parâmetros

- `type`: O tipo do objeto de geometria GeoJSON, por exemplo,,, etc. `Point` `Polygon`
- `coordinates`: uma matriz de coordenadas representando a geometria. A estrutura da matriz de coordenadas depende do tipo de geometria.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$geometry` operador para realizar uma `$geoIntersects` consulta no Amazon DocumentDB.

Crie documentos de amostra

```
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]
      ]]
    }
  }
]);
```

Exemplo 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]
        ]]
      }
    }
  }
})
```

Saída

```
[
  {
    "_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]
        ]
      ]
    }
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$geometry` comando, escolha a guia do idioma que você deseja 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

O `$geoIntersects` operador no Amazon DocumentDB é usado para encontrar documentos cujos dados geoespaciais se cruzam com um objeto GeoJSON especificado. Esse operador é útil para aplicativos que exigem a identificação de documentos com base em sua relação espacial com uma determinada forma geográfica, como um polígono ou multipolígono.

Parâmetros

- `$geometry`: Um objeto GeoJSON que representa a forma para verificar a interseção. Os tipos de objeto GeoJSON compatíveis `Point` são`LineString`,, e. `Polygon` `MultiPolygon`

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$geoIntersects` operador para encontrar o nome do estado de um determinado conjunto de coordenadas no Amazon DocumentDB.

Crie documentos de amostra

```
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]
      ]]
    }
  }
]);
```

Exemplo de consulta

```
var location = [-73.965355, 40.782865];

db.states.find({
  "loc": {
    "$geoIntersects": {
```

```
      "$geometry": {
        "type": "Point",
        "coordinates": location
      }
    }
  }
}, {
  "name": 1
});
```

Saída

```
{ "_id" : ObjectId("536b0a143004b15885c91a2c"), "name" : "New York" }
```

Exemplos de código

Para ver um exemplo de código para usar o `$geoIntersects` comando, escolha a guia do idioma que você deseja 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

O `$geoWithin` operador no Amazon DocumentDB é usado para encontrar documentos cujos dados de localização (representados como objetos GeoJSON) estão completamente contidos em uma forma especificada, como um polígono ou multipolígono. Isso é útil para consultar objetos localizados em uma região geográfica específica.

Parâmetros

- `$geometry`: um objeto GeoJSON que representa a forma a ser consultada.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$geoWithin` operador para encontrar todos os aeroportos localizados no estado de Nova York.

Crie documentos de amostra

```
// 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]
        }
    }
]);
```

Exemplo 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
});
```

Saída

```
[
  {
    "name": "John F. Kennedy International Airport",
    "type": "airport",
    "code": "JFK"
  },
  {
    "name": "LaGuardia Airport",
    "type": "airport",
    "code": "LGA"
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$geoWithin` comando, escolha a guia do idioma que você deseja 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

O `$maxDistance` operador no Amazon DocumentDB é usado para especificar a distância máxima (em metros) de um ponto GeoJSON em que os documentos devem estar para serem incluídos nos resultados da consulta. Esse operador é usado em conjunto com o `$nearSphere` operador para realizar consultas geoespaciais.

Parâmetros

- `$maxDistance`: a distância máxima (em metros) do ponto de referência em que os documentos devem estar para serem incluídos nos resultados da consulta.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$maxDistance` operador no Amazon DocumentDB para encontrar todas as capitais estaduais em um raio de 100 quilômetros de Boston.

Crie documentos de amostra

```
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] } }
]);
```

Exemplo de consulta

```
db.capitals.find(
  {
    location: {
      $nearSphere: {
        $geometry: { type: "Point", coordinates: [-71.0589, 42.3601] },
        $maxDistance: 100000
      }
    }
  },
```

```
  { state: 1, city: 1, _id: 0 }
);
```

Saída

```
[
  { "state": "Rhode Island", "city": "Providence" },
  { "state": "New Hampshire", "city": "Concord" }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$maxDistance` comando, escolha a guia do idioma que você deseja 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

`$minDistance`é um operador de busca usado em conjunto com `$nearSphere` ou `$geoNear` para filtrar documentos que estejam pelo menos na distância mínima especificada do ponto central. Esse operador é suportado no Amazon DocumentDB e funciona de forma semelhante à sua contraparte no MongoDB.

Parâmetros

- `$minDistance`: A distância mínima (em metros) do ponto central para incluir documentos nos resultados.

Exemplo (MongoDB Shell)

Neste exemplo, encontraremos todos os restaurantes em um raio de 2 quilômetros de um local específico em Seattle, Washington.

Crie documentos de amostra

```
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]
    }
  }
]);
```

Exemplo de consulta

```
db.usarestaurants.find({
```

```
  "location": {
    "$nearSphere": {
      "$geometry": {
        "type": "Point",
        "coordinates": [-122.3516, 47.6156]
      },
      "$minDistance": 1,
      "$maxDistance": 2000
    }
  }
}, {
  "name": 1
});
```

Saída

```
{ "_id" : ObjectId("611f3da985009a81ad38e74b"), "name" : "Noodle House" }
{ "_id" : ObjectId("611f3da985009a81ad38e74c"), "name" : "Pike Place Grill" }
```

Exemplos de código

Para ver um exemplo de código para usar o `$minDistance` comando, escolha a guia do idioma que você deseja 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

O `$near` operador no Amazon DocumentDB é usado para encontrar documentos que estão geograficamente próximos a um ponto especificado. Ele retorna documentos ordenados por distância, com os documentos mais próximos primeiro. Esse operador requer um índice geoespacial 2dsphere e é útil para consultas de proximidade em dados de localização.

Parâmetros

- `$geometry`: Um objeto GeoJSON Point que define o ponto central para a consulta próxima.
- `$maxDistance`: (opcional) A distância máxima em metros do ponto especificado que um documento pode estar para corresponder à consulta.
- `$minDistance`: (opcional) A distância mínima em metros do ponto especificado que um documento pode estar para corresponder à consulta.

Requisitos de índice

- `2dsphere index`: Obrigatório para consultas geoespaciais em dados do GeoJSON Point.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$near` operador para encontrar os restaurantes mais próximos de um local específico em Seattle, Washington.

Crie documentos de amostra

```
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] }
  }
]);
```

Crie um índice 2dsphere

```
db.usarestaurants.createIndex({ "location": "2dsphere" });
```

Exemplo de consulta com GeoJSON Point

```
db.usarestaurants.find({
  location: {
    $near: {
      $geometry: {
        type: "Point",
        coordinates: [-122.3516, 47.6156]
      },
      $maxDistance: 100,
      $minDistance: 10
    }
  }
});
```

Saída

```
{
  "_id" : ObjectId("69031ec9ea1c2922a1ce5f4a"),
  "name" : "Noodle House",
  "city" : "Seattle",
  "state" : "Washington",
  "rating" : 4.8,
  "location" : {
    "type" : "Point",
    "coordinates" : [ -122.3517, 47.6159 ]
  }
}
```

Exemplos de código

Para ver um exemplo de código para usar o `$near` comando, escolha a guia do idioma que você deseja 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

O `$nearSphere` operador no Amazon DocumentDB é usado para encontrar documentos que estão dentro de uma distância especificada de um ponto geoespacial. Esse operador é particularmente útil para consultas geoespaciais, como encontrar todos os restaurantes dentro de um determinado raio de um determinado local.

Parâmetros

- `$geometry`: Um objeto GeoJSON que representa o ponto de referência. Deve ser um `Point` objeto com `coordinates` campos `type` e.

- `$minDistance`: (opcional) A distância mínima (em metros) do ponto de referência que os documentos devem estar.
- `$maxDistance`: (opcional) A distância máxima (em metros) do ponto de referência que os documentos devem estar.

Exemplo (MongoDB Shell)

Neste exemplo, encontraremos todos os restaurantes a 2 quilômetros (2000 metros) de um local específico em Seattle, WA.

Crie documentos de amostra

```
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] }
  }
]);
```

Exemplo de consulta

```
db.usarestaurants.find({
  location: {
    $nearSphere: {
      $geometry: {
        type: "Point",
        coordinates: [-122.3516, 47.6156]
      },
      $minDistance: 1,
      $maxDistance: 2000
    }
  }
}, {
```

```
  name: 1
});
```

Saída

```
{ "_id" : ObjectId("611f3da985009a81ad38e74b"), "name" : "Noodle House" }
{ "_id" : ObjectId("611f3da985009a81ad38e74c"), "name" : "Pike Place Grill" }
```

Exemplos de código

Para ver um exemplo de código para usar o `$nearSphere` comando, escolha a guia do idioma que você deseja 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 e projeção

Esta seção fornece informações detalhadas sobre operadores de consulta e projeção suportados pelo Amazon DocumentDB.

Tópicos

- $

- $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

$

O operador $ de projeção limita o conteúdo de um campo de matriz para retornar somente o primeiro elemento que corresponda à condição da consulta. Ele é usado para projetar um único elemento de matriz correspondente.

Parâmetros

- `field.$`: o campo de matriz com o operador posicional para projetar o primeiro elemento correspondente.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do operador de $ projeção para retornar somente o elemento de matriz correspondente.

Crie documentos de amostra

```
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] }
]);
```

Exemplo de consulta

```
db.students.find(
  { grades: { $gte: 90 } },
  { name: 1, "grades.$": 1 }
);
```

Saída

```
{ "_id" : 1, "name" : "Alice", "grades" : [ 92 ] }
{ "_id" : 2, "name" : "Bob", "grades" : [ 92 ] }
{ "_id" : 3, "name" : "Charlie", "grades" : [ 95 ] }
```

Neste exemplo, somente a primeira nota maior ou igual a 90 é retornada para cada aluno.

Exemplos de código

Para ver um exemplo de código para usar o operador de $ projeção, escolha a guia do idioma que você deseja 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

O `$all` operador no Amazon DocumentDB é usado para combinar documentos em que o valor de um campo é uma matriz e contém todos os elementos especificados, independentemente da ordem dos elementos na matriz.

Parâmetros

- `field`: O nome do campo a ser verificado.
- `[value1, value2, ...]`: A lista de valores a serem correspondidos na matriz.

Usando **`$elemMatch`** dentro de uma **`$all`** expressão

Consulte Usando `$elemMatch` em uma `expressão $all` as limitações relacionadas ao uso do `$elemMatch` operador em uma `$all` expressão.

Dólar ($) em nomes de campo

Consulte as limitações Dólar ($) e ponto (.) nos nomes dos campos relacionadas à consulta de campos $ prefixados `$all` em objetos aninhados.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$all` operador para recuperar documentos em que o campo “Cores” é uma matriz que contém “Vermelho” e “Azul”.

Crie documentos de amostra

```
db.example.insertMany([
  { "Item": "Pen", "Colors": ["Red", "Blue", "Green"] },
  { "Item": "Notebook", "Colors": ["Blue", "White"] },
  { "Item": "Poster Paint", "Colors": ["Red", "Yellow", "White"] }
```

```
])
```

Exemplo de consulta

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

Saída

```
{
  "_id" : ObjectId("6137d6c5b3a1d35e0b6ee6ad"),
  "Item" : "Pen",
  "Colors" : [
          "Red",
          "Blue",
          "Green"
  ]
}
```

Exemplos de código

Para ver um exemplo de código para usar o `$all` comando, escolha a guia do idioma que você deseja 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

O `$and` operador no Amazon DocumentDB é usado para combinar várias expressões e avaliá-las como uma única condição. Ele retorna `true` se todas as expressões fornecidas forem avaliadas como`true`, e de `false` outra forma. Esse operador é útil para aplicar vários critérios a uma consulta.

Parâmetros

- `expression1`: uma expressão obrigatória que é avaliada como um valor booleano.
- `expression2`: uma expressão obrigatória que é avaliada como um valor booleano.
- `...`: expressões adicionais necessárias que são avaliadas em valores booleanos.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$and` operador para localizar todos os documentos na coleção “usuários” em que o campo “idade” é maior que 18 e o campo “status” é “ativo”.

Crie documentos de amostra

```
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" }
]);
```

Exemplo de consulta

```
db.users.find({
  $and: [
    { age: { $gt: 18 } },
    { status: "active" }
  ]
});
```

Saída

```
[
  { "_id" : ObjectId("614e3c4b63f5892e7c4e2345"), "name" : "John", "age" : 25,
 "status" : "active" },
  { "_id" : ObjectId("614e3c4b63f5892e7c4e2347"), "name" : "Alice", "age" : 22,
 "status" : "active" }
]
```

Exemplos de código

Para ver um exemplo de código para usar o $and comando, escolha a guia do idioma que você deseja 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

O `$bitsAllClear` operador no Amazon DocumentDB é usado para corresponder aos documentos em que todos os bits especificados em um campo são apagados (definidos como 0). Esse operador pode ser útil para realizar operações bit a bit em dados armazenados.

Parâmetros

- `field`: o campo para verificar se os bits especificados estão sendo apagados.

- `value`: a máscara de bits numérica que especifica quais bits devem ser verificados ou uma lista de posições de bits a serem verificadas. Uma máscara de bits numérica pode ser binária (0b...), decimal, hexadecimal (0x...), octal (0o...) ou binária (). BinData Em uma lista de posições de bits, a posição do bit menos significativo é 0.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$bitsAllClear` operador no Amazon DocumentDB.

Crie documentos de amostra

```
db.collection.insertMany([
  { _id: 1, bits: 0b1010 },
  { _id: 2, bits: 0b1100 },
  { _id: 3, bits: 0b0101 }
]);
```

Exemplo de consulta

```
db.collection.find({
  bits: { $bitsAllClear: 0b0011 }
})
```

Saída

```
{ "_id" : 2, "bits" : 12 }
```

A consulta verifica se todos os bits especificados pela máscara de bits `0b0011` (os dois bits menos significativos) estão apagados no `bits` campo. O documento com `_id` 2 satisfaz essa condição, pois seu `bits` campo tem esses bits limpos.

Exemplos de código

Para ver um exemplo de código para usar o `$bitsAllClear` comando, escolha a guia do idioma que você deseja 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

O `$bitsAllSet` operador no Amazon DocumentDB é usado para consultar documentos em que um conjunto específico de bits está todo definido como 1 em um campo. Esse operador permite que

você execute operações bit a bit em valores de campo e pode ser útil quando você precisa verificar o estado de bits individuais em um valor numérico.

Parâmetros

- `field`: o nome do campo numérico no qual realizar a operação bit a bit.
- `value`: a máscara de bits numérica que especifica quais bits devem ser verificados ou uma lista de posições de bits a serem verificadas. Uma máscara de bits numérica pode ser binária (0b...), decimal, hexadecimal (0x...), octal (0o...) ou binária (). BinData Em uma lista de posições de bits, a posição do bit menos significativo é 0.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$bitsAllSet` operador para encontrar documentos em que o `flags` campo tenha todos os bits definidos pela máscara de bits.

Crie documentos de amostra

```
db.collection.insert([
  { _id: 1, flags: 0b1010 },
  { _id: 2, flags: 0b1100 },
  { _id: 3, flags: 0b1110 }
])
```

Exemplo de consulta

```
db.collection.find({ flags: { $bitsAllSet: 0b1100 } })
```

Saída

```
{ "_id": 2, "flags": 12 },
{ "_id": 3, "flags": 14 }
```

Neste exemplo, a consulta verifica documentos em que o `flags` campo tem todos os bits definidos pela máscara de bits `0b1100` (que representa o valor decimal 12). Os documentos com `_id` 2 e 3 correspondem a esse critério, pois seus valores de `flags` campo têm todos os bits necessários definidos (o 3º e o 4º bits menos significativos).

Exemplos de código

Para ver um exemplo de código para usar o `$bitsAllSet` comando, escolha a guia do idioma que você deseja 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

O `$bitsAnyClear` operador no Amazon DocumentDB é usado para combinar os documentos em que qualquer um dos bits especificados em um campo é apagado (definido como 0). Isso pode ser útil para realizar operações bit a bit em valores de campo armazenados em documentos.

Parâmetros

- `field`: O campo a ser verificado.
- `value`: a máscara de bits numérica que especifica quais bits devem ser verificados ou uma lista de posições de bits a serem verificadas. Uma máscara de bits numérica pode ser binária (0b...), decimal, hexadecimal (0x...), octal (0o...) ou binária (). BinData Em uma lista de posições de bits, a posição do bit menos significativo é 0.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$bitsAnyClear` operador para verificar se algum bit está limpo no `status` campo da `items` coleção.

Crie documentos de amostra

```
db.items.insertMany([
  { "_id": 1, "status": 7 },
  { "_id": 2, "status": 15 },
  { "_id": 3, "status": 31 }
]);
```

Exemplo de consulta

```
db.items.find({ "status": { $bitsAnyClear: 8 } })
```

Saída

```
{ "_id" : 1, "status" : 7 }
```

Neste exemplo, a consulta verifica documentos em que o `status` campo tem algum bit claro (0) na máscara de bits 8 (binário`1000`). O documento com `status` valores de 7 (binário`111`) corresponde à consulta, pois tem pelo menos um bit claro na máscara de bits fornecida. O bit claro correspondente é o quarto bit menos significativo.

Exemplos de código

Para ver um exemplo de código para usar o `$bitsAnyClear` comando, escolha a guia do idioma que você deseja 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

O `$bitsAnySet` operador no Amazon DocumentDB é usado para consultar documentos em que pelo menos um bit é definido como 1 nos bits especificados em um campo. Esse operador permite que você execute operações bit a bit nos valores armazenados nos campos, permitindo consultas e análises eficientes de dados com características bit a bit.

Parâmetros

- `field`: o nome do campo ao qual aplicar a operação bit a bit.
- `value`: a máscara de bits numérica que especifica quais bits devem ser verificados ou uma lista de posições de bits a serem verificadas. Uma máscara de bits numérica pode ser binária (0b...), decimal, hexadecimal (0x...), octal (0o...) ou binária (). BinData Em uma lista de posições de bits, a posição do bit menos significativo é 0.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$bitsAnySet` operador para encontrar documentos nos quais pelo menos um bit está definido no `flags` campo.

Crie documentos de amostra

```
db.collection.insertMany([
  { _id: 1, flags: 0b1010 },
  { _id: 2, flags: 0b1100 },
  { _id: 3, flags: 0b0011 },
  { _id: 4, flags: 0b0100 }
]);
```

Exemplo de consulta

```
db.collection.find({
  flags: { $bitsAnySet: 0b1010 }
})
```

Saída

```
{ "_id" : 1, "flags" : 10 }
{ "_id" : 2, "flags" : 12 }
```

```
{ "_id" : 3, "flags" : 3 }
```

A consulta retorna os documentos em que pelo menos um dos bits especificados na máscara de bits `0b1010` está definido no `flags` campo.

Exemplos de código

Para ver um exemplo de código para usar o `$bitsAnySet` comando, escolha a guia do idioma que você deseja 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

O `$comment` operador no Amazon DocumentDB é usado para anexar um comentário a uma consulta. Isso pode ser útil para fornecer contexto ou informações adicionais sobre a consulta, o que pode ser útil para fins de depuração ou documentação. O comentário em anexo aparecerá como parte da saída de operações como db.currentOp ().

Parâmetros

- `string`: o comentário anexado à consulta.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$comment` operador no Amazon DocumentDB.

Crie documentos de amostra

```
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" }
]);
```

Exemplo de consulta

```
db.users.find({ age: { $gt: 25 } }, { _id: 0, name: 1, age: 1 }).comment("Retrieve
 users older than 25");
```

Saída

```
{ "name" : "John Doe", "age" : 30 }
{ "name" : "Bob Johnson", "age" : 35 }
```

Exemplos de código

Para ver um exemplo de código para usar o `$comment` comando, escolha a guia do idioma que você deseja 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

O `$elemMatch` operador no Amazon DocumentDB é usado para consultar um campo de matriz e retornar documentos em que pelo menos um elemento na matriz corresponda aos critérios especificados. Esse operador é particularmente útil quando você tem estruturas de dados complexas com matrizes aninhadas ou documentos incorporados.

A versão 2.0 do Planner adicionou suporte de índice para`$elemMatch`.

Parâmetros

- `field`: o campo de matriz a ser consultado.
- `query`: os critérios para comparar com os elementos da matriz.

Usando **`$elemMatch`** dentro de uma **`$all`** expressão

Consulte Usando `$elemMatch` em uma `expressão $all` as limitações relacionadas ao uso do `$elemMatch` operador em uma `$all` expressão.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$elemMatch` operador para encontrar documentos em que a `parts` matriz tenha pelo menos um elemento que corresponda aos critérios especificados.

Crie documentos de amostra

```
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 }] }
```

```
]);
```

Exemplo de consulta

```
db.col.find({
  parts: { "$elemMatch": { part: "xyz", qty: { $lt: 11 } } }
})
```

Saída

```
{ "_id" : 1, "parts" : [ { "part" : "xyz", "qty" : 10 }, { "part" : "abc", "qty" :
 20 } ] }
{ "_id" : 2, "parts" : [ { "part" : "xyz", "qty" : 5 }, { "part" : "abc", "qty" :
 10 } ] }
```

Exemplos de código

Para ver um exemplo de código para usar o `$elemMatch` comando, escolha a guia do idioma que você deseja 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

O `$eq` operador no Amazon DocumentDB é usado para combinar documentos em que o valor de um campo é igual ao valor especificado. Esse operador é comumente usado no `find()` método para recuperar documentos que atendem aos critérios especificados.

Parâmetros

- `field`: o campo para verificar a condição de igualdade.
- `value`: o valor a ser comparado com o campo.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$eq` operador para encontrar todos os documentos em que o `name` campo é igual a. `"Thai Curry Palace"`

Crie documentos de amostra

```
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"] }
]);
```

Exemplo de consulta

```
db.restaurants.find({ name: { $eq: "Thai Curry Palace" } });
```

Saída

```
{ "_id" : ObjectId("68ee586f916df9d39f3d9414"), "name" : "Thai Curry Palace",
 "cuisine" : "Thai", "features" : [ "Private Dining" ] }
```

Exemplos de código

Para ver um exemplo de código para usar o `$eq` comando, escolha a guia do idioma que você deseja 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

O `$exists` operador é usado para verificar se um campo existe ou não em um documento. `$exists`particularmente útil ao trabalhar com índices esparsos no Amazon DocumentDB, onde o campo indexado pode não estar presente em todos os documentos.

Para usar um índice esparso que você criou em uma consulta, é necessário usar a cláusula `$exists` nos campos que abrangem o índice. Se você omitir`$exists`, o Amazon DocumentDB não usará o índice esparso para a consulta.

Parâmetros

- `field`: o nome do campo para verificar a existência.
- `value`: um valor booleano (`true`ou`false`) que especifica se o campo deve existir (`true`) ou não existir (`false`) nos documentos correspondentes.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$exists` operador com um índice esparso no `special_diets` campo da coleção. `food`

Crie documentos de amostra

```
db.food.insertMany([
  { _id: 1, name: "Apple", special_diets: ["vegetarian", "gluten-free"] },
  { _id: 2, name: "Broccoli" },
```

```
  { _id: 3, name: "Chicken", special_diets: ["dairy-free"] }
]);
```

Crie um índice esparso no campo `special_diets`

```
db.food.createIndex({ "special_diets": 1 }, { sparse: true, name:
 "special_diets_sparse_asc" });
```

Exemplo de consulta

```
db.food.find({ "special_diets": { $exists: true } });
```

Saída

```
[
  { "_id" : 1, "name" : "Apple", "special_diets" : [ "vegetarian", "gluten-free" ] },
  { "_id" : 3, "name" : "Chicken", "special_diets" : [ "dairy-free" ] }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$exists` comando, escolha a guia do idioma que você deseja 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

Novo na versão 4.0.

Não é suportado pelo cluster elástico.

O `$expr` operador no Amazon DocumentDB permite que você use expressões de agregação dentro da linguagem de consulta. Ele permite que você realize comparações e cálculos complexos nos campos de um documento, da mesma forma que você usaria os estágios do pipeline de agregação.

Os operadores `$eq`,`$lt`,`$lte`,`$gt`, e de `$gte` comparação colocados em um `$expr` operador podem usar um índice, incluindo um índice na `from` coleção referenciada em um `$lookup` estágio.

Limitações

- Os índices só podem ser usados para comparações entre campos e constantes, portanto, o operando deve ser resolvido como uma constante. Por exemplo, uma comparação entre `$a` e um valor constante pode usar um índice, mas uma comparação entre `$a` e `$b` não.
- Índices multichave, parciais ou esparsos não são usados. Nesses casos, a consulta volta para uma varredura de coleção.
- Quando o `let` operando é resolvido para um valor vazio ou ausente, a comparação não pode usar uma busca de índice direcionada. Nesse caso, o Amazon DocumentDB executa uma verificação completa do índice.

Parâmetros

- `expression`: uma expressão que retorna um valor booleano, permitindo que você realize comparações e cálculos nos campos do documento.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$expr` operador para localizar todos os documentos em que o `manufacturingCost` campo é maior que o `price` campo.

Crie documentos de amostra

```
db.inventory.insertMany([
  { item: "abc", manufacturingCost: 500, price: 100 },
  { item: "def", manufacturingCost: 300, price: 450 },
  { item: "ghi", manufacturingCost: 400, price: 120 }
]);
```

Exemplo de consulta

```
db.inventory.find({
  $expr: {
    $gt: ["$manufacturingCost", "$price"]
  }
})
```

Saída

```
{ "_id" : ObjectId("60b9d4d68d2cac581bc5a89a"), "item" : "abc", "manufacturingCost" :
 500, "price" : 100 },
{ "_id" : ObjectId("60b9d4d68d2cac581bc5a89c"), "item" : "ghi", "manufacturingCost" :
 400, "price" : 120 }
```

Exemplos de código

Para ver um exemplo de código para usar o `$expr` comando, escolha a guia do idioma que você deseja 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

O `$gt` operador no Amazon DocumentDB é usado para selecionar documentos em que o valor do campo especificado é maior que o valor especificado. Esse operador é útil para filtrar e consultar dados com base em comparações numéricas.

Parâmetros

- `field`: O campo a ser comparado.
- `value`: O valor a ser comparado.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$gt` operador para encontrar todos os documentos em que o `age` campo é maior que 30.

Crie documentos de amostra

```
db.users.insertMany([
  { name: "John", age: 25 },
  { name: "Jane", age: 32 },
  { name: "Bob", age: 45 },
  { name: "Alice", age: 28 }
]);
```

Exemplo de consulta

```
db.users.find({ age: { $gt: 30 } });
```

Saída

```
{ "_id" : ObjectId("6249e5c22a5d39884a0a0001"), "name" : "Jane", "age" : 32 },
{ "_id" : ObjectId("6249e5c22a5d39884a0a0002"), "name" : "Bob", "age" : 45 }
```

Exemplos de código

Para ver um exemplo de código para usar o `$gt` comando, escolha a guia do idioma que você deseja 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

O `$gte` operador no Amazon DocumentDB é usado para combinar valores maiores ou iguais a um valor especificado. Esse operador é útil para filtrar e consultar dados com base em comparações numéricas.

Parâmetros

- `field`: o campo a ser verificado em relação ao valor fornecido.
- `value`: o valor a ser comparado com o campo.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$gte` operador no Amazon DocumentDB para encontrar todos os documentos em que o campo “idade” é maior ou igual a 25.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.users.find({ age: { $gte: 25 } }, { _id: 0, name: 1, age: 1 });
```

Saída

```
{ "name" : "Jane", "age" : 25 }
{ "name" : "Bob", "age" : 30 }
{ "name" : "Alice", "age" : 35 }
```

Exemplos de código

Para ver um exemplo de código para usar o `$gte` comando, escolha a guia do idioma que você deseja 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

O `$in` operador no Amazon DocumentDB é um operador de consulta lógica que permite encontrar documentos em que o valor de um campo é igual a qualquer um dos valores especificados em uma matriz.

Parâmetros

- `field`: O campo a ser verificado em relação à matriz fornecida.
- `[value1, value2, ...]`: uma matriz de valores a serem comparados com o campo especificado.

Dólar (**$**) em nomes de campo

Consulte as limitações Dólar ($) e ponto (.) nos nomes dos campos relacionadas à consulta de campos $ prefixados `$in` em objetos aninhados.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$in` operador para localizar documentos em que o `color` campo é um dos valores na matriz fornecida.

Crie documentos de amostra

```
db.colors.insertMany([
  { "_id": 1, "color": "red" },
  { "_id": 2, "color": "green" },
  { "_id": 3, "color": "blue" },
  { "_id": 4, "color": "yellow" },
  { "_id": 5, "color": "purple" }
])
```

Exemplo de consulta

```
db.colors.find({ "color": { "$in": ["red", "blue", "purple"] } })
```

Saída

```
{ "_id": 1, "color": "red" },
{ "_id": 3, "color": "blue" },
{ "_id": 5, "color": "purple" }
```

Exemplos de código

Para ver um exemplo de código para usar o `$in` comando, escolha a guia do idioma que você deseja 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

Novo a partir da versão 4.0.

Não é compatível com o cluster elástico.

O `$jsonSchema` operador no Amazon DocumentDB é usado para filtrar documentos com base em um esquema JSON especificado. Esse operador permite que você consulte documentos que correspondam a um esquema JSON específico, garantindo que os documentos recuperados atendam aos requisitos estruturais e de tipo de dados específicos.

Usando o operador de consulta de `$jsonSchema` avaliação como parte da criação de uma coleção, você pode validar o esquema dos documentos que estão sendo inseridos na coleção. Para obter informações adicionais, consulte Usando a validação do esquema JSON.

Parâmetros

- `required`(matriz): especifica os campos obrigatórios no documento.
- `properties`(objeto): define o tipo de dados e outras restrições para cada campo no documento.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$jsonSchema` operador para filtrar a `employees` coleção para recuperar somente documentos que tenham os `age` camposname, `employeeId` e, e o `employeeId` campo seja do tipo. `string`

Crie documentos de amostra

```
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" }
]);
```

Exemplo de consulta

```
db.employees.aggregate([
  { $match: {
    $jsonSchema: {
      required: ["name", "employeeId", "age"],
      properties: { "employeeId": { "bsonType": "string" } }
    }
  }}
]);
```

Saída

```
{ "_id" : ObjectId("6908e8b61f77fc26b2ecd26f"), "name" : { "firstName" : "Emily",
 "lastName" : "Brown" }, "employeeId" : "2", "age" : 25 }
```

Exemplos de código

Para ver um exemplo de código para usar o `$jsonSchema` comando, escolha a guia do idioma que você deseja 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

O `$lt` operador no Amazon DocumentDB é usado para combinar valores menores que o valor especificado. Esse operador é útil para filtrar e consultar dados com base em comparações numéricas.

Parâmetros

- `field`: O nome do campo a ser verificado.
- `value`: O valor a ser comparado.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$lt` operador para recuperar documentos em que o `Inventory.0nHand` valor seja menor que 50.

Crie documentos de amostra

```
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"] }
]);
```

Exemplo de consulta

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

Saída

```
{ "_id" : 1, "Inventory" : { "OnHand" : 25, "Sold" : 60 }, "Colors" : [ "Red",
 "Blue" ] }
{ "_id" : 3, "Inventory" : { "OnHand" : 10, "Sold" : 85 }, "Colors" : [ "Red",
 "Green" ] }
```

Exemplos de código

Para ver um exemplo de código para usar o `$lt` comando, escolha a guia do idioma que você deseja 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

O operador de $mod consulta seleciona documentos em que um valor de campo dividido por um divisor tem um restante especificado. Isso é útil para filtrar documentos com base nas condições aritméticas do módulo.

Parâmetros

- `divisor`: O número pelo qual dividir.
- `remainder`: O valor restante esperado.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do $mod operador para encontrar todos os pedidos em que a quantidade é um número ímpar.

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.orders.find({ quantity: { $mod: [2, 1] } });
```

Saída

```
{ "_id" : 1, "item" : "Widget", "quantity" : 15 }
{ "_id" : 3, "item" : "Tool", "quantity" : 7 }
{ "_id" : 5, "item" : "Part", "quantity" : 9 }
```

Essa consulta retorna documentos em que a quantidade dividida por 2 tem o restante de 1, selecionando efetivamente todas as quantidades ímpares.

Exemplos de código

Para ver um exemplo de código para usar o operador de $mod consulta, escolha a guia do idioma que você deseja 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

O `$lte` operador no Amazon DocumentDB é usado para combinar documentos em que o valor de um campo especificado é menor ou igual ao valor especificado. Esse operador é útil para filtrar e consultar dados com base em comparações numéricas.

Parâmetros

- `field`: O campo a ser comparado.
- `value`: O valor a ser comparado.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$lte` operador para recuperar documentos em que o `quantity` campo é menor ou igual a 10.

Crie documentos de amostra

```
db.inventory.insertMany([
  { item: "canvas", qty: 100 },
  { item: "paint", qty: 50 },
  { item: "brush", qty: 10 },
```

```
  { item: "paper", qty: 5 }
]);
```

Exemplo de consulta

```
db.inventory.find({ qty: { $lte: 10 } });
```

Saída

```
{ "_id" : ObjectId("..."), "item" : "brush", "qty" : 10 },
{ "_id" : ObjectId("..."), "item" : "paper", "qty" : 5 }
```

Exemplos de código

Para ver um exemplo de código para usar o `$lte` comando, escolha a guia do idioma que você deseja 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

O `$meta` operador é usado para acessar os metadados associados à execução da consulta atual. Esse operador é usado principalmente para operações de pesquisa de texto, nas quais os metadados podem fornecer informações sobre a relevância dos documentos correspondentes.

Parâmetros

- `textScore`: recupera a pontuação da pesquisa de texto do documento. Essa pontuação indica a relevância do documento para a consulta de pesquisa de texto.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$meta` operador para recuperar a pontuação da pesquisa de texto para documentos que correspondam a uma consulta de pesquisa de texto.

Crie documentos de amostra

```
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." }
]);
```

Criar índice de texto

```
db.documents.createIndex({ content: "text" });
```

Exemplo de consulta

```
db.documents.find(
  { $text: { $search: "coffee" } },
  { _id: 0, title: 1, content: 1, score: { $meta: "textScore" } }
).sort({ score: { $meta: "textScore" } });
```

Saída

```
[
  {
    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
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$meta` comando, escolha a guia do idioma que você deseja 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

O $ne operador é usado para combinar documentos em que o valor de um campo não é igual ao valor especificado. É um operador de comparação que pode ser usado em predicados de consulta para filtrar documentos.

A versão 2.0 do Planner adicionou suporte de índice para$ne.

Parâmetros

- `field`: O campo a ser verificado.
- `value`: O valor a ser comparado.

Exemplo (MongoDB Shell)

Neste exemplo, encontraremos todos os documentos na `users` coleção em que o `status` campo não é igual `"active"` a.

Crie documentos de amostra

```
db.users.insertMany([
  { name: "John", status: "active" },
  { name: "Jane", status: "inactive" },
  { name: "Bob", status: "suspended" },
  { name: "Alice", status: "active" }
]);
```

Exemplo de consulta

```
db.users.find({ status: { $ne: "active" } });
```

Saída

```
[
  {
    _id: ObjectId('...'),
    name: 'Jane',
    status: 'inactive'
  },
  {
    _id: ObjectId('...'),
    name: 'Bob',
    status: 'suspended'
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o $ne comando, escolha a guia do idioma que você deseja 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

O `$nin` operador é usado para combinar valores que não estão na matriz especificada. É o inverso do `$in` operador, que corresponde aos valores que estão na matriz especificada.

A versão 2.0 do Planner adicionou suporte de índice para`$nin`.

Parâmetros

- `field`: O campo a ser verificado.
- `array`: a matriz de valores a serem comparados.

Dólar (**$**) em nomes de campo

Consulte as limitações Dólar ($) e ponto (.) nos nomes dos campos relacionadas à consulta de campos $ prefixados `$nin` em objetos aninhados.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$nin` operador para encontrar documentos em que o `category` campo não é igual a “Ficção” ou “Mistério”.

Crie documentos de amostra

```
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" }
])
```

Exemplo de consulta

```
db.books.find({
  category: {
    $nin: ["Fiction", "Mystery"]
  }
```

```
})
```

Saída

```
[
  {
    _id: ObjectId('...'),
    title: 'The Martian',
    author: 'Andy Weir',
    category: 'Science Fiction'
  },
  {
    _id: ObjectId('...'),
    title: 'The Alchemist',
    author: 'Paulo Coelho',
    category: 'Philosophy'
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$nin` comando, escolha a guia do idioma que você deseja 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

O `$nor` operador é usado para combinar documentos em que nenhuma das condições de consulta especificadas é verdadeira. É semelhante à operação lógica “NOR”, em que o resultado é verdadeiro se nenhum dos operandos for verdadeiro.

Parâmetros

- `expression1`: A primeira expressão a ser avaliada.
- `expression2`: A segunda expressão a ser avaliada.
- `expressionN`: Expressões adicionais para avaliar.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$nor` operador recuperando documentos em que o `qty` campo não é menor que 20 e o `size` campo não é igual a “XL”.

Crie documentos de amostra

```
db.items.insertMany([
  { qty: 10, size: "M" },
  { qty: 15, size: "XL" },
  { qty: 25, size: "L" },
  { qty: 30, size: "XL" }
])
```

Exemplo de consulta

```
db.items.find({
  $nor: [
    { qty: { $lt: 20 } },
    { size: "XL" }
  ]
})
```

Saída

```
[
  { "_id" : ObjectId("..."), "qty" : 25, "size" : "L" }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$nor` comando, escolha a guia do idioma que você deseja 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

O `$not` operador é usado para negar o resultado de uma determinada expressão. Ele permite que você selecione documentos nos quais a condição especificada não corresponda.

A versão 2.0 do Planner adicionou suporte de índice para `$not {eq}` e. `$not {in}`

Parâmetros

- `expression`: A expressão a ser negada.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$not` operador para encontrar documentos em que o `status` campo não é igual a “ativo”.

Crie documentos de amostra

```
db.users.insertMany([
  { name: "John", status: "active" },
  { name: "Jane", status: "inactive" },
  { name: "Bob", status: "pending" },
  { name: "Alice", status: "active" }
]);
```

Exemplo de consulta

```
db.users.find({ status: { $not: { $eq: "active" } } });
```

Saída

```
[
  {
    _id: ObjectId('...'),
    name: 'Jane',
    status: 'inactive'
  },
  {
    _id: ObjectId('...'),
    name: 'Bob',
    status: 'pending'
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$not` comando, escolha a guia do idioma que você deseja 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

O `$or` operador é usado para realizar uma operação lógica OR em uma matriz de duas ou mais expressões. Ele retorna documentos que correspondem a pelo menos uma das expressões. Esse operador é útil quando você precisa consultar documentos que satisfaçam qualquer uma das várias condições.

Parâmetros

- `expression1`: A primeira expressão a ser avaliada.
- `expression2`: A segunda expressão a ser avaliada.
- `...`: Expressões adicionais para avaliar (opcional).

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$or` operador para localizar documentos em que `make` seja "TruckForYou" com o modelo “Heavy H1" ou" SportForYou "com o modelo “Bolid 1".

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.cars.find({
  $or: [
    { make: "TruckForYou", model: "Heavy H1" },
    { make: "SportForYou", model: "Bolid 1" }
  ]
});
```

Saída

```
[
```

```
  {
    _id: ObjectId('...'),
    make: 'TruckForYou',
    model: 'Heavy H1',
    year: 2020
  },
  {
    _id: ObjectId('...'),
    make: 'SportForYou',
    model: 'Bolid 1',
    year: 2021
  }
]
```

Exemplos de código

Para ver um exemplo de código para usar o `$or` comando, escolha a guia do idioma que você deseja 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

O `$regex` operador permite que você realize a correspondência de expressões regulares em campos de string. É uma ferramenta poderosa para pesquisar e filtrar documentos com base em padrões complexos.

Parâmetros

- `regular expression`: o padrão de expressão regular correspondente ao campo.
- `$options`: (opcional) Fornece opções para modificar o comportamento da pesquisa, como distinção entre maiúsculas e minúsculas, correspondência global etc.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$regex` operador para pesquisar documentos em que o campo “nome” corresponda a um padrão específico.

Crie documentos de amostra

```
db.users.insertMany([
  { name: "John Doe" },
  { name: "Jane Smith" },
  { name: "Alice Johnson" },
  { name: "Bob Williams" },
  { name: "Charlie Davis" }
]);
```

Exemplo de consulta

```
db.users.find({ name: { $regex: /^A/ } })
```

Saída

```
[
  { "_id" : ObjectId("..."), "name" : "Alice Johnson" }
]
```

Essa consulta retornará todos os documentos em que o campo “nome” comece com a letra “A”.

Exemplos de código

Para ver um exemplo de código para usar o `$regex` comando, escolha a guia do idioma que você deseja 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

Novo na versão 8.0.1.

Não é suportado pelo cluster elástico.

O `$sampleRate` operador no Amazon DocumentDB combina uma amostra aleatória dos documentos de entrada com base na taxa especificada. Você pode usá-lo em um filtro de `find()` consulta ou em um `$match` estágio de agregação. Como a seleção é probabilística, o número de documentos devolvidos é aproximado e pode variar entre as execuções.

Parâmetros

- `value`: um número no intervalo `[0, 1]` (inclusive), passado diretamente para o operador (por exemplo,`{ $sampleRate: 0.3 }`), que define a probabilidade de incluir cada documento. Um valor de `0` retorna nenhum documento; um valor de 1 retorna todos os documentos.

Exemplo (MongoDB Shell)

O exemplo a seguir é usado `$sampleRate` para devolver aproximadamente 30% dos documentos em uma coleção.

Crie documentos de amostra

```
db.events.insertMany([
  { _id: 1, type: "click" },
  { _id: 2, type: "view" },
  { _id: 3, type: "click" },
  { _id: 4, type: "view" },
  { _id: 5, type: "purchase" }
]);
```

Exemplo de consulta

```
db.events.aggregate([
  { $match: { $sampleRate: 0.3 } }
]);
```

Saída

A operação retorna um subconjunto aleatório dos documentos. Por ser `$sampleRate` probabilístico, os documentos específicos retornados e sua contagem variam sempre que a consulta é executada.

Exemplos de código

Para ver um exemplo de código para usar o `$sampleRate` comando, escolha a guia do idioma que você deseja 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('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

O operador `$slice` de projeção limita o número de elementos da matriz retornados em um resultado de consulta. Ele permite que você recupere um número específico de elementos do início ou do final de um campo de matriz sem carregar a matriz inteira.

Parâmetros

- `field`: O campo de matriz a ser projetado.
- `count`: Número de elementos a serem retornados. Valores positivos retornam elementos do início, valores negativos do final.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o operador de `$slice` projeção para retornar somente os dois primeiros itens de um campo de matriz.

Crie documentos de amostra

```
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"] }
]);
```

Exemplo de consulta

```
db.inventory.find(
  {},
  { item: 1, tags: { $slice: 2 } }
)
```

Saída

```
{ "_id" : 1, "item" : "notebook", "tags" : [ "office", "school" ] }
{ "_id" : 2, "item" : "pen", "tags" : [ "office", "writing" ] }
{ "_id" : 3, "item" : "folder", "tags" : [ "office", "supplies" ] }
```

Exemplos de código

Para ver um exemplo de código para usar o operador de `$slice` projeção, escolha a guia do idioma que você deseja 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

O operador de `$size` consulta corresponde a documentos em que um campo de matriz tem exatamente o número especificado de elementos. Isso é útil para filtrar documentos com base no tamanho da matriz.

Parâmetros

- `field`: O campo de matriz a ser verificado.
- `count`: o número exato de elementos que a matriz deve conter.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$size` operador para encontrar todos os produtos que têm exatamente três etiquetas.

Crie documentos de amostra

```
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"] }
]);
```

Exemplo de consulta

```
db.products.find({ tags: { $size: 3 } });
```

Saída

```
{ "_id" : 1, "name" : "Laptop", "tags" : [ "electronics", "computers", "portable" ] }
{ "_id" : 3, "name" : "Desk", "tags" : [ "furniture", "office", "workspace" ] }
```

Exemplos de código

Para ver um exemplo de código para usar o operador de `$size` consulta, escolha a guia do idioma que você deseja 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

O `$text` operador é usado para realizar a pesquisa de texto completo em campos indexados por texto em uma coleção de documentos. Esse operador permite pesquisar documentos que contenham palavras ou frases específicas e pode ser combinado com outros operadores de consulta para filtrar os resultados com base em critérios adicionais.

Parâmetros

- `$search`: A string de texto a ser pesquisada.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$text` operador para pesquisar documentos contendo a palavra “interesse” e filtrar os resultados com base no campo “star_rating”.

Crie documentos de amostra

```
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" }
]);
```

Criar índice de texto

```
db.test.createIndex({ comments: "text" });
```

Exemplo de consulta

```
db.test.find({$and: [{star_rating: 5}, {$text: {$search: "interest"}}]})
```

Saída

```
{ "_id" : 5, "star_rating" : 5, "comments" : "interesting couch" }
{ "_id" : 6, "star_rating" : 5, "comments" : "interested in couch for sale, year
 2022" }
```

O comando acima retorna documentos com um campo indexado por texto contendo qualquer forma de “interesse” e uma “star_rating” igual a 5.

Exemplos de código

Para ver um exemplo de código para usar o `$text` comando, escolha a guia do idioma que você deseja 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

O `$type` operador é usado para verificar o tipo de dados de um campo em um documento. Ele pode ser usado quando operações ou validações específicas do tipo são necessárias. O `$type` operador retorna o tipo BSON da expressão avaliada. O tipo retornado é uma string, que corresponde ao tipo do campo ou da expressão.

A versão 2.0 do Planner adicionou suporte de índice para`$type`.

Parâmetros

- `expression`: A expressão a ser avaliada.

Exemplo (MongoDB Shell)

Crie documentos de amostra

```
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 }
]);
```

Exemplo de consulta

```
db.documents.find({
  $or: [
    { age: { $type: "number" } },
    { email: { $type: "string" } },
```

```
    { name: { $type: "string" } }
  ]
})
```

Saída

```
[
  { "_id": 1, "name": "John", "age": 30, "email": "john@example.com" },
  { "_id": 2, "name": "Jane", "age": "25", "email": 123456 }
]
```

Essa consulta retornará os documentos em que o `age` campo é do tipo “número”, o `email` campo é do tipo “string” e o `name` campo é do tipo “string”.

Exemplos de código

Para ver um exemplo de código para usar o `$type` comando, escolha a guia do idioma que você deseja 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 atualização

Esta seção fornece informações detalhadas sobre operadores de atualização compatíveis com o Amazon DocumentDB.

Tópicos

- $
- $[]
- $ [<identificador>]
- $adicionar ToSet
- $bit
- $currentDate
- $each

- $inc
- $max
- $min
- $mul
- $pop
- $position
- $pull
- $pullAll
- $push
- $rename
- $set
- $ set OnInsert
- $slice
- $sort
- $unset

$

O operador $ posicional atualiza o primeiro elemento da matriz que corresponde à condição de consulta. Ele atua como um espaço reservado para a posição do elemento de matriz correspondente.

Parâmetros

- `field.$`: o campo de matriz com o operador posicional para atualizar o primeiro elemento correspondente.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do operador $ posicional para atualizar um elemento específico da matriz.

Crie documentos de amostra

```
db.inventory.insertMany([
```

```
  { _id: 1, item: "Widget", quantities: [10, 20, 30] },
  { _id: 2, item: "Gadget", quantities: [5, 15, 25] }
]);
```

Exemplo de consulta

```
db.inventory.updateOne(
  { _id: 1, quantities: 20 },
  { $set: { "quantities.$": 22 } }
);
```

Saída

```
{
  "_id" : 1,
  "item" : "Widget",
  "quantities" : [ 10, 22, 30 ]
}
```

Exemplos de código

Para ver um exemplo de código para usar o operador $ posicional, escolha a guia do idioma que você deseja 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()
```

$[]

O operador `$[]` all posicional atualiza todos os elementos em uma matriz. Ele é usado quando você precisa modificar cada elemento em um campo de matriz.

Parâmetros

- `field.$[]`: o campo de matriz com o operador posicional all para atualizar todos os elementos.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$[]` operador para atualizar todos os elementos da matriz.

Crie documentos de amostra

```
db.products.insertOne({
  _id: 1,
  name: "Laptop",
  prices: [1000, 1100, 1200]
});
```

Exemplo de consulta

```
db.products.updateOne(
  { _id: 1 },
  { $inc: { "prices.$[]": 50 } }
);
```

Saída

```
{
  "_id" : 1,
  "name" : "Laptop",
  "prices" : [ 1050, 1150, 1250 ]
}
```

Exemplos de código

Para ver um exemplo de código para usar o `$[]` operador, escolha a guia do idioma que você deseja 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>]

O operador posicional `$[<identifier>]` filtrado atualiza todos os elementos da matriz que correspondem às condições de filtro especificadas. Ele é usado com a `arrayFilters` opção de atualizar seletivamente os elementos da matriz.

Parâmetros

- `field.$[identifier]`: o campo de matriz com o operador posicional filtrado.

- `arrayFilters`: uma matriz de condições de filtro que determinam quais elementos atualizar.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$[<identifier>]` operador para atualizar elementos específicos da matriz com base em uma condição.

Crie documentos de amostra

```
db.students.insertOne({
  _id: 1,
  name: "Alice",
  grades: [
    { subject: "Math", score: 85 },
    { subject: "Science", score: 92 },
    { subject: "History", score: 78 }
  ]
});
```

Exemplo de consulta

```
db.students.updateOne(
  { _id: 1 },
  { $inc: { "grades.$[elem].score": 5 } },
  { arrayFilters: [{ "elem.score": { $gte: 80 } }] }
);
```

Saída

```
{
  "_id" : 1,
  "name" : "Alice",
  "grades" : [
    { "subject" : "Math", "score" : 90 },
    { "subject" : "Science", "score" : 97 },
    { "subject" : "History", "score" : 78 }
  ]
}
```

Exemplos de código

Para ver um exemplo de código para usar o `$[<identifier>]` operador, escolha a guia do idioma que você deseja 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()
```

$adicionar ToSet

O `$addToSet` operador no Amazon DocumentDB é usado para adicionar um valor a uma matriz somente se o valor ainda não estiver presente na matriz. Isso é útil para garantir que uma matriz contenha elementos exclusivos.

Parâmetros

- `field`: O campo a ser atualizado.
- `value`: o valor a ser adicionado ao campo da matriz. Isso pode ser um valor único ou uma expressão.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$addToSet` operador para adicionar elementos exclusivos a uma matriz.

Crie documentos de amostra

```
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"] }
])
```

Exemplo de consulta

```
db.products.update(
  { "item": "apple" },
```

```
  { $addToSet: { "tags": "green" } }
)
```

Saída

```
{ "_id": 1, "item": "apple", "tags": ["fruit", "red", "round", "green"] }
```

Neste exemplo, o `$addToSet` operador adiciona a tag “verde” à matriz “tags” do documento em que o campo “item” é “maçã”. Como “verde” ainda não estava na matriz, ele foi adicionado.

Exemplos de código

Para ver um exemplo de código para usar o `$addToSet` comando, escolha a guia do idioma que você deseja 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

O `$bit` operador no Amazon DocumentDB permite que você execute operações bit a bit nos bits de um determinado campo. Isso pode ser útil para tarefas como definir, limpar ou verificar o estado de bits individuais em um número.

Parâmetros

- `field`: O campo no qual realizar operações bit a bit.
- `and`: um valor inteiro usado para realizar uma operação AND bit a bit no campo.
- `or`: um valor inteiro usado para realizar uma operação OR bit a bit no campo.
- `xor`: um valor inteiro usado para realizar uma operação XOR bit a bit no campo.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$bit` operador para realizar operações bit a bit em um campo numérico.

Crie documentos de amostra

```
db.numbers.insert([
  { "_id": 1, "number": 5 },
  { "_id": 2, "number": 12 }
])
```

Exemplo de consulta

```
db.numbers.update(
  { "_id": 1 },
  { "$bit": { "number": { "and": 3 } } }
)
```

Saída

```
{
  "_id": 1,
  "number": 1
}
```

Neste exemplo, o `$bit` operador é usado para realizar uma operação AND bit a bit no campo "número" do documento com o `_id` de 1. O resultado é que o valor do campo "número" é definido como 1, que é o resultado da operação AND bit a bit entre o valor original de 5 e o valor 3.

Exemplos de código

Para ver um exemplo de código para usar o `$bit` comando, escolha a guia do idioma que você deseja 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

O `$currentDate` operador é usado para definir o valor de um campo para a data e a hora atuais. Esse operador é útil para atualizar automaticamente um campo com o carimbo de data/hora atual quando um documento é inserido ou atualizado.

Parâmetros

- `field`: o campo a ser atualizado com a data e a hora atuais.
- `type`: (opcional) Especifica o tipo de BSON a ser usado na data atual. Pode ser `date` ou `timestamp`.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$currentDate` operador para definir o `lastModified` campo com a data e a hora atuais quando um novo documento é inserido.

Crie documentos de amostra

```
db.users.insert({
  name: "John Doe",
  email: "john.doe@example.com"
})
```

Exemplo de consulta

```
db.users.updateOne(
  { name: "John Doe" },
  { $currentDate: { lastModified: true } }
)
```

Exibir documento atualizado

```
db.users.findOne({ name: "John Doe" })
```

Saída

```
{
  _id: ObjectId('...'),
  name: 'John Doe',
  email: 'john.doe@example.com',
  lastModified: ISODate('2025-10-25T22:50:29.963Z')
}
```

Exemplos de código

Para ver um exemplo de código para usar o `$currentDate` comando, escolha a guia do idioma que você deseja 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

O `$each` operador é usado em conjunto com outros operadores de atualização, como `$push` e`$addToSet`, para adicionar vários valores a um campo de matriz. Ele permite adicionar vários

elementos a uma matriz em uma única operação, em vez de precisar executar várias operações de atualização.

Parâmetros

- `value`: a matriz de valores a serem adicionados ao campo da matriz.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$each` operador com o `$push` operador para adicionar vários elementos a um campo de matriz.

Crie documentos de amostra

```
db.fruits.insertOne({
  _id: 1,
  fruits: ["apple", "banana"]
})
```

Exemplo de consulta

```
db.fruits.updateOne(
  { _id: 1 },
  { $push: { fruits: { $each: ["cherry", "durian", "elderberry"] } } }
)
```

Exibir documento atualizado

```
db.fruits.findOne({ _id: 1 })
```

Saída

```
{
  _id: 1,
  fruits: [ 'apple', 'banana', 'cherry', 'durian', 'elderberry' ]
}
```

Exemplos de código

Para ver um exemplo de código para usar o `$each` comando, escolha a guia do idioma que você deseja 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

O `$inc` operador é usado para incrementar o valor de um campo em um valor especificado. Ele é usado para atualizar um campo numérico, como um contador ou uma classificação, sem precisar recuperar o valor atual, calcular o novo valor e, em seguida, atualizar o campo.

Parâmetros

- `field`: o nome do campo a ser incrementado.
- `amount`: o valor pelo qual incrementar o campo. Isso pode ser um valor positivo ou negativo.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$inc` operador para incrementar o `age` campo de um documento.

Crie documentos de amostra

```
db.users.insertOne({_id: 123, name: "John Doe", age: 30})
```

Exemplo de consulta

```
db.users.updateOne({_id: 123}, {$inc: {age: 1}})
```

Exibir documento atualizado

```
db.users.findOne({_id: 123})
```

Saída

```
{ "_id" : 123, "name" : "John Doe", "age" : 31 }
```

Exemplos de código

Para ver um exemplo de código para usar o `$inc` comando, escolha a guia do idioma que você deseja 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

O operador de `$max` atualização atualiza o valor de um campo somente se o valor especificado for maior que o valor do campo atual. Esse operador é útil para manter valores máximos em todas as atualizações.

Parâmetros

- `field`: O campo a ser atualizado.
- `value`: o valor a ser comparado com o valor do campo atual.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$max` operador para atualizar a pontuação mais alta registrada para um jogador.

Crie documentos de amostra

```
db.scores.insertMany([
  { _id: 1, player: "Alice", highScore: 85 },
  { _id: 2, player: "Bob", highScore: 92 },
  { _id: 3, player: "Charlie", highScore: 78 }
])
```

Exemplo de atualização

```
db.scores.updateOne(
  { _id: 1 },
  { $max: { highScore: 95 } }
)
```

Resultado

O `highScore` campo para Alice é atualizado para 95 porque 95 é maior que o valor atual de 85.

```
{ "_id": 1, "player": "Alice", "highScore": 95 }
```

Exemplos de código

Para ver um exemplo de código para usar o `$max` comando, escolha a guia do idioma que você deseja 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

O operador de `$min` atualização atualiza o valor de um campo somente se o valor especificado for menor que o valor do campo atual. Esse operador é útil para manter valores mínimos em todas as atualizações.

Parâmetros

- `field`: O campo a ser atualizado.
- `value`: o valor a ser comparado com o valor do campo atual.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$min` operador para atualizar a temperatura mais baixa registrada para uma estação meteorológica.

Crie documentos de amostra

```
db.weather.insertMany([
  { _id: 1, station: "Station A", lowestTemp: 15 },
  { _id: 2, station: "Station B", lowestTemp: 20 },
  { _id: 3, station: "Station C", lowestTemp: 18 }
])
```

Exemplo de atualização

```
db.weather.updateOne(
  { _id: 1 },
  { $min: { lowestTemp: 12 } }
)
```

Resultado

O `lowestTemp` campo da Estação A é atualizado para 12 porque 12 é menor que o valor atual de 15.

```
{ "_id": 1, "station": "Station A", "lowestTemp": 12 }
```

Exemplos de código

Para ver um exemplo de código para usar o `$min` comando, escolha a guia do idioma que você deseja 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

O `$mul` operador no Amazon DocumentDB é usado para multiplicar o valor de um campo por um número especificado. Isso pode ser útil para atualizar vários documentos de forma atômica e consistente, como atualizar milhas de voo com base no status do cartão de crédito.

Parâmetros

- `field`: O campo a ser multiplicado.
- `multiplier`: o número pelo qual multiplicar o valor do campo.

Exemplo (MongoDB Shell)

Este exemplo demonstra como usar o `$mul` operador para dobrar o `flight_miles` valor de todos os documentos em que o `credit_card` campo está`true`.

Crie documentos de amostra

```
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] }
]);
```

Exemplo de consulta

```
db.miles.update(
  { "credit_card": { "$eq": true } },
  { "$mul": { "flight_miles.$[]": NumberInt(2) } },
  { "multi": true }
);
```

Saída

```
{ "_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 os clientes que têm cartão de crédito, suas milhas de voo foram dobradas.

O operador de matriz `$[]` posicional é usado para aplicar a `$mul` operação a cada elemento na `flight_miles` matriz.

Exemplos de código

Para ver um exemplo de código para usar o `$mul` comando, escolha a guia do idioma que você deseja 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

O $pop operador no Amazon DocumentDB é usado para remover o primeiro ou o último elemento de um campo de matriz. É particularmente útil quando você precisa manter uma matriz de tamanho fixo ou implementar uma estrutura de dados semelhante a uma fila em um documento.

Parâmetros

- `field`: o nome do campo de matriz do qual remover um elemento.
- `value`: um valor inteiro que determina a posição do elemento a ser removido. Um valor de `1` remove o último elemento, enquanto um valor de `-1` remove o primeiro elemento.

Exemplo (MongoDB Shell)

Este exemplo demonstra como usar o $pop operador para remover o primeiro e o último elementos de um campo de matriz.

Crie documentos de amostra

```
db.users.insertMany([
  { "_id": 1, "name": "John Doe", "hobbies": ["reading", "swimming", "hiking"] },
  { "_id": 2, "name": "Jane Smith", "hobbies": ["cooking", "gardening", "painting"] }
])
```

Exemplo 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 } })
```

Saída

```
{ "_id" : 1, "name" : "John Doe", "hobbies" : [ "swimming", "hiking" ] }
{ "_id" : 2, "name" : "Jane Smith", "hobbies" : [ "cooking", "gardening" ] }
```

Exemplos de código

Para ver um exemplo de código para usar o $pop comando, escolha a guia do idioma que você deseja 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

O `$position` modificador no Amazon DocumentDB especifica a localização na matriz na qual `$push` o operador insere elementos. Sem o `$position` modificador, o `$push` operador insere elementos no final da matriz.

Parâmetros

- `field`: o campo de matriz a ser atualizado.
- num: a posição na matriz em que os elementos devem ser inseridos, com base na indexação com base em zero.

Nota: Para usar o `$position` modificador, ele deve aparecer com o `$each` modificador.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$position` operador para inserir tarefas em posições específicas em um sistema de gerenciamento de projetos.

Crie documentos de amostra

```
db.projects.insertOne({ "_id": 1, "name": "Website Redesign", "tasks": ["Design
 mockups"] })
```

Exemplo de consulta 1 - Adicione tarefas urgentes no início

```
db.projects.updateOne(
   { _id: 1 },
   {
     $push: {
        tasks: {
           $each: ["Security audit", "Performance review"],
           $position: 0
        }
     }
   }
)
```

Saída 1

```
{ "_id": 1, "name": "Website Redesign", "tasks": ["Security audit", "Performance
 review", "Design mockups"] }
```

Exemplo de consulta 2 - Adicionar tarefas em uma posição 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
         }
      }
   }
)
```

Saída 2

```
{ "_id": 2, "name": "Mobile App", "tasks": ["Setup project", "Create wireframes", "Code
 review", "Testing phase", "Deploy to store"] }
```

Exemplos de código

Para ver um exemplo de código para usar o `$position` comando, escolha a guia do idioma que você deseja 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

O `$pull` operador é usado para remover de uma matriz todas as instâncias de um valor ou valores que correspondam a uma condição especificada. Esse operador é útil quando você precisa remover elementos específicos de um campo de matriz em um documento.

Parâmetros

- `field`: o nome do campo de matriz do qual remover o (s) valor (es).
- `value`: o valor ou condição que determina quais elementos devem ser removidos da matriz.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$pull` operador para remover elementos de um campo de matriz.

Crie documentos de amostra

```
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"]
  }
])
```

Exemplo de consulta

```
db.restaurants.updateMany(
  { cuisine: "Italian" },
  { $pull: { features: "Takeout" } }
)
```

Saída

```
{
  "acknowledged" : true,
  "matchedCount" : 1,
  "modifiedCount" : 1
```

```
}
```

A consulta acima remove o recurso “Takeout” de todos os documentos em que o `cuisine` campo é “italiano”.

Exemplos de código

Para ver um exemplo de código para usar o `$pull` comando, escolha a guia do idioma que você deseja 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

O `$pullAll` operador no Amazon DocumentDB é usado para remover todas as instâncias dos valores especificados de um campo de matriz. Isso é particularmente útil quando você precisa remover vários elementos de uma matriz em uma única operação.

Parâmetros

- `field`: o nome do campo de matriz do qual remover os elementos.
- `value`: uma matriz de valores a serem removidos do campo da matriz.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$pullAll` operador para remover vários elementos de um campo de matriz.

Crie documentos de amostra

```
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"]
  }
])
```

Exemplo de consulta

```
db.restaurants.update(
  { "name": "Taj Mahal" },
  { $pullAll: { "features": ["Private Dining", "Live Music"] } }
)
```

Saída

```
{
  "name": "Taj Mahal",
  "cuisine": "Indian",
  "features": []
}
```

Exemplos de código

Para ver um exemplo de código para usar o `$pullAll` comando, escolha a guia do idioma que você deseja 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

O `$push` operador no Amazon DocumentDB é usado para adicionar um item a um campo de matriz em um documento. Esse operador é particularmente útil quando você precisa acrescentar novos dados a uma matriz existente sem sobrescrever a matriz inteira.

Parâmetros

- `field`: o nome do campo de matriz ao qual o novo elemento deve ser adicionado.
- `value`: o valor a ser adicionado à matriz.
- `position`: (opcional) Um modificador que especifica a posição na matriz em que o novo elemento deve ser adicionado. Os modificadores compatíveis incluem $ (adicionar ao final da matriz) e `$[]` (adicionar ao final da matriz, ignorando quaisquer filtros da matriz).

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$push` operador para adicionar novos elementos a um campo de matriz em um documento.

Crie documentos de amostra

```
db.users.insert([
  { _id: 1, name: "John Doe", hobbies: ["reading", "swimming"] },
  { _id: 2, name: "Jane Smith", hobbies: ["gardening", "cooking"] }
])
```

Exemplo de consulta

```
db.users.updateOne(
  { _id: 1 },
  { $push: { hobbies: "hiking" } }
)
```

Saída

```
{
  "acknowledged" : true,
  "matchedCount" : 1,
  "modifiedCount" : 1
}
```

Depois de executar a atualização, o documento com `_id: 1` terá a `hobbies` matriz atualizada para`["reading", "swimming", "hiking"]`.

Exemplos de código

Para ver um exemplo de código para usar o `$push` comando, escolha a guia do idioma que você deseja 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

O `$rename` operador no Amazon DocumentDB é usado para renomear um campo em um documento. Esse operador pode ser particularmente útil quando você precisa atualizar a estrutura de seus documentos ou alinhá-los com novos modelos de dados.

Parâmetros

- `field`: O campo a ser renomeado.
- `newName`: o novo nome do campo.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$rename` operador para renomear o `"Date.DoW"` campo para `"Date.DayOfWeek"` em um documento com o `"DocName"` campo definido como. `"Document 1"`

Crie documentos de amostra

```
db.example.insertOne({
    "DocName": "Document 1",
    "Date": {
        "Month": 4,
        "Day": 18,
        "Year": 1987,
        "DoW": "Saturday"
    },
    "Words": 2482
})
```

Exemplo de consulta

```
db.example.update(
    { "DocName": "Document 1" },
    { $rename: { "Date.DoW": "Date.DayOfWeek" } }
)
```

Saída

```
{
```

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

Exemplos de código

Para ver um exemplo de código para usar o `$rename` comando, escolha a guia do idioma que você deseja 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

O `$set` operador no Amazon DocumentDB é usado para atualizar o valor de um campo especificado em um documento. Esse operador permite que você adicione novos campos ou modifique os existentes em um documento. É um operador de atualização fundamental no driver Java do MongoDB, compatível com o Amazon DocumentDB.

Parâmetros

- `field`: O campo a ser atualizado.
- `value`: o novo valor do campo.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$set` operador para atualizar o `Item` campo em um documento.

Crie documentos de amostra

```
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
    }
  }
])
```

Exemplo de consulta

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

Saída

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

Exemplos de código

Para ver um exemplo de código para usar o `$set` comando, escolha a guia do idioma que você deseja 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()
```

$ set OnInsert

O `$setOnInsert` operador no Amazon DocumentDB é usado para definir o valor de um campo se um documento estiver sendo inserido, mas não terá efeito se o documento estiver sendo atualizado.

Parâmetros

- `field`: O campo a ser definido.
- `value`: o valor a ser atribuído ao campo.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$setOnInsert` operador no Amazon DocumentDB. Ele cria um novo documento se o documento ainda não existir, mas não tem efeito se o documento estiver sendo atualizado.

Crie documentos de amostra

```
db.users.insertOne({
  _id: 1,
  name: "John Doe",
  age: 30
})
```

Exemplo de consulta 1 - Atualizar documento existente

```
db.users.update(
  { _id: 1 },
  {
    $set: { age: 31 },
    $setOnInsert: { createdAt: new Date() }
  },
  { upsert: true }
)
```

Saída 1

```
{
  _id: 1,
  name: "John Doe",
```

```
  age: 31
}
```

A saída mostra que o documento foi atualizado, mas o `createdAt` campo NÃO foi adicionado porque o documento já existia. O `$setOnInsert` operador só se aplica ao inserir novos documentos.

Exemplo de consulta 2 - Inserir novo documento (upsert)

```
db.users.update(
  { _id: 2 },
  {
    $set: { name: "Jane Smith", age: 25 },
    $setOnInsert: { createdAt: new Date() }
  },
  { upsert: true }
)
```

Saída 2

```
{
  _id: 2,
  name: "Jane Smith",
  age: 25,
  createdAt: ISODate("2025-10-31T09:57:52.459Z")
}
}
```

Exemplos de código

Para ver um exemplo de código para usar o `$setOnInsert` comando, escolha a guia do idioma que você deseja 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

O operador de `$slice` atualização modifica uma matriz limitando seu tamanho. Quando usado com o `$push` operador, ele restringe o número de elementos em uma matriz, mantendo somente o número especificado dos elementos mais recentes ou mais antigos.

Parâmetros

- `field`: o campo de matriz a ser modificado.
- `count`: Número máximo de elementos a serem mantidos. Valores positivos mantêm os primeiros N elementos, valores negativos mantêm os últimos N elementos.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o operador de `$slice` atualização com `$push` para manter uma matriz de tamanho fixo das pontuações recentes.

Crie documentos de amostra

```
db.students.insertOne({
  _id: 1,
  name: "Alice",
  scores: [85, 90, 78]
});
```

Exemplo de consulta

```
db.students.updateOne(
  { _id: 1 },
  {
    $push: {
      scores: {
        $each: [92, 88],
        $slice: -3
      }
    }
```

```
  }
)
```

Saída

```
{
  "_id" : 1,
  "name" : "Alice",
  "scores" : [ 78, 92, 88 ]
}
```

Neste exemplo, o `$slice: -3` modificador mantém somente os últimos três elementos após inserir novos valores na matriz.

Exemplos de código

Para ver um exemplo de código para usar o operador de `$slice` atualização, escolha a guia do idioma que você deseja 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

O modificador de `$sort` atualização ordena os elementos da matriz quando usado com o `$push` operador. Ele organiza os elementos da matriz em ordem crescente ou decrescente com base nos valores de campo especificados ou nos próprios elementos.

Parâmetros

- `field`: o campo de matriz a ser modificado.
- `order`: Use 1 para ordem crescente ou `-1` decrescente.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra o uso do `$sort` modificador with `$push` para adicionar novas pontuações do questionário e mantê-las classificadas em ordem decrescente.

Crie documentos de amostra

```
db.students.insertOne({
  _id: 1,
  name: "Bob",
  quizzes: [
    { score: 85, date: "2024-01-15" },
    { score: 92, date: "2024-02-10" }
  ]
});
```

Exemplo de consulta

```
db.students.updateOne(
  { _id: 1 },
  {
    $push: {
      quizzes: {
        $each: [{ score: 78, date: "2024-03-05" }],
        $sort: { score: -1 }
      }
    }
  }
)
```

Saída

```
{
  "_id" : 1,
  "name" : "Bob",
```

```
  "quizzes" : [
    { "score" : 92, "date" : "2024-02-10" },
    { "score" : 85, "date" : "2024-01-15" },
    { "score" : 78, "date" : "2024-03-05" }
  ]
}
```

Exemplos de código

Para ver um exemplo de código para usar o modificador de `$sort` atualização, escolha a guia do idioma que você deseja 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

O `$unset` operador no Amazon DocumentDB é usado para remover um campo especificado de um documento. Quando um campo é removido usando`$unset`, o campo é excluído do documento e o tamanho do documento é reduzido de acordo. Isso pode ser útil quando você deseja remover dados desnecessários de seus documentos.

Parâmetros

- `field`: o campo a ser removido do documento. Isso pode ser um único campo ou um caminho pontilhado para um campo aninhado.

Exemplo (MongoDB Shell)

O exemplo a seguir demonstra como usar o `$unset` operador para remover o `Words` campo de um documento na `example` coleção.

Crie documentos de amostra

```
db.example.insert({
    "DocName": "Document 1",
    "Date": {
        "Month": 4,
        "Day": 18,
        "Year": 1987,
        "DoW": "Saturday"
    },
    "Words": 2482
})
```

Exemplo de consulta

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

Saída

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

Neste exemplo, o `$unset` operador é usado para remover o `Words` campo do documento `DocName` igual a “Documento 1". O documento resultante não contém mais o `Words` campo.

Exemplos de código

Para ver um exemplo de código para usar o `$unset` comando, escolha a guia do idioma que você deseja 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()
```

Gerenciamento de clusters e instâncias do Amazon DocumentDB

Essas seções abrangem os vários componentes e suas tarefas relacionadas para gerenciar sua implementação do Amazon DocumentDB (compatível com MongoDB).

Tópicos

- Visão geral das tarefas operacionais do Amazon DocumentDB
- Visão geral dos clusters globais do Amazon DocumentDB
- Gerenciar clusters do Amazon DocumentDB
- Gerenciar instâncias do Amazon DocumentDB
- Gerenciamento de grupos de sub-rede do Amazon DocumentDB
- Alta disponibilidade e replicação do Amazon DocumentDB
- Gerenciando eventos do Amazon DocumentDB
- Gerenciando grupos de parâmetros de cluster do Amazon DocumentDB
- Noções básicas sobre endpoints do Amazon DocumentDB
- Compreender os nomes do recursos da Amazon (ARN) do Amazon DocumentDB
- Marcação de recursos do Amazon DocumentDB
- Manutenção do Amazon DocumentDB
- Noções básicas dos perfis vinculados ao serviço

Visão geral das tarefas operacionais do Amazon DocumentDB

Esta seção aborda as tarefas operacionais do seu cluster do Amazon DocumentDB e como realizar essas tarefas usando a AWS CLI.

Tópicos

- Adicionar uma réplica a um cluster do Amazon DocumentDB
- Descrever clusters e instâncias
- Criar um snapshot de cluster

- Restauração a partir de um snapshot
- Remover uma instância de um cluster
- Excluir um cluster

Adicionar uma réplica a um cluster do Amazon DocumentDB

Depois de criar a instância principal para seu cluster do Amazon DocumentDB, você pode adicionar uma ou mais réplicas. Uma réplica é uma instância somente leitura que serve a duas finalidades:

- Escalabilidade — se houver um grande número de clientes que exijam acesso simultâneo, você poderá adicionar mais réplicas para leitura em escala.
- Alta disponibilidade — se a instância principal falhar, o Amazon DocumentDB executará um failover automaticamente em uma instância de réplica e a designará como a nova principal. Se uma réplica falhar, outras instâncias no cluster ainda poderão atender as solicitações até que o nó com falha possa ser recuperado.

Cada cluster do Amazon DocumentDB pode oferecer suporte a até 15 réplicas.

> **Note**
>
> Para obter a máxima tolerância a falhas, você deve implantar as réplicas em Zonas de Disponibilidade separadas. Isso ajuda a garantir que seu cluster do Amazon DocumentDB possa continuar a funcionar, mesmo que toda uma Zona de Disponibilidade fique indisponível.

O AWS CLI exemplo a seguir mostra como adicionar uma nova réplica. O parâmetro `--availability-zone` coloca a réplica na zona de disponibilidade 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
```

Descrever clusters e instâncias

O AWS CLI exemplo a seguir lista todos os clusters do Amazon DocumentDB em uma região. Para determinados atributos de gerenciamento, como o gerenciamento do ciclo de vida de clusters e instâncias, o Amazon DocumentDB aproveita a tecnologia operacional compartilhada com o Amazon RDS. O parâmetro de filtro `filterName=engine,Values=docdb` só retorna clusters do Amazon DocumentDB.

Para obter mais informações sobre como descrever e modificar clusters, consulte Ciclo de vida do cluster do Amazon DocumentDB.

```
aws docdb describe-db-clusters --filter Name=engine,Values=docdb
```

A saída dessa operação é semelhante à seguinte.

```
{
    "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",
            ...
        }
    ]
}
```

O AWS CLI exemplo a seguir lista as instâncias em um cluster do Amazon DocumentDB. Para obter mais informações sobre como descrever e modificar clusters, consulte Ciclo de vida da instância do Amazon DocumentDB.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].[DBClusterMembers]'
```

A saída é semelhante ao apontado abaixo. Nesta saída, há duas instâncias. A instância principal é `sample-instance-1` (`"IsClusterWriter": true`). Há também uma instância 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
            }
        ]
    ]
]
```

Criar um snapshot de cluster

Uma captura de tela de cluster é um backup completo dos dados no seu cluster do Amazon DocumentDB. Quando a captura de tela estiver sendo criada, o Amazon DocumentDB lerá os dados diretamente do volume de cluster. Por causa disso, você pode criar uma captura de tela mesmo que seu cluster não tenha nenhuma instância em execução no momento. O tempo necessário para criar uma captura de tela depende do tamanho do volume do cluster.

O Amazon DocumentDB oferece suporte a backups automáticos, que ocorrem diariamente durante a janela de backup preferencial — um período de 30 minutos durante o dia. O AWS CLI exemplo a seguir mostra como visualizar a janela de backup do seu cluster:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].PreferredBackupWindow'
```

A saída mostra a janela de backup (em UTC):

```
[
    "00:18-00:48"
]
```

Você pode definir a janela de backup ao criar o cluster do Amazon DocumentDB. Você também pode alterar a janela de backup, conforme mostrado no exemplo a seguir. Se você não definir uma janela de backup, o Amazon DocumentDB atribuirá uma automaticamente ao cluster.

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --preferred-backup-window "02:00-02:30"
```

Além dos backups automáticos, você pode criar manualmente uma captura de tela de cluster a qualquer momento. Ao fazer isso, você especifica o cluster cujo backup deseja fazer e um nome exclusivo para a captura de tela, para que você possa restaurar a partir dela posteriormente.

O AWS CLI exemplo a seguir mostra como criar um instantâneo dos seus dados.

```
aws docdb create-db-cluster-snapshot \
    --db-cluster-identifier sample-cluster \
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

Restauração a partir de um snapshot

Você pode restaurar uma captura de tela de cluster em um novo cluster do Amazon DocumentDB. Para fazer isso, forneça o nome da captura de tela e o nome de um novo cluster. Não é possível restaurar a partir de uma captura de tela para um cluster existente. Em vez disso, o Amazon DocumentDB criará um cluster quando você fizer a restauração e o preencher com os dados da captura de tela.

O exemplo a seguir mostra todas as capturas de tela de um determinado cluster `sample-cluster`.

```
aws docdb describe-db-cluster-snapshots \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusterSnapshots[*].[DBClusterSnapshotIdentifier,SnapshotType,Status]'
```

A saída é semelhante ao apontado abaixo. Uma captura de tela manual é criada manualmente, enquanto uma captura de tela automatizada é criado pelo Amazon DocumentDB dentro da janela de backup do cluster.

```
[
        "sample-cluster-snapshot",
        "manual",
        "available"
    ],
    [
        "rds:sample-cluster",
        "automated",
        "available"
    ]
]
```

O exemplo a seguir mostra como restaurar um cluster do Amazon DocumentDB a partir de uma captura de tela.

```
aws docdb restore-db-cluster-from-snapshot \
```

```
    --engine docdb \
    --db-cluster-identifier new-sample-cluster \
    --snapshot-identifier sample-cluster-snapshot
```

O novo cluster não tem nenhuma instância associada a ele, portanto, para interagir com o cluster, adicione uma instância a ele.

```
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
```

Você pode usar as AWS CLI operações a seguir para monitorar o progresso da criação de clusters e instâncias. Quando os status do cluster e da instância estiverem disponíveis, você poderá se conectar ao endpoint do novo cluster e acessar seus dados.

```
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]'
```

Remover uma instância de um cluster

O Amazon DocumentDB armazena todos os dados no volume do cluster. Os dados persistem nesse volume de cluster, mesmo se você remover todas as instâncias do cluster. Se precisar acessar os dados novamente, adicione uma instância ao cluster a qualquer momento e continue de onde parou.

O exemplo a seguir mostra como remover uma instância de seu cluster do Amazon DocumentDB.

```
aws docdb delete-db-instance \
    --db-instance-identifier sample-instance
```

Excluir um cluster

Para excluir um cluster do Amazon DocumentDB, é necessário primeiro remover todas as instâncias. O AWS CLI exemplo a seguir retorna informações sobre as instâncias em um cluster. Se essa

operação retornar qualquer identificador de instância, você terá que excluir cada uma das instâncias. Para obter mais informações, consulte Remover uma instância de um cluster.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].DBClusterMembers[*].DBInstanceIdentifier'
```

Quando não houver mais instâncias restantes, você poderá excluir o cluster. Nesse momento, você deve escolher uma das seguintes opções:

- Criar uma captura de tela final — capture todos os dados do cluster em uma captura de tela para recriar uma instância com esses dados posteriormente. O exemplo a seguir mostra como fazê-lo:

  ```
  aws docdb delete-db-cluster \
      --db-cluster-identifier sample-cluster \
      --final-db-snapshot-identifier sample-cluster-snapshot
  ```

- Ignore o snapshot final — descarte permanentemente todos os dados do cluster. Essa ação não pode ser revertida. O exemplo a seguir mostra como fazê-lo:

  ```
  aws docdb delete-db-cluster \
      --db-cluster-identifier sample-cluster \
      --skip-final-snapshot
  ```

Visão geral dos clusters globais do Amazon DocumentDB

O que é um cluster global?

Um cluster global consiste em uma região primária e até 10 regiões secundárias somente para leitura. Você emite operações de gravação diretamente no cluster primário na região primária e o Amazon DocumentDB replica automaticamente os dados para as regiões secundárias usando uma infraestrutura dedicada. Geralmente, a latência é inferior a um segundo.

Por que os clusters globais são úteis?

- Recuperação de Region-wide interrupções — no caso de uma Region-wide interrupção, você pode promover um dos clusters secundários a um cluster primário em minutos, com um objetivo de tempo de recuperação (RTO) típico de menos de um minuto. Normalmente, o objetivo de ponto

de recuperação (RPO) é medido em segundos, mas isso depende do atraso em toda a rede no momento da falha.

- Leituras globais com latência local — Se você tem escritórios em todo o mundo, pode usar um cluster global para manter suas principais fontes de informações atualizadas na região principal. Escritórios em outras regiões podem acessar as informações em sua própria região com latência local.
- Clusters secundários escaláveis — Você pode escalar seus clusters secundários adicionando mais instâncias somente para leitura a uma região secundária. O cluster secundário é somente leitura, portanto pode oferecer suporte a até 16 instâncias de réplica somente leitura, em vez do limite normal de 15 para um único cluster.
- Replicação rápida de clusters de primários para secundários: replicação realizada por um cluster global tem pouco impacto no desempenho do cluster de banco de dados primário. Os recursos das instâncias de banco de dados são totalmente dedicados para atender as workloads de leitura e gravação.
- Fluxos de alteração — Você pode ler fluxos de alteração de um cluster secundário em clusters globais, reduzindo a carga no cluster primário e permitindo escalar cargas de trabalho de leitura de regiões secundárias.

Quais são as limitações atuais dos clusters globais?

- Os clusters globais não são compatíveis com a versão 3.6 do Amazon DocumentDB.
- Os clusters globais são compatíveis com todos os tipos de instância, exceto db.t3, db.t4g e db.r4.
- Os clusters globais não estão disponíveis nas regiões a seguir: América do Sul (São Paulo), Europa (Milão), China (Pequim) e China (Ningxia).
- A alternância e o failover global não são suportados quando as regiões estão em versões de mecanismo diferentes. Há suporte para failover manual quando existe uma incompatibilidade entre as versões do mecanismo.
- Somente o cluster primário realiza operações de gravação. Os clientes que realizam operações de gravação se conectam ao endpoint do cluster primário.
- Você pode ter no máximo 10 regiões secundárias e uma região primária para seu cluster.
- Um cluster secundário não pode ser interrompido. Um cluster primário não pode ser interrompido se tiver clusters secundários associados a ele. Somente um cluster regional que não tenha clusters secundários pode ser interrompido.

- Réplicas anexadas ao cluster secundário podem ser reiniciadas em determinadas circunstâncias. Se a instância da região primária for reiniciada ou falhar, as réplicas na região secundária também serão reiniciadas. O cluster fica indisponível até que todas as réplicas estejam novamente sincronizadas com a instância do gravador do cluster de banco de dados primário. Esse comportamento é esperado. Certifique-se de que você entendeu o impacto no seu cluster global antes de fazer alterações no cluster primário.

Tópicos

- Guia de início rápido: clusters globais
- Gerenciar um cluster global do Amazon DocumentDB
- Conexão com um cluster global do Amazon DocumentDB
- Monitorar clusters globais do Amazon DocumentDB
- Recuperação de desastres e clusters globais do Amazon DocumentDB

Guia de início rápido: clusters globais

Tópicos

- Configuração
- Criação de um cluster global do Amazon DocumentDB
- Adicionando um Região da AWS para um cluster global do Amazon DocumentDB
- Usando um snapshot para seu cluster Amazon DocumentDB global

Configuração

O cluster global Amazon DocumentDB abrange pelo menos dois. Regiões da AWS A região primária oferece suporte a um cluster que tem uma instância primária (gravadora) e até 15 instâncias de réplica, enquanto uma região secundária executa um cluster somente de leitura composto inteiramente por até 16 instâncias de réplica. Um cluster global pode ter até 10 regiões secundárias. A tabela lista o máximo de clusters, instâncias e réplicas permitidos em um cluster global.

Description	Primário Região da AWS	Secundário Região da AWS
Clusters	1	10 (máximo)

Description	Primário Região da AWS	Secundário Região da AWS
Instâncias do gravador	1	0
Read-only instâncias (réplicas do Amazon DocumentDB), por cluster	15 (máximo)	16 (total)
Read-only instâncias (máximo permitido, dado o número real de regiões secundárias)	15 - s	s = número total de secundários Regiões da AWS

Os clusters têm os seguintes requisitos específicos:

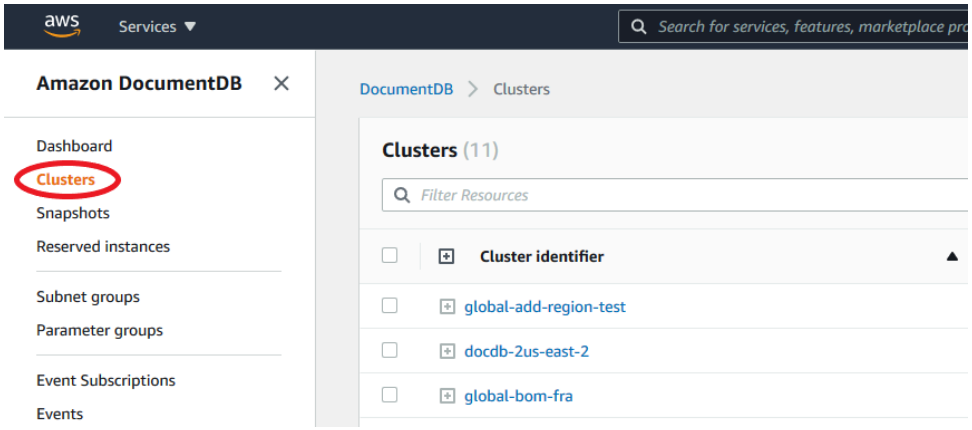
- Requisitos de classe de instância de banco de dados — Você só pode usar as classes de instância `db.r5` e `db.r6g`.
- Região da AWS requisitos — O cluster primário deve estar em uma região e pelo menos um cluster secundário deve estar em uma região diferente da mesma conta. Você pode criar até 10 clusters secundários (somente leitura), e cada um deve estar em uma região diferente. Em outras palavras, não pode haver dois clusters na mesma região.
- Requisitos de nome: os nomes escolhidos para cada um de seus clusters devem ser exclusivos em todas as regiões. Não é possível usar o mesmo nome para clusters diferentes, mesmo que eles estejam em regiões diferentes.

Criação de um cluster global do Amazon DocumentDB

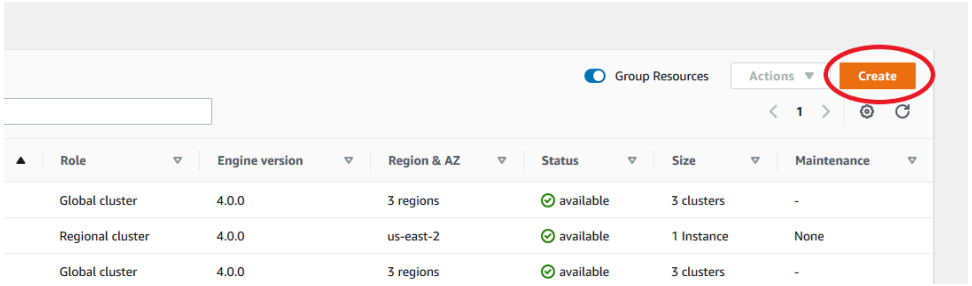
Você está pronto para criar seu primeiro cluster global? Nesta seção, explicaremos como criar um novo cluster global com novos clusters e instâncias de banco de dados, usando Console de gerenciamento da AWS ou AWS CLI com as instruções a seguir.

Usar o Console de gerenciamento da AWS

1. Em Console de gerenciamento da AWS, navegue até Amazon DocumentDB.
2. Ao acessar o console do Amazon DocumentDB, escolha Clusters.

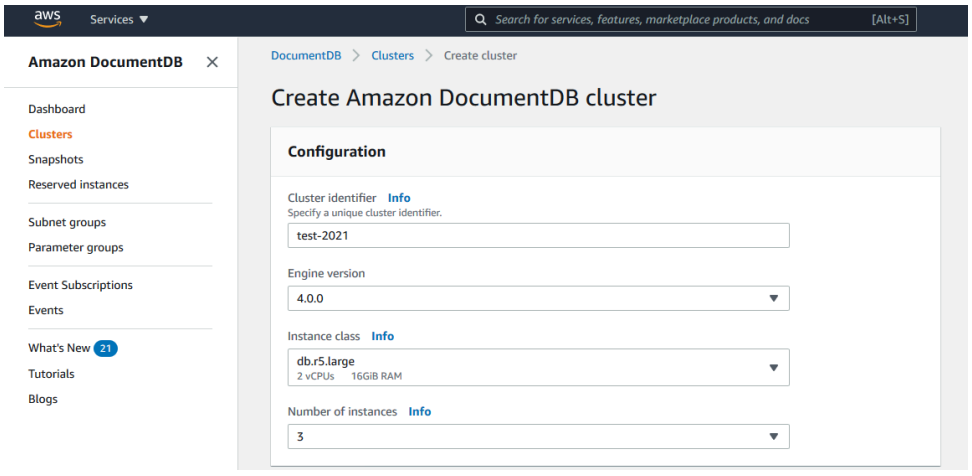


3. Escolha Criar.

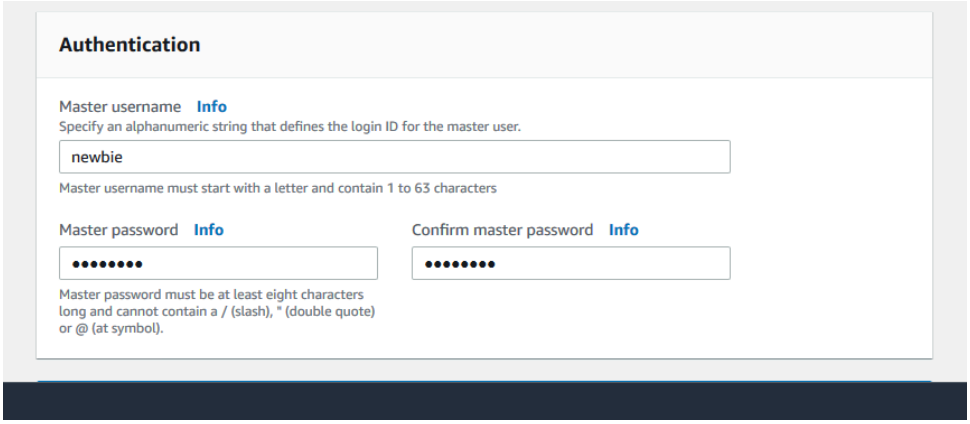


4. Preencha a seção Configuração do formulário Criar cluster do Amazon DocumentDB:

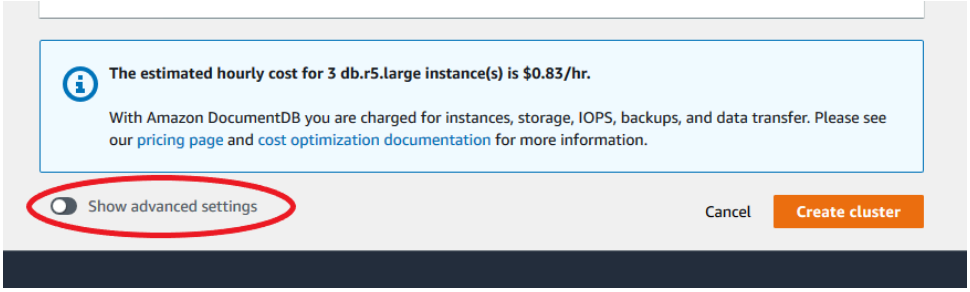
   - Identificador de cluster: você pode inserir um identificador exclusivo para essa instância ou permitir que o Amazon DocumentDB forneça o identificador da instância com base no identificador do cluster.
   - Versão do mecanismo: escolha 4.0.0
   - Classe de instância: escolha db.r5.large
   - Número de instâncias: escolha 3.



5. Na seção Autenticação, preencha um nome de usuário principal e uma senha mestra.

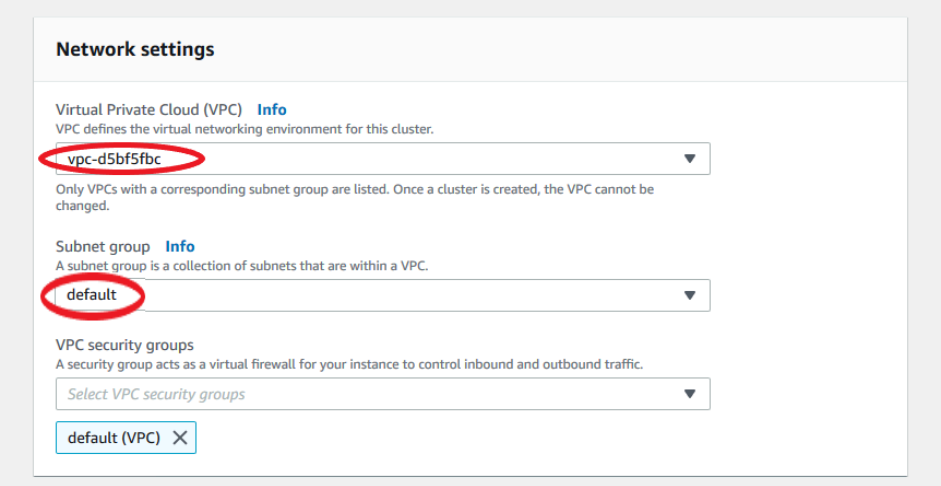


6. Escolha Exibir configurações avançadas.

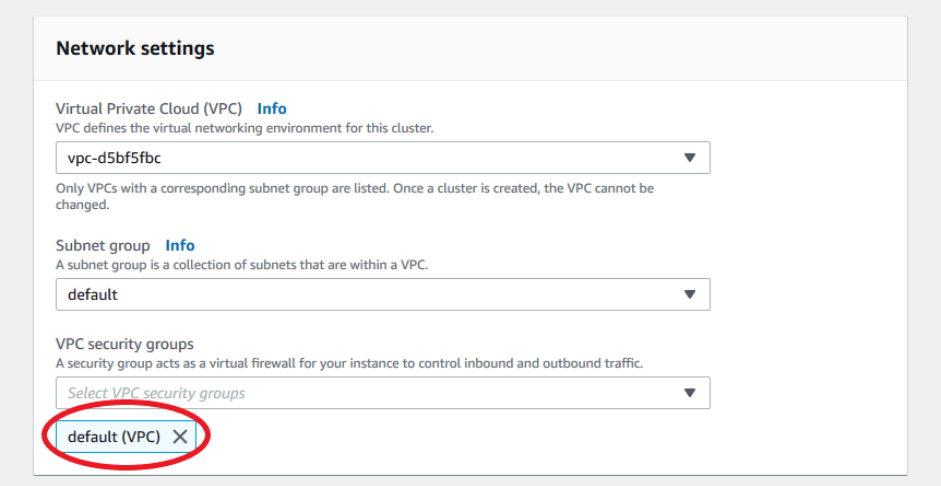


7. Na seção Configurações de rede:

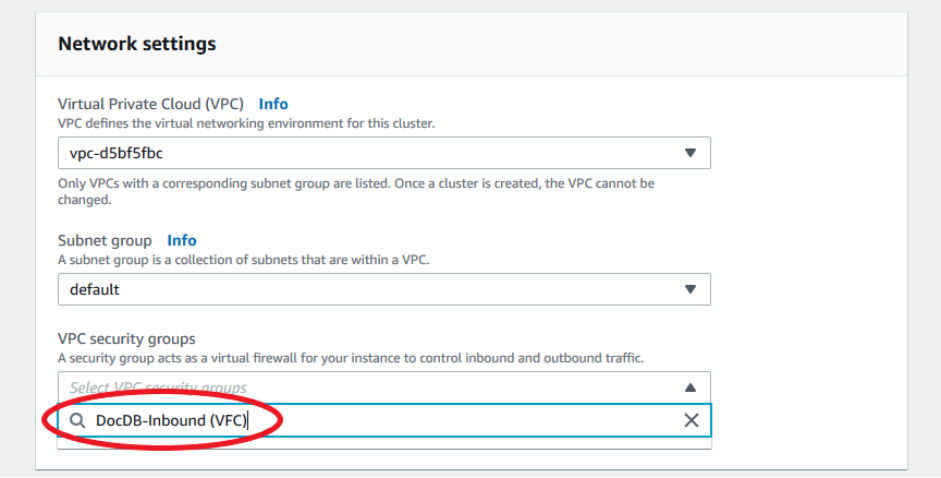
   - Mantenha as opções padrão para nuvem privada virtual (VPC) e Grupo de sub-redes.



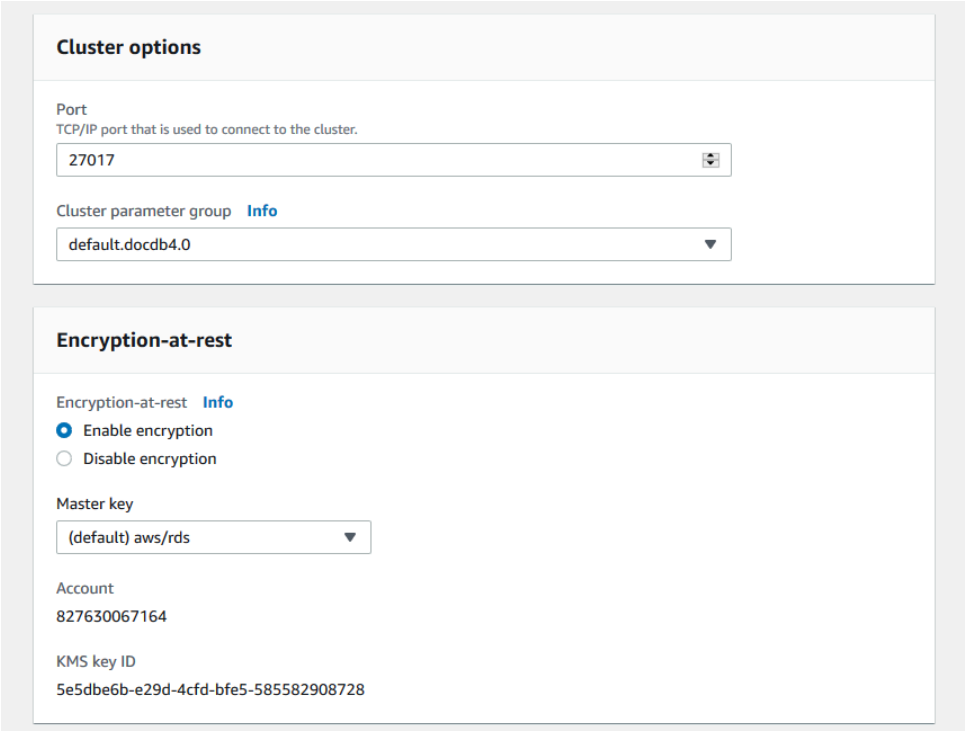
   - Em Grupos de segurança de VPC, um padrão (VPC) já deve ter sido adicionado.



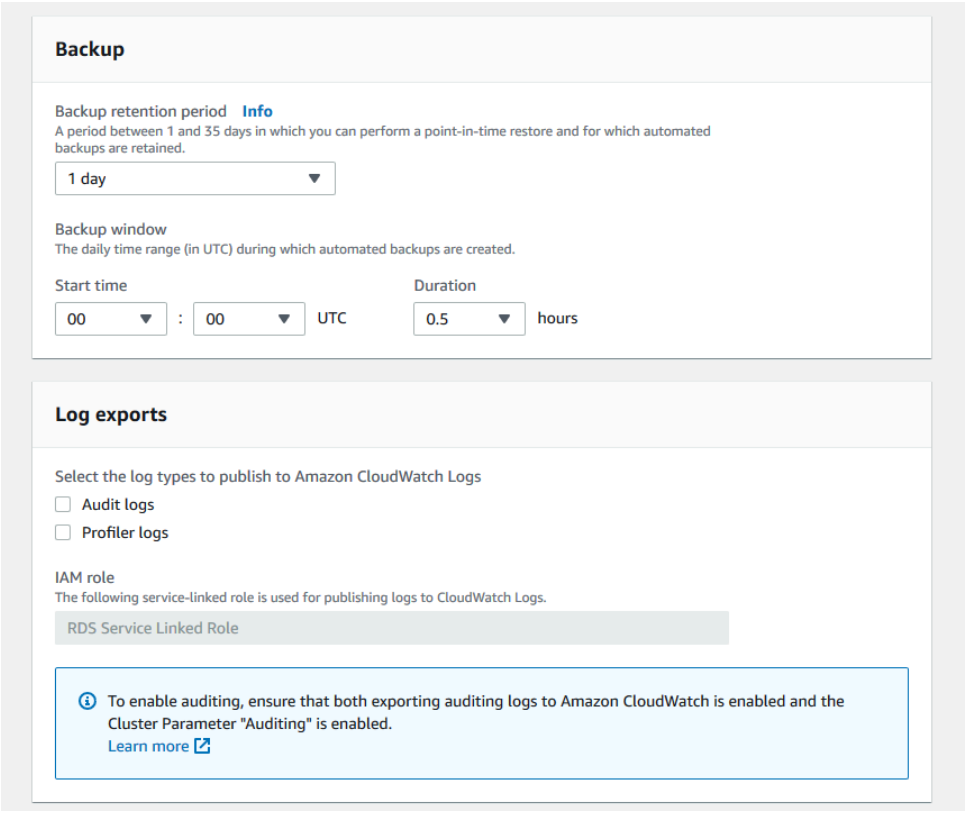
- Digite DocDB no campo VPC security groups e selecione DocDB-Inbound (VFC).



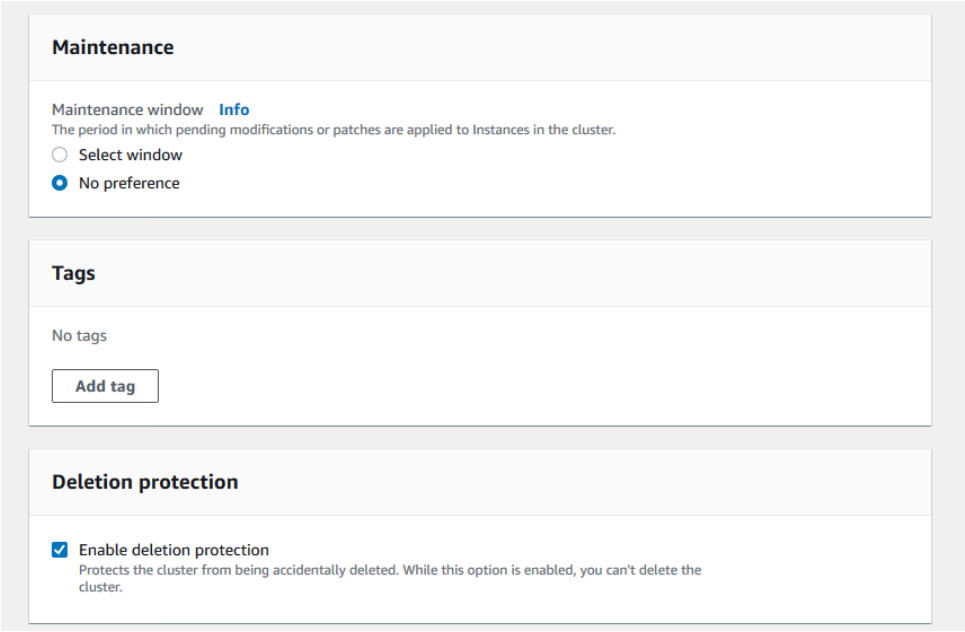
8. Para opções de cluster e Encryption-at-rest, deixe as seleções padrão.



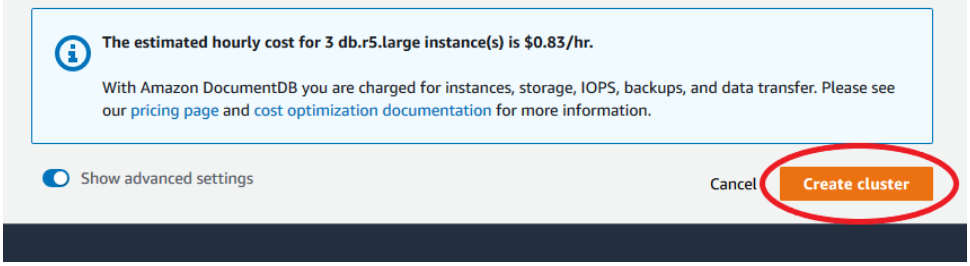
9. Em Backup e Exportações de logs, mantenha as seleções padrão.



10. Em Manutenção, Tags e Proteção contra exclusão, mantenha as seleções padrão.



11. Agora escolha Criar cluster.



Usar o AWS CLI

Para criar um cluster regional do Amazon DocumentDB, chame o AWS CLI create-global-cluster. O AWS CLI comando a seguir cria um cluster Amazon DocumentDB chamado`global-cluster-id`. Para obter mais informações sobre a proteção contra exclusão, consulte Excluir um cluster do Amazon DocumentDB.

Além disso, `--engine-version` é um parâmetro opcional padrão para a versão mais recente do mecanismo principal. A versão padrão atual do mecanismo é `5.0.0` (observação: o Amazon DocumentDB 8.0 está disponível, mas deve ser explicitamente especificado como`8.0.0`). Quando novas versões principais do motor são lançadas, a versão padrão do motor `--engine-version` é atualizada para refletir a última versão do motor principal. Como resultado, para cargas de trabalho de produção, especialmente aquelas que dependem de scripts, automação ou CloudFormation modelos, especifique explicitamente `--engine-version` a versão principal pretendida.

Se um `db-subnet-group-name` ou `vpc-security-group-id` não for especificado, o Amazon DocumentDB usará o grupo de sub-redes padrão e o grupo de segurança da Amazon VPC para a região em questão.

No exemplo a seguir, substitua cada *`user input placeholder`* por suas próprias informações.

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante ao seguinte (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"
    }
}
```

Leva alguns minutos para criar o cluster. Você pode usar o Console de gerenciamento da AWS ou AWS CLI para monitorar o status do seu cluster. Para obter mais informações, consulte Monitoramento do status de um cluster do Amazon DocumentDB.

> **Important**
>
> Quando você usa o AWS CLI para criar um cluster regional do Amazon DocumentDB, nenhuma instância é criada. Consequentemente, é necessário criar explicitamente uma instância principal e qualquer instância de réplica de que precise. Você pode usar o console ou AWS CLI criar as instâncias. Para obter mais informações, consulte Adicionar uma instância do Amazon DocumentDB a um cluster e crie DBCluster na Referência de API do Amazon DocumentDB.

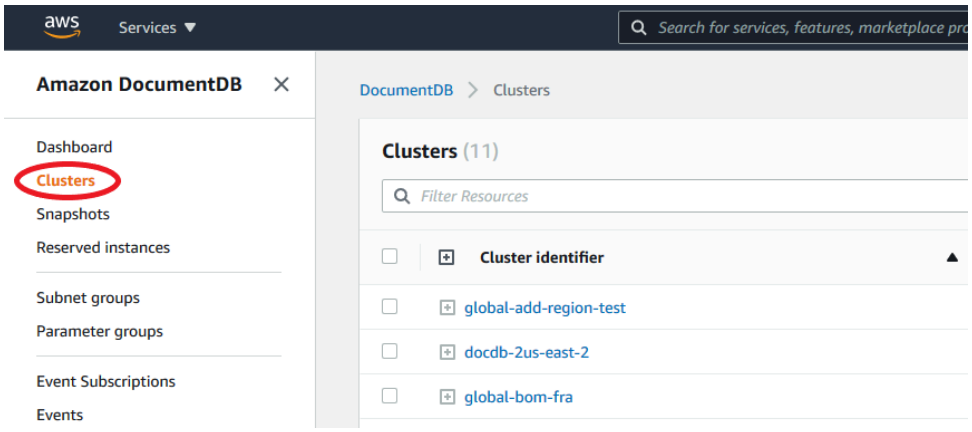
Quando seu cluster regional estiver disponível, será possível adicionar um cluster secundário em outra região com as instruções a seguir: Adicionando um Região da AWS para um cluster global do Amazon DocumentDB. Quando você adicionar uma região, seu cluster regional se tornará seu cluster primário, e você terá um novo cluster secundário na região escolhida.

Adicionando um Região da AWS para um cluster global do Amazon DocumentDB

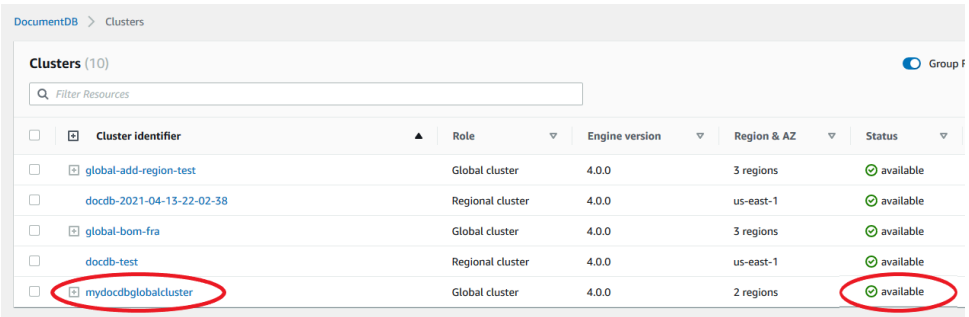
Um cluster global precisa de pelo menos um cluster secundário em uma região diferente do cluster primário, e você pode adicionar até 10 clusters secundários. Observe que, para cada cluster secundário que você adicionar, deverá reduzir em um o número de réplicas permitidas no cluster primário. Por exemplo, se seu cluster global tiver 10 regiões secundárias, seu cluster primário poderá ter apenas 5 (em vez de 15) réplicas. Para obter mais informações, consulte Configuração.

Usar o Console de gerenciamento da AWS

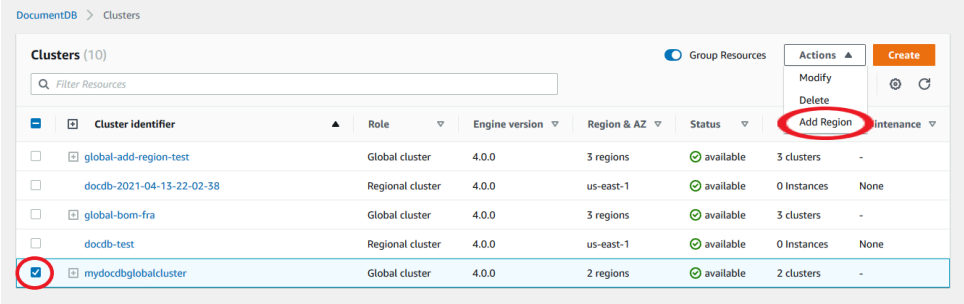
1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon DocumentDB.
2. No painel de navegação, escolha Clusters.



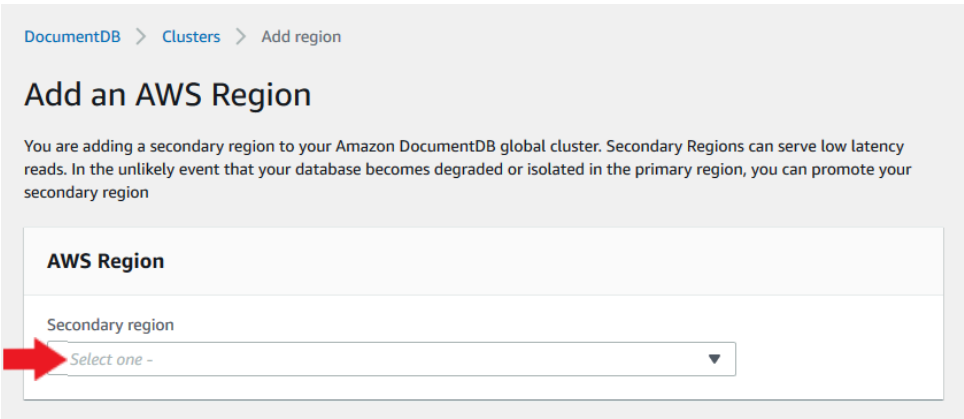
3. Escolha o cluster ao qual você deseja adicionar ao cluster secundário. Certifique-se de que o cluster seja `Available`.



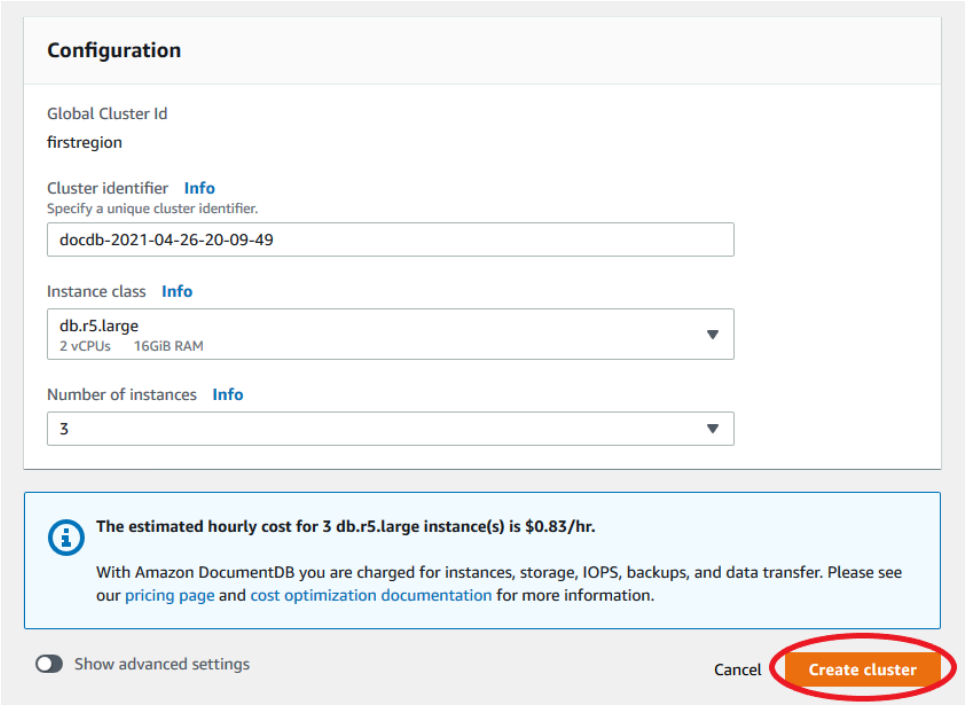
4. Selecione a lista suspensa de Ações e escolha Adicionar região.



5. Na página Adicionar uma Região da AWS, escolha a região secundária. Observe que você não pode escolher uma região que já tenha um cluster secundário para o mesmo cluster global. Além disso, não ela pode ser a mesma região que o cluster primário. Se essa for a primeira região que estiver adicionando, também será necessário especificar um identificador de cluster global de sua escolha.



6. Preencha os campos restantes para o cluster secundário na nova região e selecione Criar cluster. Depois de terminar de adicionar a região, será possível vê-la na lista de Clusters no Console de gerenciamento da AWS.



Usar o AWS CLI

- Use o comando da CLI `create-db-cluster` com o nome (`--global-cluster-identifier`) de seu cluster global. Para outros parâmetros, faça o seguinte:
  - Para`--region`, escolha uma Região da AWS diferente da sua região principal.
  - Escolha valores específicos para os parâmetros `--engine` e `--engine-version`.
  - Para um cluster criptografado, especifique seu primário Região da AWS como o `--source-region` para criptografia.

O exemplo a seguir cria um novo cluster do Amazon DocumentDB e o anexa ao cluster global como um cluster secundário somente leitura. Na última etapa, a instância é adicionada ao novo cluster.

No exemplo a seguir, substitua cada *`user input placeholder`* por suas próprias informações.

Para Linux, macOS ou 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
```

Usando um snapshot para seu cluster Amazon DocumentDB global

É possível restaurar um snapshot de um cluster do Amazon DocumentDB para usar como ponto de partida para seu cluster global. Para fazer isso, você deve restaurar o snapshot e criar um novo cluster. Isso servirá como o cluster principal do seu cluster global. Em seguida, adicione uma outra região ao cluster restaurado, transformando-o em um cluster global.

Gerenciar um cluster global do Amazon DocumentDB

Realize a maVocê realiza a maioria das operações de gerenciamento nos clusters individuais que compõem um cluster global.ioria das operações de gerenciamento nos clusters individuais que constituem um banco de dados global do . Ao selecionar Recursos relacionados ao grupo na página Clusters do console, você verá o cluster primário e os clusters secundários agrupados sob o cluster global associado.

A guia Configuração de um cluster global mostra Regiões da AWS onde os clusters estão sendo executados, a versão e o identificador global do cluster.

Tópicos

- Modificar um cluster global do Amazon DocumentDB
- Modificando parâmetros de um cluster global do Amazon DocumentDB
- Remover um cluster de um cluster global do Amazon DocumentDB
- Exclusão de um cluster global do Amazon DocumentDB
- Criação de um cluster sem cabeçalho do Amazon DocumentDB em uma região secundária

Modificar um cluster global do Amazon DocumentDB

A página Clusters na Console de gerenciamento da AWS lista todos os seus clusters globais, mostrando o cluster primário e os clusters secundários de cada um. O cluster global tem suas próprias definições de configuração. Especificamente, ele tem regiões associadas a seus clusters primários e secundários.

Quando você faz alterações no cluster global, você tem a chance de cancelar as alterações.

Quando você escolhe Continue, você confirma as alterações.

Modificando parâmetros de um cluster global do Amazon DocumentDB

É possível configurar os grupos de parâmetros do cluster independentemente para cada cluster dentro do cluster global. A maioria dos parâmetros funciona da mesma forma que para outros tipos de clusters do Amazon DocumentDB. Mantenha as configurações consistentes entre todos os clusters em um banco de dados global para evitar mudanças inesperadas de comportamento se você promover um cluster secundário como primário.

Por exemplo, use as mesmas configurações para os fusos horários e os conjuntos de caracteres a fim de evitar um comportamento inconsistente caso um cluster diferente assuma como o cluster primário.

Remover um cluster de um cluster global do Amazon DocumentDB

Há várias situações em que você pode querer remover clusters do seu cluster global. Por exemplo, você pode querer remover um cluster de um clster global se o cluster primário se tornar degradado ou isolado. Em seguida, ele se torna um cluster provisionado autônomo que pode ser usado para

criar um novo cluster global. Para saber mais, consulte Realizar um failover manual para um cluster global do Amazon DocumentDB.

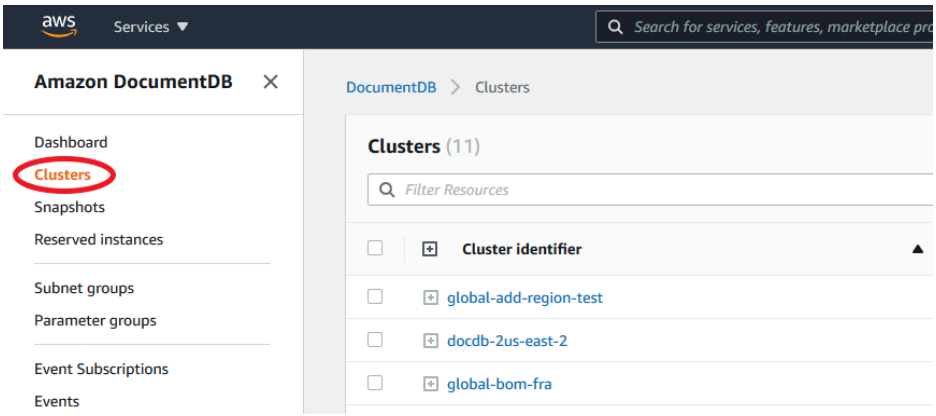
Você também pode querer remover clusters porque deseja excluir um cluster global que não é mais necessário. Não é possível excluir o cluster global até que você desanexe todos os clusters associados, deixando o primário por último. Para obter mais informações, consulte Exclusão de um cluster global do Amazon DocumentDB.

> **Note**
>
> Quando um cluster é separado do cluster global, ele não é mais sincronizado com o primário. Ele se torna um cluster provisionado independente com recursos completos. read/write Além disso, ele não está mais visível no console do Amazon DocumentDB. Ele só fica visível quando você seleciona a região no console em que o cluster estava localizado.

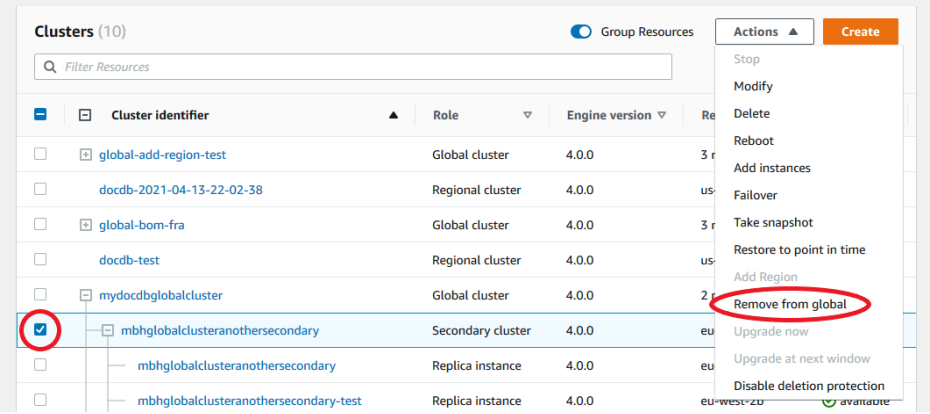
Você pode remover clusters do seu cluster global usando a API Console de gerenciamento da AWS AWS CLI, a ou a RDS.

Using the Console de gerenciamento da AWS

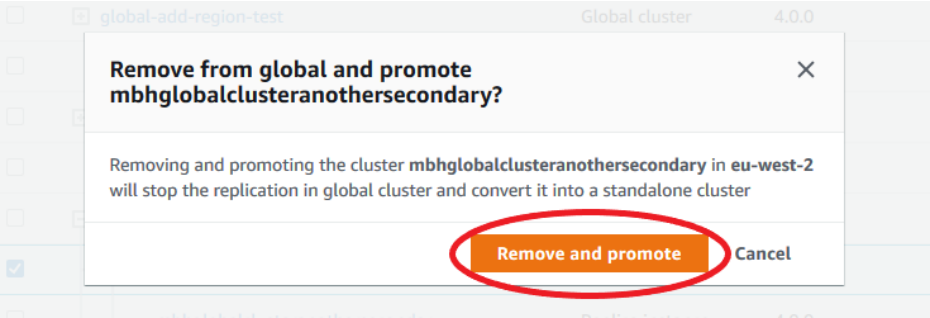
1. Faça login no Console de gerenciamento da AWS e navegue até o console do Amazon DocumentDB.
2. No painel de navegação esquerdo, escolha Clusters.



3. Expanda o cluster global para que você possa ver todos os clusters secundários. Selecione os clusters secundários que você deseja remover. Escolha Ações e, no menu suspenso, escolha Remover do global.



4. Será exibido um prompt solicitando a confirmação de que você deseja desconectar o secundário do cluster global. Escolha Remover e promover para remover o cluster do cluster global.



Agora esse cluster não está mais servindo como secundário e não está mais sincronizado com o cluster primário. É um cluster independente com read/write capacidade total.

Depois de remover ou excluir todos os clusters secundários, remova o cluster primário da mesma maneira. Não é possível desanexar ou remover o cluster primário do cluster global até que você tenha removido todos os clusters secundários. O cluster global pode permanecer na lista de clusters, com zero regiões e AZs. É possível excluir se não quiser mais usar esse cluster global.

Using the AWS CLI

Para remover um cluster de um cluster global, execute o comando CLI `remove-from-global-cluster` com os seguintes parâmetros:

- `--global-cluster-identifier`: o nome (identificador) do seu cluster global.
- `--db-cluster-identifier` — O nome de cada cluster a ser removido do cluster global.

Os exemplos a seguir removem primeiro um cluster secundário e depois o cluster primário de um cluster global.

Para Linux, macOS ou 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
```

Repita o `remove-from-global-cluster --db-cluster-identifier secondary_cluster_ARN` comando para cada região secundária em seu cluster 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
```

Repita o `remove-from-global-cluster --db-cluster-identifier secondary_cluster_ARN` comando para cada região secundária em seu cluster global.

Exclusão de um cluster global do Amazon DocumentDB

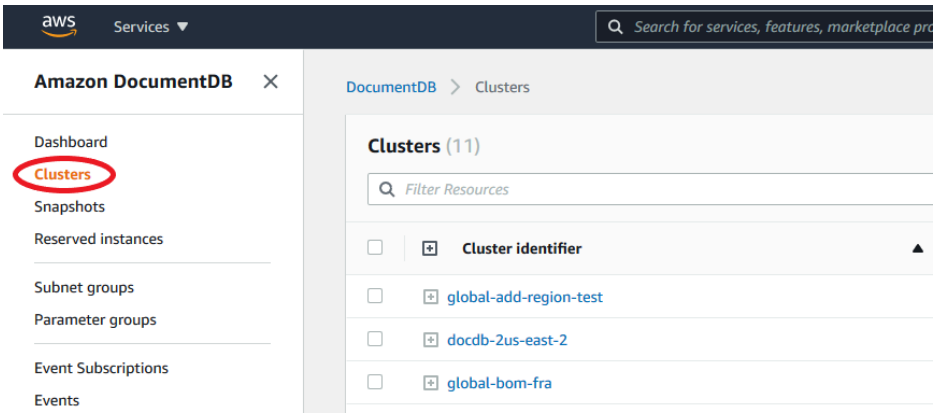
Para excluir um cluster global, faça o seguinte:

- Remova todos os clusters secundários do cluster global. Cada cluster se torna um cluster autônomo. Veja a seção anterior,Remover um cluster de um cluster global do Amazon DocumentDB.

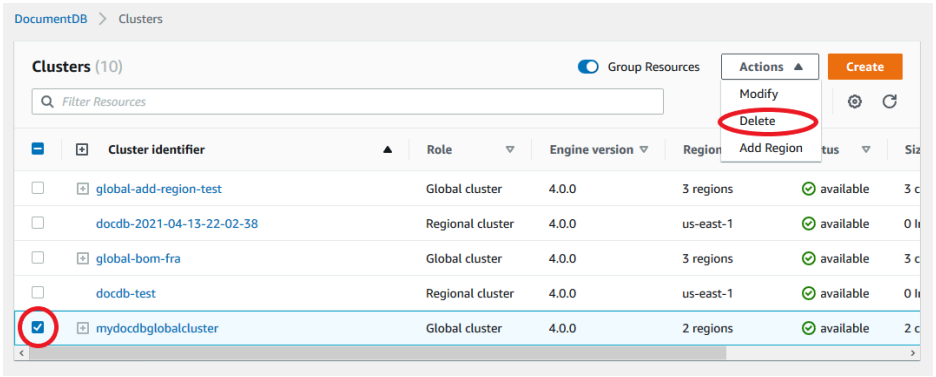
- Em cada cluster autônomo, exclua todas as réplicas.
- Remova o cluster primário do cluster global. Isso se torna um cluster autônomo.
- No cluster primário, primeiro exclua todas as réplicas e, em seguida, exclua a instância primária. A exclusão da instância primária do cluster recém-autônomo também remove, normalmente, o cluster e o cluster global.

Using the Console de gerenciamento da AWS

1. Faça login no Console de gerenciamento da AWS e navegue até o console do Amazon DocumentDB.
2. Escolha Clusters e encontre o cluster global que você deseja excluir.



3. Com seu cluster global selecionado, escolha Excluir no menu Ações.



Confirme se todos os clusters foram removidos do cluster global. O cluster global deve mostrar zero regiões e AZs e um tamanho de zero clusters. Se o cluster global contiver clusters, você ainda não poderá excluí-lo. Primeiro, você terá que seguir as instruções da etapa anterior, Remover um cluster de um cluster global do Amazon DocumentDB.

Using the AWS CLI

Para excluir um cluster global, execute o comando `delete-global-cluster` CLI com o nome do Região da AWS e o identificador do cluster global, conforme mostrado no exemplo a seguir.

Para Linux, macOS ou 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
```

Criação de um cluster sem cabeçalho do Amazon DocumentDB em uma região secundária

Embora um cluster global do Amazon DocumentDB exija pelo menos um cluster secundário Região da AWS diferente do primário, você pode usar uma configuração headless para o cluster secundário. Um cluster secundário sem periféricos do Amazon DocumentDB é aquele sem uma instância. Esse tipo de configuração pode reduzir as despesas de um cluster global. Em um cluster do Amazon DocumentDB, a computação e o armazenamento são desacoplados. Sem a instância, você não é cobrado pela computação, apenas pelo armazenamento. Se for configurado corretamente, o volume de armazenamento de um secundário sem cabeça é mantido em sincronia com o cluster primário.

Você adiciona o cluster secundário como normalmente faz ao criar um cluster global do Amazon DocumentDB. No entanto, depois que o cluster primário iniciar a replicação para o secundário, você excluirá a instância somente leitura do cluster secundário. Esse cluster secundário agora é considerado “sem cabeça” porque não tem mais uma instância. No entanto, o volume de armazenamento é mantido em sincronia com o cluster principal do Amazon DocumentDB.

> **Important**
>
> Use clusters sem cabeça somente se você puder tolerar Region-wide falhas por 15 minutos ou mais. A recuperação de uma Region-wide falha com um cluster secundário headless exige que você crie uma nova instância após o failover. Uma nova instância pode levar de 10 a 15 minutos para ficar disponível.

Como adicionar um cluster secundário headless ao seu cluster global

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon DocumentDB.
2. No painel de navegação esquerdo, escolha Clusters.
3. Escolha o cluster global que precisa de um cluster secundário. Certifique-se de que o cluster primário seja `Available`.
4. Em Ações, escolha Adicionar região.
5. Na página Adicionar uma região, escolha a região secundária.

   > **Note**
   >
   > Você não pode escolher uma região que já tenha um cluster secundário para o mesmo cluster global. Além disso, não ela pode ser a mesma região que o cluster primário.

6. Preencha os campos restantes do cluster secundário na nova Região . Essas são as mesmas opções de configuração de qualquer instância de cluster.
7. Adicione uma região. Depois que você terminar de adicionar a região ao seu cluster global, ela aparecerá `Clusters` na lista de Console de gerenciamento da AWS.
8. Verifique o status do cluster secundário e de sua instância de leitura antes de continuar, usando o Console de gerenciamento da AWS ou AWS CLI o. Aqui está um exemplo de comando se você usar o AWS CLI:

   ```
   $ aws docdb describe-db-clusters --db-cluster-identifier secondary-cluster-id --
   query '*[].[Status]' --output text
   ```

   Pode levar vários minutos para que o status de um cluster secundário recém-adicionado mude de criado para disponível. Quando o cluster estiver disponível, será possível excluir a instância do leitor.
9. Selecione a instância do leitor no cluster secundário e, em seguida, selecione Excluir.
10. Depois de excluir a instância do leitor, o cluster secundário continua fazendo parte do cluster global. Ele não deve ter nenhuma instância associada a ele.

> **Note**
>
> Você pode usar esse cluster secundário sem cabeçalho do Amazon DocumentDB para recuperar manualmente seu cluster global do Amazon DocumentDB de uma interrupção não planejada na região primária, caso tal interrupção ocorra.

Conexão com um cluster global do Amazon DocumentDB

A forma como você se conecta a um cluster global depende de se você precisa gravar no cluster ou ler a partir do cluster.

- Para solicitações ou consultas somente para leitura, conecte-se ao endpoint do leitor para o cluster em sua Região da AWS.
- Para executar instruções Data Manipulation Language (DML) ou Data Definition Language (DDL), conecte-se ao endpoint cluster do cluster primário. Esse endpoint pode estar em um local Região da AWS diferente do seu aplicativo.

Ao exibir um cluster global no console, é possível ver todos os endpoints de finalidade geral associados a todos os clusters.

A forma como você se conecta a um cluster global depende de se você precisa gravar no banco de dados ou ler a partir do banco de dados. Para operações de DDL, DML e leitura que você gostaria de atender na região primária, conecte-se ao seu cluster primário usando o endpoint do cluster no modo de conjunto de réplicas, com uma preferência de leitura de. `secondaryPreferred=true` Isso direciona o tráfego de gravação para a instância de gravação do cluster primário e o tráfego de leitura para a instância de réplica do cluster primário.

Para tráfego entre regiões, somente para leitura, conecte-se a um de seus clusters secundários usando o endpoint do cluster no modo de conjunto de réplicas. Como todas as instâncias são instâncias de réplica somente para leitura, especifique uma preferência de leitura diferente de `primary` (por exemplo, `secondarysecondaryPreferred`, ou). `nearest` Para minimizar latência, escolha o endpoint de leitor que está na sua região de ou na região de mais próxima de você.

Monitorar clusters globais do Amazon DocumentDB

O Amazon DocumentDB (com compatibilidade com o MongoDB) se integra CloudWatch para que você possa reunir e analisar métricas operacionais para seus clusters. Você pode monitorar essas

métricas usando o CloudWatch console, o console do Amazon DocumentDB, o AWS Command Line Interface (AWS CLI) ou a CloudWatch API.

Para monitorar um cluster global, use as CloudWatch métricas a seguir.

Métrica	Description
`GlobalClusterReplicatedWriteIO`	O número médio de I/O operações de gravação faturadas replicadas do volume do cluster no primário Região da AWS para o volume do cluster no secundário Região da AWS, relatado em intervalos de 5 minutos. O número de replicações `ReplicatedWriteIOs` em cada região secundária é igual ao número de regiões internas `VolumeWriteIOPs` realizadas pela região primária.
`GlobalClusterDataTransferBytes`	A quantidade de dados transferidos do cluster primário Região da AWS para o secundário Região da AWS, medida em bytes.
`GlobalClusterReplicationLag`	A quantidade de atraso, em milissegundos, ao replicar eventos de alteração do cluster primário Região da AWS para o secundário Região da AWS

Para obter mais informações sobre como visualizar essas métricas, consulte Visualização de CloudWatch dados.

Recuperação de desastres e clusters globais do Amazon DocumentDB

Tópicos

- Realizar um failover gerenciado para um cluster global do Amazon DocumentDB
- Realizar um failover manual para um cluster global do Amazon DocumentDB
- Como realizar uma transição para um cluster global do Amazon DocumentDB
- Desbloqueio da transição ou do failover de um cluster global

- Gerenciamento de RPOs para clusters globais do Amazon DocumentDB

Ao usar um cluster global, você pode se recuperar rapidamente de desastres, como falhas na região. A recuperação de desastres é normalmente medida usando valores de RTO e RPO.

- Objetivo de tempo de recuperação (RTO): tempo que um sistema leva para retornar a um estado de trabalho após um desastre. Em outras palavras, o RTO mede o tempo de inatividade. Para um cluster global, RTO em minutos.
- Objetivo de ponto de recuperação (RPO): quantidade de dados que podem ser perdidos (medidos no tempo). Para um cluster global, o RPO é normalmente medido em segundos.
- Para se recuperar de uma interrupção não planejada, você pode realizar um failover entre regiões para um dos secundários em seu cluster global. Quando seu cluster global tiver várias regiões secundárias, certifique-se de separar todas as regiões secundárias que você deseja promover como primárias. Em seguida, você promove uma dessas regiões secundárias para ser a nova primária Região da AWS. Finalmente, você cria novos clusters em cada uma das outras regiões secundárias e anexa esses clusters ao seu cluster global.

Realizar um failover gerenciado para um cluster global do Amazon DocumentDB

Essa abordagem destina-se à continuidade dos negócios no caso de um desastre regional real ou de uma interrupção completa do nível de serviço.

Durante um failover gerenciado, o cluster primário sofre failover para que você escolha a região secundária, enquanto a topologia de replicação existente do cluster global do Amazon DocumentDB é mantida. O cluster secundário escolhido promove um dos nós somente leitura ao status de gravador completo. Essa etapa permite que o cluster assuma a função de cluster primário. O banco de dados ficará indisponível por um curto período enquanto o cluster estiver assumindo a nova função. Os dados que não foram replicados do cluster primário antigo para o cluster secundário escolhido são perdidos quando esse secundário se torna o novo primário. O volume primário antigo faz o possível para tirar um snapshot antes de sincronizar com o novo primário, para que os dados não replicados sejam preservados no instantâneo.

> **Note**
>
> Você só poderá realizar um failover de cluster gerenciado entre regiões em um cluster global do Amazon DocumentDB se os clusters primário e todos os secundários tiverem as mesmas versões de mecanismo. Se as versões do mecanismo forem incompatíveis, será possível

executar o failover manualmente seguindo as etapas em Realizar um failover manual para um cluster global do Amazon DocumentDB.
Se as versões do mecanismo da região não corresponderem, o failover será bloqueado. Verifique se há atualizações pendentes e aplique-as para garantir que todas as versões do mecanismo da região correspondam e que o failover global do cluster seja desbloqueado. Para obter mais informações, consulte Desbloqueio da transição ou do failover de um cluster global.

Para minimizar a perda de dados, faça o seguinte antes de usar esse recurso:

- Deixe as aplicações off-line para evitar que as gravações sejam enviadas para o cluster primário do cluster global do Amazon DocumentDB.
- Verifique os tempos de atraso de todos os clusters secundários do Amazon DocumentDB. Escolher a região secundária com o menor atraso de replicação pode minimizar a perda de dados na atual região primária com falha. Verifique os tempos de atraso de todos os clusters secundários do Amazon DocumentDB no cluster global visualizando a `GlobalClusterReplicationLag` métrica na Amazon. CloudWatch Essa métrica informa a que distância (em milissegundos) está a replicação de um cluster secundário em relação ao cluster primário.

  Para obter mais informações sobre CloudWatch métricas para o Amazon DocumentDB, consulteMétricas do Amazon DocumentDB.

Durante um failover gerenciado, o cluster secundário escolhido é promovido à sua nova função como primário. No entanto, ele não herda as várias opções de configuração do cluster primário. Uma incompatibilidade na configuração pode levar a problemas de performance, incompatibilidades de workload e outros comportamentos anômalos. Para evitar esses problemas, resolva as diferenças entre seus clusters globais do Amazon DocumentDB para o seguinte:

- Configure um grupo de parâmetros de cluster Amazon DocumentDB para o novo primário, se necessário — Você pode configurar seus grupos de parâmetros de cluster do Amazon DocumentDB de forma independente para cada cluster em seu cluster global do Amazon DocumentDB. Por isso, quando você promove um cluster secundário para assumir a função primária, o grupo de parâmetros do secundário pode ser configurado diferentemente do primário. Em caso afirmativo, modifique o grupo de parâmetros do cluster secundário promovido para estar em conformidade com as configurações do cluster principal. Para saber como, consulte Modificando grupos de parâmetros de cluster do Amazon DocumentDB.

- Configure ferramentas e opções de monitoramento, como CloudWatch eventos e alarmes da Amazon — Configure o cluster promovido com a mesma capacidade de registro, alarmes etc. conforme necessário para o cluster global. Tal como acontece com os parameter groups, a configuração desses recursos não é herdada do primário no processo de failover. Algumas CloudWatch métricas, como atraso na replicação, só estão disponíveis para regiões secundárias. Assim, um failover altera a forma de visualizar essas métricas e definir alarmes para elas, e pode exigir alterações em qualquer painel predefinido. Para obter mais informações sobre clusters e monitoramento do Amazon DocumentDB, consulte Monitoramento e registro no Amazon DocumentDB.

Normalmente, o cluster secundário escolhido assume o perfil principal dentro de um minuto. Assim que o nó de gravação da nova região principal estiver disponível, será possível conectar suas aplicações a ele e retomar suas workloads. Depois que o Amazon DocumentDB promove o novo cluster primário, ele reconstrói automaticamente todos os clusters adicionais da região secundária.

Como os clusters globais do Amazon DocumentDB usam replicação assíncrona, o atraso na replicação em cada região secundária pode variar. O Amazon DocumentDB reconstrói essas regiões secundárias para ter exatamente os mesmos dados de um ponto no tempo que o novo cluster da região primária. A duração da tarefa de reconstrução completa pode levar de alguns minutos a várias horas, dependendo do tamanho do volume de armazenamento e da distância entre as regiões. Quando os clusters da região secundária terminam de ser reconstruídos com base na nova região primária, eles ficam disponíveis para acesso de leitura. Assim que o novo gravador principal for promovido e disponibilizado, o cluster da nova região primária poderá lidar com operações de leitura e gravação no cluster global do Amazon DocumentDB.

Para restaurar a topologia original do cluster global, o Amazon DocumentDB monitora a disponibilidade da antiga região primária. Assim que essa região estiver íntegra e disponível novamente, o Amazon DocumentDB a adicionará automaticamente ao cluster global como uma região secundária. Antes de criar o novo volume de armazenamento na antiga região primária, o Amazon DocumentDB tenta capturar um snapshot do volume de armazenamento antigo no ponto da falha. Ele faz isso para que você possa usá-lo para recuperar os dados perdidos. Se essa operação for bem-sucedida, o Amazon DocumentDB colocará esse snapshot chamado “rds:docdb-unplanned-global-failover-name-of-old-primary-" na seção de snapshot do. DB-cluster-timestamp Console de gerenciamento da AWS Você também pode ver esse snapshot listado nas informações retornadas pela operação de API `DescribeDBClusterSnapshots`.

> **Note**
>
> O snapshot do volume de armazenamento antigo é um snapshot do sistema sujeito ao período de retenção de backup configurado no antigo cluster primário. Para preservar esse instantâneo fora do período de retenção, você pode copiá-lo para salvá-lo como um snapshot manual. Para saber mais sobre como copiar snapshots, incluindo os preços, consulte Cópia de um snapshot de cluster.

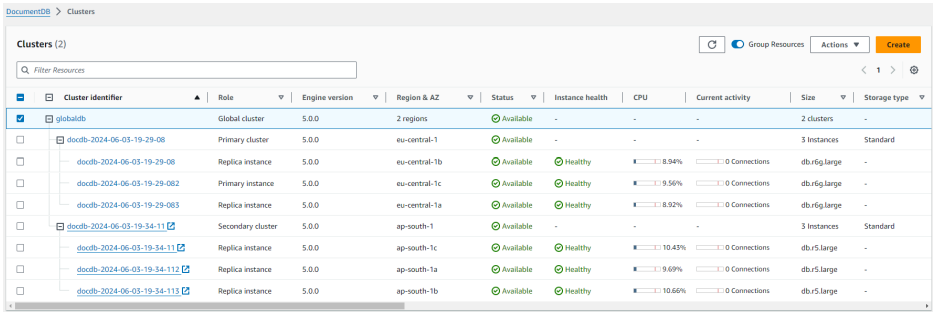
Depois que a topologia original for restaurada, será possível fazer failback do clustert global para a região primária original executando uma operação de transição quando fizer mais sentido para seus negócios e sua workload. Para isso, siga as etapas em Como realizar uma transição para um cluster global do Amazon DocumentDB.

Você pode fazer o failover do seu cluster global do Amazon DocumentDB usando a Console de gerenciamento da AWS AWS CLI, a ou a API do Amazon DocumentDB.

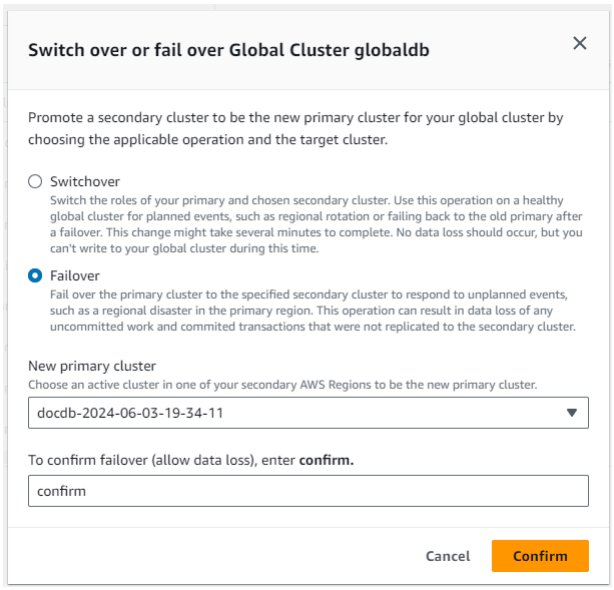
Using the Console de gerenciamento da AWS

Realize um failover gerenciado no seu cluster global do Amazon DocumentDB

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Clusters.
3. Localize e escolha o cluster global do Amazon DocumentDB do qual você deseja fazer failover.



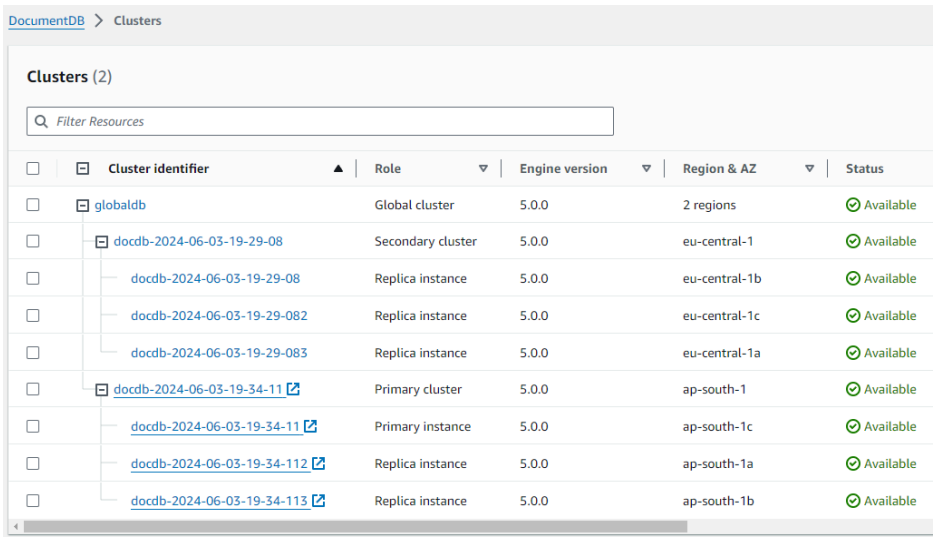
4. Escolha Transição ou failover no menu Ações.
5. Na caixa de diálogo exibida, escolha Failover e, em seguida, escolha o cluster secundário na lista suspensa do campo Novo cluster primário.



6. Digite “confirmar” no último campo. Depois, selecione Confirmar.

   O status do cluster primário muda para "Failing-over”. Essa condição deve levar aproximadamente um minuto. Durante esse período, o status do novo cluster primário mostra "Modificando…". Depois que o novo primário for promovido, ele mostrará "Disponível" e poderá veicular transações de leitura e gravação. As regiões secundárias, incluindo a primária antiga, mostrarão "Ressincronizando... "enquanto ele ressincroniza com o novo primário. Semelhante ao novo primário, ele só poderá servir a transação quando o status mudar para "Disponível".

7. Quando concluído, o cluster primário original se torna o cluster secundário. O cluster secundário selecionado se torna o cluster primário.



Using the AWS CLI

Realize um failover gerenciado no seu cluster global do Amazon DocumentDB

Execute o comando `failover-global-cluster` da CLI para fazer failover sobre seu cluster global do Amazon DocumentDB. Com o comando, passe valores para as opções a seguir:

- `--region`
- `--global-cluster-identifier`
- `--target-db-cluster-identifier`
- `--allow-data-loss`

Nos exemplos a seguir, substitua cada um *`user input placeholder`* pelas informações do seu cluster.

Para Linux, macOS ou 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
```

Realizar um failover manual para um cluster global do Amazon DocumentDB

Se um cluster inteiro em um Região da AWS ficar indisponível, você poderá promover que outro cluster no cluster global tenha read/write capacidade.

É possível ativar manualmente o mecanismo de failover do cluster global se um cluster em uma Região da AWS diferente seja uma opção melhor para ser o cluster primário. Por exemplo, você

pode aumentar a capacidade de um dos clusters secundários e promovê-lo para ser o cluster primário. Ou o equilíbrio de atividades entre eles Regiões da AWS pode mudar, de modo que a mudança do cluster primário para um diferente Região da AWS pode oferecer menor latência para operações de gravação.

O procedimento a seguir descreve o que fazer para promover um dos clusters secundários em um cluster global do Amazon DocumentDB.

Como promover um cluster secundário:

1. Pare de emitir instruções DML e outras operações de gravação no cluster primário durante Região da AWS a interrupção.
2. Identifique um cluster de um secundário Região da AWS para usar como um novo cluster primário. Se você tiver dois (ou mais) secundários Regiões da AWS em seu cluster global, escolha o cluster secundário com o menor tempo de atraso.
3. Desanexe o cluster secundário escolhido do banco de dados global.

   A remoção de um cluster secundário de um cluster global interrompe imediatamente a replicação do primário para esse secundário e o promove para um cluster provisionado independente com recursos completos. read/write Qualquer outro cluster secundário associado ao cluster primário na região com a interrupção ainda está disponível e pode aceitar chamadas do seu aplicativo. Eles também consomem recursos. Como você está recriando o cluster global para evitar problemas de cérebro dividido, entre outros, remova os outros clusters secundários antes de criar o cluster global nas etapas a seguir.

   Para obter as etapas detalhadas para desanexar, consulte Remover um cluster de um cluster global do Amazon DocumentDB.
4. Esse cluster se torna o cluster principal de um novo cluster global quando você começa a adicionar Regiões a ele, na próxima etapa.
5. Adicione um Região da AWS ao cluster. Quando você faz isso, o processo de replicação de primário para secundário começa.
6. Adicione mais Regiões da AWS conforme necessário para recriar a topologia necessária para dar suporte ao seu aplicativo. Certifique-se de que as gravações de aplicações sejam enviadas para o cluster correto antes, durante e depois de fazer alterações como essas, para evitar inconsistências de dados entre os clusters do cluster global (problemas de cérebro dividido).
7. Quando a interrupção for resolvida e você estiver pronto para reatribuir a Região da AWS original como o cluster primário, siga as mesmas etapas em sentido inverso:

8. Remova um dos clusters secundários do cluster global. Isso permitirá que ele atenda ao read/write tráfego.
9. Redirecione todo o tráfego de leitura para o cluster primário na Região da AWS original.
10. Adicione um Região da AWS para configurar um ou mais clusters secundários da Região da AWS mesma forma que antes.

Os clusters globais do Amazon DocumentDB podem ser gerenciados usando AWS SDKs, permitindo que você crie soluções para automatizar o processo global de failover de clusters para casos de uso de recuperação de desastres e planejamento de continuidade de negócios. Uma dessas soluções é disponibilizada para nossos clientes sob o licenciamento do Apache 2.0 e pode ser acessada em nosso repositório de ferramentas aqui. Essa solução utiliza o Amazon Route 53 para gerenciamento de endpoints e fornece AWS Lambda funções que podem ser acionadas com base em eventos apropriados.

Como realizar uma transição para um cluster global do Amazon DocumentDB

Ao usar transições, é possível alterar a região do cluster primário rotineiramente. Essa abordagem destina-se a ambientes controlados, como manutenção operacional e outros procedimentos operacionais planejados.

Há três casos de uso comuns para o uso de transições.

- Para requisitos de “alternância regional” impostos a setores específicos. Por exemplo, os regulamentos de serviços financeiros podem querer que os sistemas de nível 0 mudem para uma região diferente por vários meses a fim de garantir que os procedimentos de recuperação de desastres sejam executados regularmente.
- Para aplicações multirregionais ininterruptas (dia e noite). Por exemplo, uma empresa pode querer fornecer gravações de menor latência em diferentes regiões com base no horário comercial em diferentes fusos horários.
- Como um método sem perda de dados para fazer failback para a região primária original após um failover.

> **Note**
>
> As transições foram projetadas para serem usadas em um cluster global íntegro do Amazon DocumentDB. Para se recuperar de uma interrupção não planejada, siga o procedimento apropriado em Realizar um failover manual para um cluster global do Amazon DocumentDB.

Para realizar uma transição, todas as regiões secundárias devem estar executando exatamente a mesma versão de motor da primária. Se as versões do motor da região não corresponderem, a transição será bloqueada. Verifique se há atualizações pendentes e aplique-as para garantir que todas as versões do motor da região correspondam e que a transição global do cluster seja desbloqueada. Para obter mais informações, consulte Desbloqueio da transição ou do failover de um cluster global.

Durante uma transição, o Amazon DocumentDB muda o cluster primário para a região secundária escolhida e mantém a topologia de replicação existente do cluster global. Antes de iniciar o processo de transição, o Amazon DocumentDB espera que todos os clusters da região secundária estejam totalmente sincronizados com o cluster da região primária. Em seguida, o cluster de banco de dados na região primária torna-se somente leitura e o cluster secundário escolhido promove um dos respectivos nós somente leitura ao status de gravador completo. A promoção desse nó a um gravador permite que esse cluster secundário assuma a função de cluster primário. Como todos os clusters secundários foram sincronizados com o primário no início do processo, o novo primário continua as operações para o cluster global do Amazon DocumentDB sem perder dados. Seu banco de dados fica indisponível por um curto período enquanto os clusters primários e secundários selecionados estão assumindo suas novas funções.

Para otimizar a disponibilidade do aplicativo, faça o seguinte antes de usar esse recurso:

- Execute essa operação em períodos de pouca demanda ou em outro momento quando as gravações no cluster primário forem mínimas.
- Deixe as aplicações off-line para evitar que as gravações sejam enviadas para o cluster primário do cluster global do Amazon DocumentDB.
- Verifique os tempos de atraso de todos os clusters secundários do Amazon DocumentDB no cluster global visualizando a `GlobalClusterReplicationLag` métrica na Amazon. CloudWatch Essa métrica informa a que distância (em milissegundos) está a replicação de um cluster secundário em relação ao cluster primário. Esse valor é diretamente proporcional ao tempo que o Amazon DocumentDB levará para concluir a transição. Portanto, quanto maior o valor do atraso, mais tempo a transição levará.

  Para obter mais informações sobre CloudWatch métricas para o Amazon DocumentDB, consulteMétricas do Amazon DocumentDB.

Durante uma transição, o cluster de banco de dados secundário escolhido é promovido à nova função como primário. No entanto, ele não herda as várias opções de configuração do cluster de banco de dados primário. Uma incompatibilidade na configuração pode levar a problemas de performance, incompatibilidades de workload e outros comportamentos anômalos. Para evitar esses problemas, resolva as diferenças entre seus clusters globais do Amazon DocumentDB para o seguinte:

- Configurar um grupo de parâmetros do cluster do Amazon DocumentDB para o novo primário, se necessário: você pode configurar seus grupos de parâmetros de cluster do Amazon DocumentDB independentemente para cada cluster no cluster global do Amazon DocumentDB. Isso significa que quando você promove um cluster de banco de dados secundário para assumir a função primária, o parameter group do secundário pode ser configurado de forma diferente do que para o primário. Em caso afirmativo, modifique o parameter group do cluster de banco de dados secundário promovido para estar em conformidade com as configurações do cluster principal. Para saber como, consulte Gerenciando grupos de parâmetros de cluster do Amazon DocumentDB.
- Configure ferramentas e opções de monitoramento, como Amazon CloudWatch Events e alarmes — Configure o cluster promovido com a mesma capacidade de registro, alarmes etc. conforme necessário para o cluster global. Tal como ocorre com os grupos de parâmetros, a configuração desses recursos não é herdada do primário no processo de transição. Algumas CloudWatch métricas, como atraso na replicação, só estão disponíveis para regiões primárias. Assim, uma transição altera a forma de visualizar essas métricas e definir alarmes para elas, e pode exigir alterações em qualquer painel predefinido. Para obter mais informações, consulte Monitoramento e registro no Amazon DocumentDB.

> **Note**
>
> Normalmente, a transição de função pode levar alguns minutos.

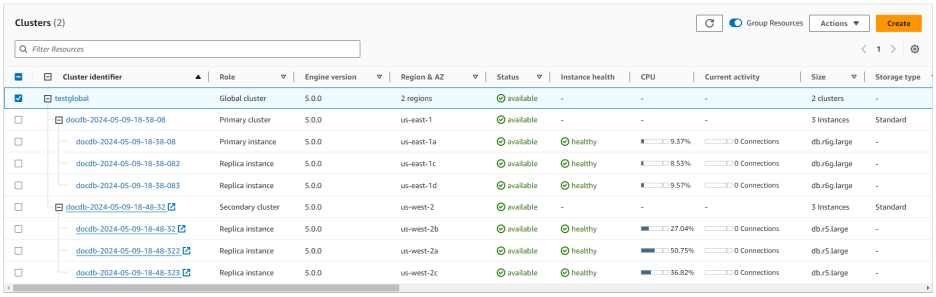
Quando o processo de transição for concluído, o cluster de do Amazon DocumentDB promovido poderá lidar com operações de gravação para o cluster global.

Você pode alternar seu cluster global do Amazon DocumentDB usando o Console de gerenciamento da AWS ou o AWS CLI:

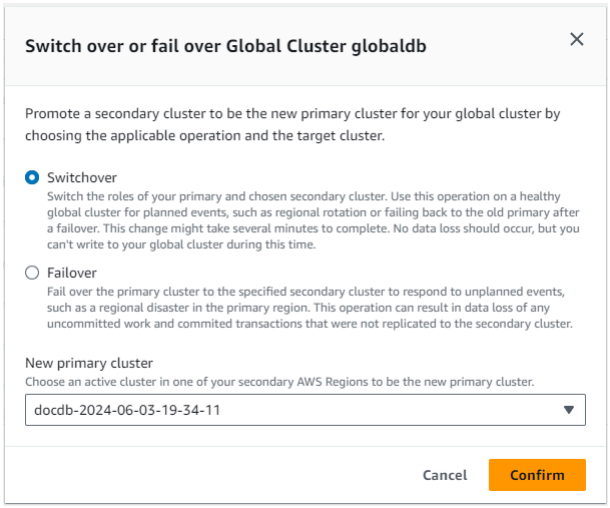
Using the Console de gerenciamento da AWS

Realize uma transição no seu cluster global do Amazon DocumentDB

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Clusters.
3. Localize e selecione o cluster global do Amazon DocumentDB do qual você deseja realizar a transição.

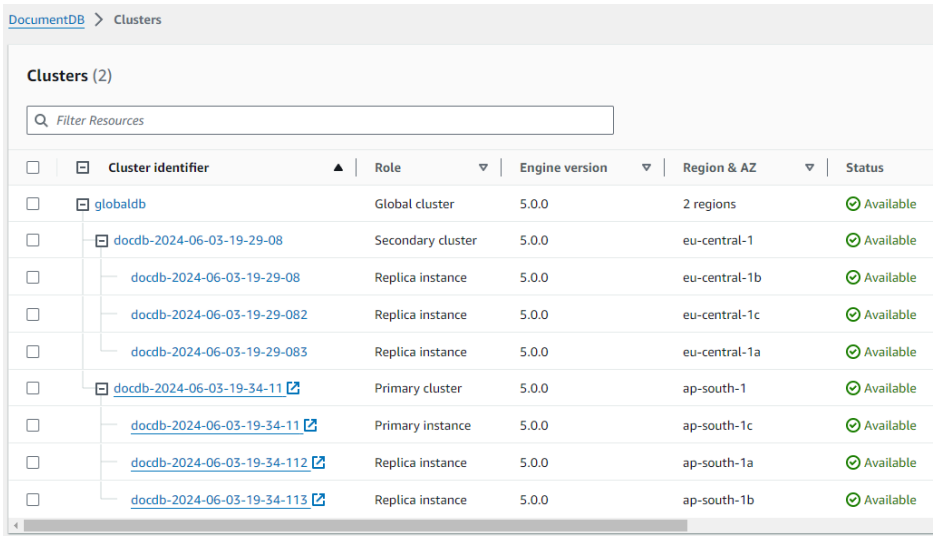


4. Escolha Transição ou failover no menu Ações.
5. Na caixa de diálogo exibida, escolha Transição e, em seguida, escolha o cluster secundário na lista suspensa do campo Novo cluster primário.



6. Escolha Confirmar.

   O status do cluster primário muda para "Switching-over". Essa condição deve levar aproximadamente três minutos. Durante esse período, o status de todos os clusters regionais é "Modificando...". Depois que as regiões forem sincronizadas e a nova primária for promovida, ela mostrará “Disponível” em todos os campos de status e poderá atender às transações.
7. Quando concluído, o cluster primário original se torna o cluster secundário. O cluster secundário selecionado se torna o cluster primário.



Using the AWS CLI

Realize uma transição no seu cluster global do Amazon DocumentDB

Execute o comando `switchover-global-cluster` da CLI para fazer a transição do seu cluster global do Amazon DocumentDB. Com o comando, passe valores para as opções a seguir:

- `--region`
- `--global-cluster-identifier`
- `--target-db-cluster-identifier`

Nos exemplos a seguir, substitua cada um *`user input placeholder`* pelas informações do seu cluster.

Para Linux, macOS ou 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
```

Desbloqueio da transição ou do failover de um cluster global

Regiões em diferentes versões de motor

As transições e os failovers globais do cluster são bloqueados quando nem todos os clusters regionais no cluster global estão na mesma versão de mecanismo. Se as versões não corresponderem, você poderá ver este erro ao chamar um switchover ou failover: O cluster de banco de dados de destino especificado está executando uma versão de mecanismo com um nível de patch diferente do cluster de banco de dados de origem. Aplique rotineiramente as versões mais recentes do mecanismo para manter seus clusters globais em um estado saudável.

Para resolver esse erro, atualize primeiro todas as regiões secundárias e depois a região primária para a mesma versão do mecanismo aplicando qualquer item de ação de manutenção pendente. Para visualizar os itens de ação de manutenção pendentes e aplicar as alterações necessárias para corrigir o problema, execute as instruções em uma das guias a seguir:

Using the Console de gerenciamento da AWS

Para desbloquear uma transição ou um failover global de clusters, você deve determinar se há alguma ação de manutenção pendente para seus clusters e aplicá-la. Siga estas etapas para visualizar e aplicar as ações de manutenção:

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Clusters.
3. Na tabela Clusters, localize seu cluster global na coluna Identificador do cluster. Em seu cluster global, anote cada cluster secundário e o cluster primário e execute as etapas a seguir para cada um.
4. Em cada cluster secundário:
    a. Se uma atualização estiver disponível para seu cluster, ela será indicada como Disponível, Obrigatória ou Próxima janela na coluna Manutenção.
    b. Para executar uma ação, escolha o cluster para mostrar seus detalhes e, em seguida, escolha Manutenção e backups. Os itens de manutenção pendentes são exibidos.

c. Em Descrição, se houver indicação de que “Uma nova atualização de manutenção está disponível”, selecione-a e escolha Aplicar agora.

5. Em seu cluster primário:

   a. Se uma atualização estiver disponível para seu cluster, ela será indicada como Disponível, Obrigatória ou Próxima janela na coluna Manutenção.

   b. Para executar uma ação, escolha o cluster para mostrar seus detalhes e, em seguida, escolha Manutenção e backups. Os itens de manutenção pendentes são exibidos.

   c. Em Descrição, se houver indicação de que “Uma nova atualização de manutenção está disponível”, selecione-a e escolha Aplicar agora.

Using the AWS CLI

Para desbloquear uma transição ou um failover de cluster global, você deve determinar se há alguma ação de manutenção pendente para o cluster e aplicá-la. Siga estas etapas para visualizar e aplicar ações de manutenção primeiro nos clusters secundários e depois no cluster primário do seu cluster global:

1. Execute o seguinte primeiro no cluster regional de cada região secundária e depois no cluster regional de regiões primárias.
2. Execute o comando `describe-pending-maintenance-actions` da CLI com a opção `--resource-identifier` para determinar se alguma ação de manutenção está disponível para seu cluster regional do Amazon DocumentDB.

   Nos exemplos a seguir, substitua cada um *`user input placeholder`* pelas informações do seu cluster.

   Para Linux, macOS ou 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
```

O resultado deve ser semelhante a este:

```
{
    "PendingMaintenanceActions": [
        {
            "ResourceIdentifier": "arn:aws:rds:us-
east-1:001234567890:cluster:docdb-2025-03-27-19-21-15",
            "PendingMaintenanceActionDetails": [
                {
                    "Action": "system-update",
                    "CurrentApplyDate": "2025-04-11T03:01:00Z",
                    "Description": "db-version-upgrade",
                    "ForcedApplyDate": "2025-06-18T03:01:00Z",
                    "AutoAppliedAfterDate": "2025-05-11T03:01:00Z"
                    "OptInStatus": "pending"
                }
            ]
        }
    ]
}
```

3. Se for necessária uma ação de manutenção, execute o comando apply-pending-maintenance-action da CLI com as opções a seguir:

   - `--resource-identifier`
   - `--apply-action`
   - `--opt-in-type`
   - `--region`

   Nos exemplos a seguir, substitua cada um *user input placeholder* pelas informações do seu cluster.

   Para Linux, macOS ou 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. Depois que a ação de manutenção for concluída, execute o comando `describe-pending-maintenance-actions` novamente para garantir que não haja outras ações pendentes para seu cluster.

   O resultado que você deseja é:

```
{
    "PendingMaintenanceActions": []
}
```

Using the Amazon DocumentDB API

Para desbloquear uma transição ou um failover de cluster global, você deve determinar se há alguma ação de manutenção pendente para o cluster e aplicá-la. Use as APIs a seguir para visualizar e aplicar ações de manutenção:

1. Execute o seguinte primeiro no cluster regional de cada região secundária e depois no cluster regional de regiões primárias.
2. Chame a API PendingMaintenanceAction para determinar se alguma ação de manutenção está disponível para seu cluster global do Amazon DocumentDB.
3. Aplique todas as alterações chamando a API ApplyPendingMaintenanceAction.

Alterações pendentes no cluster de destino

Os switchovers e os failovers também são bloqueados quando o cluster secundário de destino tem uma alteração programada que ainda não foi aplicada, como uma modificação ou ação de manutenção que você solicitou para a próxima janela de manutenção. Nesse caso, você pode ver esse erro ao chamar um switchover ou failover: você não pode fazer o failover para o cluster com ARN *arn:aws:rds:us-east-1:001234567890:cluster:docdb-2025-03-27-19-21-15* porque ele está sendo modificado ou porque as modificações estão pendentes. Tente novamente quando o cluster estiver disponível. Para desbloquear a operação, remova a alteração agendada no cluster de destino. As etapas dependem do tipo de alteração. Depois que a alteração agendada for removida e o status do cluster retorna`available`, tente novamente a alternância ou o failover.

Cancelar uma ação de manutenção programada

Você não pode cancelar uma ação de manutenção programada (desfazer um opt-in) do Console de gerenciamento da AWS, então use o. AWS CLI

Using the AWS CLI

1. Execute o comando CLI describe-pending-maintenance-actions com a `--resource-identifier` opção e procure uma ação de manutenção que seja. `OptInStatus` `next-maintenance`

   Nos exemplos a seguir, substitua cada um *`user input placeholder`* pelas informações do seu cluster.

   Para Linux, macOS ou 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
   ```

O resultado é semelhante ao seguinte. Anote o `Action` valor (`os-upgrade`neste exemplo), que você usa na próxima etapa.

```
{
    "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. Cancele a ação agendada executando o comando da CLI apply-pending-maintenance-action com`--opt-in-type undo-opt-in`, passando o valor da etapa anterior para o. `Action` `--apply-action`

   Para Linux, macOS ou 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
```

Na resposta, a ação de manutenção não tem mais um`OptInStatus`, o que confirma que a aceitação agendada foi cancelada.

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

Reverter uma modificação programada do cluster

Using the AWS CLI

1. Verifique as modificações programadas executando o comando CLI describe-db-clusters e inspecionando o campo de saída. `PendingModifiedValues`

> **Note**
>
> Use a CLI do Amazon RDS (`aws rds`) para esse comando, não a CLI do Amazon DocumentDB (`aws docdb`), porque a `describe-db-clusters` API do Amazon DocumentDB não retorna um campo no nível do cluster. `PendingModifiedValues`

Nos exemplos a seguir, substitua cada um *`user input placeholder`* pelas informações do seu cluster.

Para Linux, macOS ou 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
```

O resultado mostra as mudanças programadas. Neste exemplo, uma alteração no nome do cluster (`DBClusterIdentifier`) está pendente, de `docdb-2025-03-27-19-21-15` para`docdb-new-name`.

```
{
    "DBClusterIdentifier": "docdb-new-name"
}
```

2. Reverta a alteração executando o comando CLI modify-db-cluster com`--apply-immediately`, redefinindo o valor modificado para seu valor original. Este exemplo reverte a alteração de nome pendente `--new-db-cluster-identifier` retornando ao nome atual do cluster.

   Para Linux, macOS ou 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
```

Gerenciamento de RPOs para clusters globais do Amazon DocumentDB

Com um cluster global do Amazon DocumentDB, você pode gerenciar o objetivo do ponto de recuperação (RPO) usando o `global_db_rpo` parâmetro. O RPO representa a quantidade máxima de dados que podem ser perdidos em caso de interrupção.

Quando você define um RPO para seu cluster global do Amazon DocumentDB, o Amazon DocumentDB monitora o tempo de atraso do RPO de todos os clusters secundários. Esse monitoramento garante que pelo menos um cluster secundário permaneça dentro da janela de RPO de destino.

A configuração de RPO controla como o Amazon DocumentDB gerencia transações de gravação no cluster primário para limitar a possível perda de dados se ocorrer um failover. O Amazon DocumentDB avalia os tempos de atraso de RPO e RPO para confirmar (ou bloquear) transações no primário da seguinte forma:

- Compromete a transação se pelo menos um cluster de banco de dados secundário tiver um tempo de atraso de RPO menor que o RPO.
- Bloqueia a transação se todos os clusters de banco de dados secundários tiverem tempos de atraso de RPO maiores que o RPO.

Em outras palavras, se todos os clusters secundários estiverem atrás do RPO de destino, o Amazon DocumentDB pausa as transações no cluster primário. O Amazon DocumentDB retoma e confirma as transações pausadas assim que o tempo de atraso de pelo menos um cluster de banco de dados secundário cair abaixo do RPO. O resultado é que nenhuma transação pode confirmar até que o RPO seja atendido.

O parâmetro `global_db_rpo` é dinâmico. Se você decidir que não quer que todas as transações de gravação sejam proteladas até que o atraso diminua o suficiente, você pode redefini-lo rapidamente. Nesse caso, o Amazon DocumentDB aplica a alteração após um pequeno atraso.

> **Important**
>
> Em um banco de dados global com apenas duas AWS regiões, recomendamos manter o valor padrão do `global_db_rpo` parâmetro no grupo de parâmetros da região secundária. Caso contrário, realizar um failover devido à perda da AWS região primária poderia fazer com que o Amazon DocumentDB pausasse as transações. Em vez disso, espere até que o

Amazon DocumentDB conclua a reconstrução do cluster na antiga AWS região que falhou antes de alterar esse parâmetro para impor um RPO máximo.

Tópicos

- Configurar o objetivo de ponto de recuperação
- Visualizar o objetivo de ponto de recuperação
- Desabilitar o objetivo de ponto de recuperação

Configurar o objetivo de ponto de recuperação

O `global_db_rpo` parâmetro controla a configuração de RPO para um banco de dados Amazon DocumentDB. Os valores válidos variam de 20 segundos a 2.147.483.647 segundos (68 anos). Escolha um valor realista para atender às suas necessidades comerciais. Por exemplo, você pode querer permitir até 10 minutos para o seu RPO, caso em que você define o valor para 600.

Você pode definir esse valor para seu cluster global do Amazon DocumentDB usando a Console de gerenciamento da AWS, a ou a AWS CLI API do Amazon DocumentDB.

Using the Console de gerenciamento da AWS

Como definir o RPO

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. Escolha o cluster primário do seu cluster global Amazon DocumentDB e abra a guia Configuração para encontrar seu grupo de parâmetros de cluster de banco de dados.

   Os grupos de parâmetros não podem ser editados diretamente. Em vez disso, você pode fazer o seguinte:

   - Crie um parameter group de cluster de banco de dados personalizado usando o parameter group padrão apropriado como o ponto de partida.
   - Em seu grupo de parâmetros de cluster de banco de dados personalizado, defina o valor do parâmetro global_db_rpo para atender ao seu caso de uso. Os valores válidos são de 20 segundos até o valor inteiro máximo de 2.147.483.647 (68 anos).
   - Aplique o grupo de parâmetros de cluster de banco de dados modificado ao seu cluster de banco de dados Amazon DocumentDB.

Para obter mais informações sobre como modificar grupos de parâmetros de clusters de banco de dados, consulteModificando grupos de parâmetros de cluster do Amazon DocumentDB.

Using the AWS CLI

Para definir o parâmetro `global_db_rpo`, use o comando CLI modify-db-cluster-parameter-group. No comando, especifique o nome do grupo de parâmetros do cluster primário e os valores do parâmetro RPO.

O exemplo a seguir define o RPO como 600 segundos (10 minutos) para o grupo de parâmetros de cluster de banco de dados primário chamado `my_custom_global_parameter_group`.

Para Linux, macOS ou 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 o `global_db_rpo` parâmetro, use a operação ModifyDBClusterParameterGroup da API.

Visualizar o objetivo de ponto de recuperação

O objetivo de ponto de recuperação (RPO) de um cluster global é armazenado no `global_db_rpo` parâmetro de cada cluster de banco de dados.

Você pode usar a CLI para visualizar o `global_db_rpo` parâmetro de um cluster de banco de dados Amazon DocumentDB. Use a `--query` opção para retornar somente o `global_db_rpo` parâmetro do grupo de parâmetros.

Para Linux, macOS ou 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']"
```

Esse comando retorna uma saída semelhante à seguinte.

```
[
    {
        "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 obter mais informações sobre a visualização dos parâmetros do grupo de parâmetros do cluster, consulteGerenciando grupos de parâmetros de cluster do Amazon DocumentDB.

Desabilitar o objetivo de ponto de recuperação

Para desabilitar o RPO, redefina o parâmetro `global_db_rpo`. Você pode redefinir parâmetros usando a Console de gerenciamento da AWS AWS CLI, a ou a API Amazon DocumentDB.

Using the Console de gerenciamento da AWS

Como desabilitar o RPO

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.

2. No painel de navegação, selecione Parameter groups.
3. Na lista, escolha o grupo de parâmetros de cluster do banco de dados primário.
4. Escolha o botão de rádio ao lado do parâmetro global_db_rpo.
5. Escolha Redefinir para o padrão e confirme.

Para obter mais informações sobre como redefinir um parâmetro com o console, consulte Modificando grupos de parâmetros de cluster do Amazon DocumentDB.

Using the AWS CLI

Para redefinir o parâmetro `global_db_rpo`, use o comando reset-db-cluster-parameter-group .

Para Linux, macOS ou 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 redefinir o `global_db_rpo` parâmetro, use a operação ResetDBClusterParameterGroup da API.

Gerenciar clusters do Amazon DocumentDB

Para gerenciar um cluster do Amazon DocumentDB, é necessário ter uma política do IAM com as permissões de ambiente de gerenciamento apropriadas do Amazon DocumentDB. Essas permissões permitem criar, modificar e excluir clusters e instâncias. A política `AmazonDocDBFullAccess` fornece todas as permissões necessárias para administrar um cluster do Amazon DocumentDB.

Os tópicos a seguir mostram como executar várias tarefas ao trabalhar com clusters do Amazon DocumentDB, inclusive criação, exclusão, modificação, conexão e visualização.

Tópicos

- Noções básicas de clusters
- Configurações do cluster Amazon DocumentDB
- Configurações de armazenamento de cluster do Amazon DocumentDB
- Determinar o status de um cluster
- Ciclo de vida do cluster do Amazon DocumentDB
- Escalar clusters do Amazon DocumentDB
- Clonar um volume para um cluster Amazon DocumentDB
- Entender a tolerância a falhas de cluster do Amazon DocumentDB

Noções básicas de clusters

O Amazon DocumentDB separa computação de armazenamento, transfere replicação de dados e backup para o volume do cluster. Um volume de cluster fornece uma camada de armazenamento durável, confiável e altamente disponível, que replica dados de seis maneiras em três Zonas de Disponibilidade. As réplicas permitem maior disponibilidade de dados e leitura em escala. Cada cluster pode ter sua escala aumentada verticalmente em até 15 réplicas.

Substantivo	Description	Operações de API (verbos)

Cluster	Consiste em uma ou mais instâncias e em um volume de armazenamento de cluster que gerencia os dados para essas instâncias.	`create-db-cluster`
		`delete-db-cluster`
		`describe-db-clusters`
		`modify-db-cluster`

Instância	A leitura e a gravação de dados no volume de armazenamento do cluster são feitas por meio de instâncias. Em um determina do cluster, há dois tipos de instâncias: principal e de réplica. Um cluster sempre	`create-db-instance`
		`delete-db-instance`
		`describe-db-instan ces`
		`modify-db-instance`

Substantivo	Description	Operações de API (verbos)

	tem uma instância principal e pode ter entre 0 e 15 réplicas.	`describe-orderable-db-instance-options`
		
		`reboot-db-instance`

Volume do cluster	Um volume de armazenamento de banco de dados virtual que abrange três Zonas de Disponibilidade, no qual cada Zona de Disponibilidade conta com duas cópias dos dados do cluster.	N/A
Instância principal	Oferece suporte a operações de leitura e gravação, além de realizar todas as modificações de dados no volume do cluster. Cada cluster tem uma instância principal.	N/A
Instância de réplica	É compatível apenas com operações de leitura. Cada cluster do Amazon DocumentDB pode ter até 15 instâncias de réplica além da instância principal. Várias réplicas distribuem workloads de leitura. Ao localizar réplicas em Zonas de Disponibilidade separadas, você também pode aumentar a disponibilidade do banco de dados.	N/A

Substantivo	Description	Operações de API (verbos)
Endpoint do cluster	Um endpoint para um cluster do Amazon DocumentDB que se conecta à instância principal atual do mesmo. Cada cluster do Amazon DocumentDB tem um endpoint de cluster e uma instância principal.	N/A
Endpoint de leitor	Um endpoint para um cluster do Amazon DocumentDB que se conecta a uma das réplicas disponíveis para esse cluster. Cada cluster do Amazon DocumentDB tem um endpoint de leitor. Se houver mais de uma réplica, o endpoint de leitor direcionará cada solicitação de conexão para uma das réplicas do Amazon DocumentDB.	N/A
Endpoint da instância	Um endpoint para uma instância em um cluster do Amazon DocumentDB que se conecta a uma instância específica. Cada instância em um cluster, independente do tipo, tem seu próprio endpoint de instância exclusivo.	N/A

Configurações do cluster Amazon DocumentDB

Ao criar ou modificar um cluster, é importante entender quais parâmetros são imutáveis e quais podem ser modificados após o cluster ser criado. A tabela a seguir lista todos as configurações ou os parâmetros que são específicos de um cluster. Conforme especificado na tabela, alguns são modificáveis, outros não.

> **Note**
>
> Essas configurações não devem ser confundidas com grupos de parâmetros de cluster do Amazon DocumentDB e seus parâmetros. Para obter mais informações sobre parameter groups de cluster, consulte Gerenciando grupos de parâmetros de cluster do Amazon DocumentDB.

Parâmetro	Modificável	Observações
`DBClusterIdentifier`	Sim	Restrições de nomenclatura: • O comprimento é de [1 a 63] letras, números ou hifens. • O primeiro caractere deve ser uma letra. • Não pode terminar com um hífen ou conter dois hifens consecutivos. • Deve ser exclusivo para todos os clusters do Amazon RDS, Amazon Neptune e Amazon DocumentDB por Conta da AWS região.
`Engine`	Não	Deve ser `docdb`.
`BackupRetentionPeriod`	Sim	Defina um período entre [1 e 35] dias.
`DBClusterParameterGroupName`	Sim	Restrições de nomenclatura: • O comprimento é de [1–255] caracteres alfanuméricos. • O primeiro caractere deve ser uma letra.

Parâmetro	Modificável	Observações
		• Não pode terminar com um hífen ou conter dois hifens consecutivos.
DBSubnetGroupName	Não	Depois que um cluster for criado, você não poderá modificar a sub-rede do cluster.
EngineVersion	Não	O valor pode ser `5.0.0` (padrão), `4.0.0` ou `3.6.0`.
KmsKeyId	Não	Se você optar por criptografar seu cluster, não poderá alterar a AWS KMS chave usada para criptografar seu cluster.
MasterUsername	Não	Depois que um cluster for criado, você não poderá modificar o `MasterUsername` . Restrições de nomenclatura: • O comprimento é de [1–63] caracteres alfanuméricos. • O primeiro caractere deve ser uma letra. • Não pode ser uma palavra reservada pelo mecanismo de banco de dados.

MasterUserPassword	Sim	Restrições:
		
		• O comprimento é de [8–100] caracteres ASCII imprimíveis.
		• Pode usar quaisquer caracteres ASCII imprimíveis, exceto:
		  • **/** (barra)
		  • **"** (aspas duplas)
		  • **@** (arroba)

Port	Sim	O número da porta se aplica a todas as instâncias no cluster.

Parâmetro	Modificável	Observações
PreferredBackupWindow	Sim	
PreferredMaintenanceWindow	Sim	
StorageEncrypted	Não	Se você optar por criptografar seu cluster, ele não poderá ser descriptografado.
StorageType	Sim	O tipo de armazenamento para o cluster de banco de dados: Standard (`standard`) ou I/O-Optimized (`iopt1`). Padrão: `standard` Esse parâmetro pode ser configurado com `CreateDBCluster` e `ModifyDBCluster` . Para obter mais informações, consulte Configurações de armazenamento de cluster do Amazon DocumentDB.
Tags	Sim	
VpcSecurityGroupIds	Não	Depois que um cluster for criado, você não poderá modificar a VPC em que o cluster reside.

Configurações de armazenamento de cluster do Amazon DocumentDB

A partir do Amazon DocumentDB 5.0, os clusters baseados em instâncias oferecem suporte a dois tipos de configurações de armazenamento:

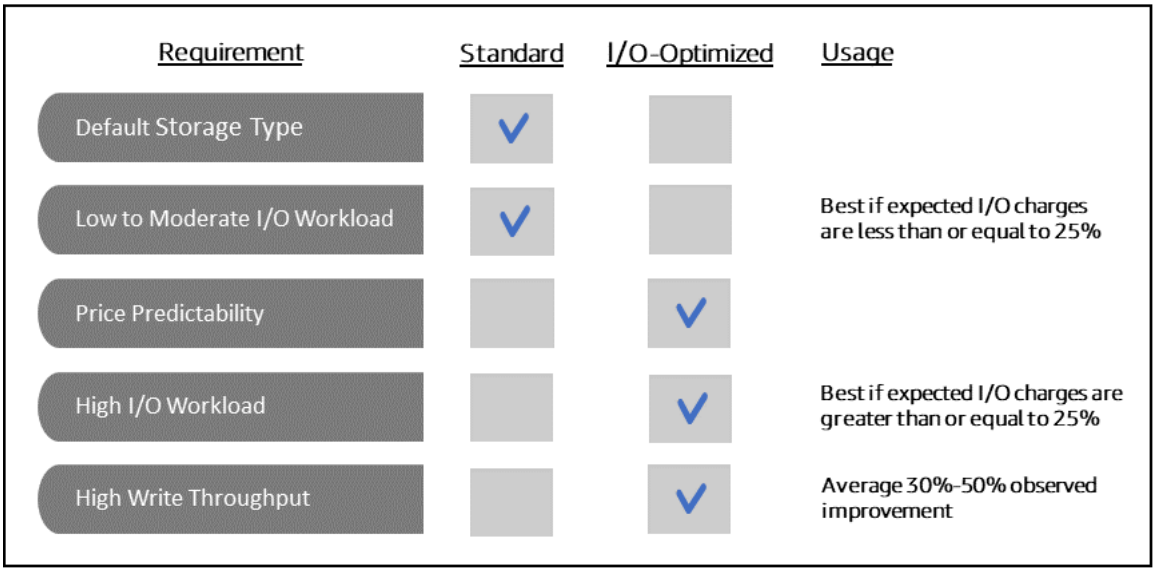
- Armazenamento padrão Amazon DocumentDB: projetado para clientes com I/O consumo baixo a moderado. Se você espera que seus I/O custos sejam inferiores a 25% do total do seu cluster Amazon DocumentDB, essa opção pode ser ideal para você. Com a configuração de armazenamento padrão do Amazon DocumentDB, você é cobrado com I/O base no pagamento

por solicitação, além das cobranças de instância e armazenamento. Isso significa que seu faturamento pode variar de um ciclo para outro com base no uso. A configuração é personalizada para acomodar as I/O demandas flutuantes de sua aplicação.

- I/O-optimized Armazenamento Amazon DocumentDB: projetado para clientes que priorizam a previsibilidade de preços ou têm aplicativos intensivos. I/O A I/O-optimized configuração oferece desempenho aprimorado, maior taxa de transferência e latência reduzida para clientes com cargas de I/O trabalho intensivas. Se você espera que seus I/O custos excedam 25% dos custos totais do cluster Amazon DocumentDB, essa opção oferece melhor relação preço/desempenho. Com a configuração de I/O-optimized armazenamento do Amazon DocumentDB, você não será cobrado com base nas I/O operações, garantindo custos previsíveis em cada ciclo de cobrança. A configuração estabiliza os custos e, ao mesmo tempo, melhora a performance.

Você pode migrar seus clusters de banco de dados existentes uma vez a cada 30 dias para o I/O-optimized armazenamento Amazon DocumentDB. É possível voltar para o armazenamento padrão do Amazon DocumentDB a qualquer momento. A próxima data para modificar a configuração de armazenamento I/O-optimized pode ser rastreada com o `describe-db-clusters` comando usando AWS CLI ou por meio da Console de gerenciamento da AWS página de configuração do cluster.

Você pode criar um novo cluster de banco de dados, incluindo a I/O-optimized configuração do Amazon DocumentDB, ou converter seus clusters de banco de dados existentes com alguns cliques no Console de gerenciamento da AWS, uma única alteração de parâmetro no AWS Command Line Interface (AWS CLI) ou por meio AWS de SDKs. Não é necessário tempo de inatividade ou reinicialização das instâncias durante ou após a modificação da configuração de armazenamento.

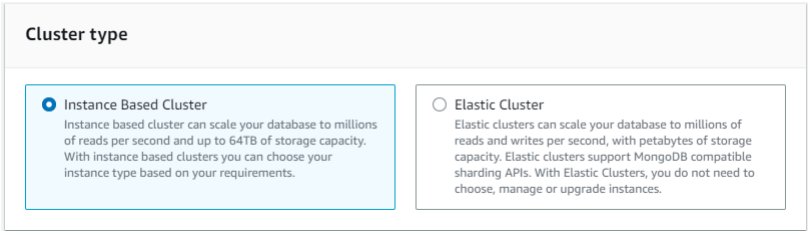


Criação de um I/O-optimized cluster

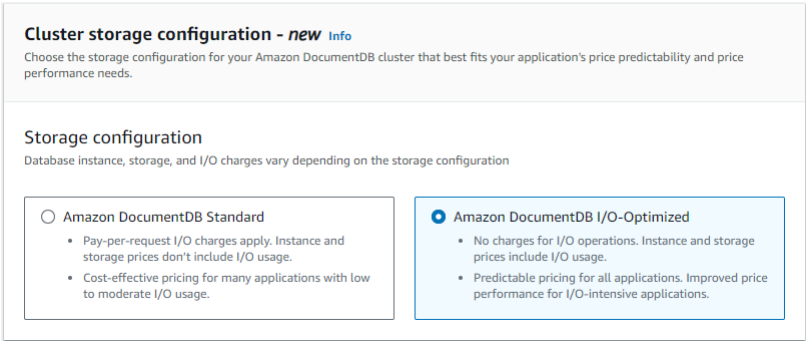
Using the Console de gerenciamento da AWS

Para criar ou modificar um I/O-optimized cluster usando o Console de gerenciamento da AWS:

1. No console de gerenciamento do Amazon DocumentDB, em Clusters, escolha Criar ou selecione o cluster e escolha Ações e, em seguida, escolha Modificar.
2. Se você estiver criando um novo cluster, escolha Cluster baseado em instância na seção Tipo de cluster (essa é a opção padrão).



3. Na seção Configuração, em Configuração de armazenamento em cluster, escolha Amazon DocumentDB I/O-Optimized.



4. Conclua a criação ou modificação do cluster e escolha Criar cluster ou Modificar cluster.

   Para obter o processo completo de criação de cluster, consulte Criação de um cluster e uma instância primária usando o Console de gerenciamento da AWS.

   Para obter o processo completo de modificação de cluster, consulte Modificar um cluster do Amazon DocumentDB.

Using the AWS CLI

Para criar um I/O-optimized cluster usando o AWS CLI:

Nos exemplos a seguir, substitua cada um *user input placeholder* por suas próprias informações.

Para Linux, macOS ou 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
```

Análise de custos para determinar a configuração de armazenamento

Com o Amazon DocumentDB, você tem a flexibilidade de escolher a configuração de armazenamento para todos os seus clusters de banco de dados. Para alocar adequadamente seus clusters entre padrão e I/O-optimized, você pode rastrear seus custos do Amazon DocumentDB em termos de cluster. Para fazer isso, você pode adicionar tags aos clusters existentes, habilitar a marcação de alocação de custos em seu painel do Gerenciamento de Faturamento e Custos da AWS e analisar os custos de um determinado cluster no AWS Cost Explorer Service. Para obter informações sobre análise de custos, consulte nosso blog Usar tags de alocação de custos.

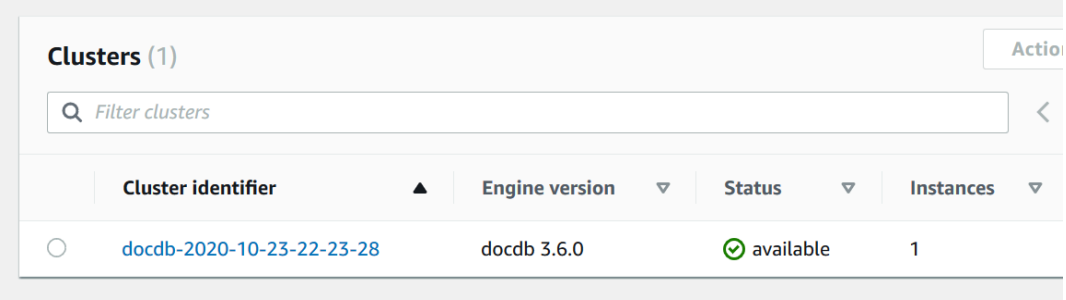
Determinar o status de um cluster

Você pode determinar o status de um cluster usando o Console de gerenciamento da AWS ou AWS CLI.

Using the Console de gerenciamento da AWS

Use o procedimento a seguir para ver o status do seu cluster Amazon DocumentDB usando o Console de gerenciamento da AWS

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Clusters.
3. Na coluna Identificador de cluster, encontre o nome do cluster desejado. Depois, para descobrir o status do cluster, leia essa linha até a coluna Status, conforme mostrado abaixo.



Using the AWS CLI

Use a operação `describe-db-clusters` para ver o status do cluster do Amazon DocumentDB utilizando AWS CLI.

O código a seguir descobre o status do cluster `sample-cluster`.

Para Linux, macOS ou 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]'
```

A saída dessa operação é semelhante ao seguinte (formato JSON).

```
[
    [
        "sample-cluster",
        "available"
    ]
]
```

Ciclo de vida do cluster do Amazon DocumentDB

O ciclo de vida de um cluster do Amazon DocumentDB inclui a criação, a descrição, a modificação e a exclusão do cluster. Esta seção fornece informações sobre como concluir esses processos.

Tópicos

- Criar um cluster do Amazon DocumentDB
- Descrever Clusters do Amazon DocumentDB
- Modificar um cluster do Amazon DocumentDB
- Determinar manutenção pendente
- Executar uma atualização de patch para a versão do mecanismo de um cluster
- Atualizando o sistema operacional de um cluster
- Interromper e iniciar um cluster do Amazon DocumentDB
- Excluir um cluster do Amazon DocumentDB

Criar um cluster do Amazon DocumentDB

Um cluster do Amazon DocumentDB consiste em instâncias e em um volume de cluster que representa os dados para o mesmo. O volume do cluster é replicado de seis maneiras em três Zonas de Disponibilidade, como volume único e virtual. O cluster contém uma instância principal e, como opção, até 15 instâncias de réplica.

As seções a seguir mostram como criar um cluster do Amazon DocumentDB usando o Console de gerenciamento da AWS ou o. AWS CLI Em seguida, você pode adicionar instâncias de réplica adicionais para esse cluster. Quando o console é usado para criar o cluster do Amazon DocumentDB, uma instância primária é criada automaticamente ao mesmo tempo. Se você usar o AWS CLI para criar seu cluster Amazon DocumentDB, depois que o status do cluster estiver disponível, você deverá criar a instância primária para esse cluster.

Pré-requisitos

Veja a seguir os pré-requisitos para a criação de um cluster do Amazon DocumentDB.

Se você não tiver um Conta da AWS, conclua as etapas a seguir para criar um.

Para se inscrever em um Conta da AWS

1. Aberto https://signin.aws.amazon.com/signup? request_type=registrar.
2. Siga as instruções online.

   Parte do procedimento de inscrição envolve receber uma chamada telefônica ou uma mensagem de texto e inserir um código de verificação pelo teclado do telefone.

   Quando você se inscreve em um Conta da AWS, um Usuário raiz da conta da AWS é criado. O usuário-raiz tem acesso a todos os Serviços da AWS e recursos na conta. Como prática recomendada de segurança, atribua o acesso administrativo a um usuário e use somente o usuário-raiz para executar tarefas que exigem acesso de usuário-raiz.

Pré-requisitos da VPC

É possível criar apenas um cluster do Amazon DocumentDB em uma Amazon Virtual Private Cloud (Amazon VPC). Seu Amazon VPC deve ter no mínimo uma sub-rede em pelo menos duas Zonas de Disponibilidade para que você possa usá-la com um cluster de banco de dados do Amazon DocumentDB. Ao distribuir suas instâncias de cluster por Zonas de Disponibilidade, você garante que as instâncias estejam disponíveis em seu cluster no improvável caso de ocorrer uma falha na Zona de Disponibilidade.

Pré-requisitos de sub-rede

Ao criar um cluster do Amazon DocumentDB, você deve escolher uma VPC e um grupo de sub-rede correspondente dentro dessa VPC para iniciar o seu cluster. Sub-redes determinam a Zona de Disponibilidade e o intervalo de IP dentro dela para executar uma instância. Para essa discussão, usaremos os termos sub-rede e Zona de Disponibilidade alternadamente. Um grupo de sub-redes é um conjunto nomeado de sub-redes (ou Zonas de Disponibilidade). Um grupo de sub-redes permite que você especifique as Zonas de Disponibilidade que deseja usar para executar instâncias do Amazon DocumentDB. Por exemplo, em um cluster com três instâncias, a fim de manter a disponibilidade alta, é recomendável que cada uma dessas instâncias seja provisionada em Zonas de Disponibilidade separadas. Dessa forma, se uma única Zona de Disponibilidade falhar, ela só afetará uma única instância.

As instâncias do Amazon DocumentDB podem ser provisionadas atualmente em até três Zonas de Disponibilidade. Mesmo que um grupo tenha mais de três sub-redes, você só pode usar três dessas sub-redes para criar um cluster do Amazon DocumentDB. Como resultado, é recomendável que, ao criar um grupo de sub-redes, você escolha somente as três sub-redes nas quais deseja implantar suas instâncias. No Leste dos EUA (Norte da Virgínia), seu grupo de sua sub-rede pode ter seis sub-redes (ou Zonas de Disponibilidade). No entanto, quando um cluster do Amazon DocumentDB é fornecido, o Amazon DocumentDB escolhe três dessas Zonas de Disponibilidade e as utiliza para fornecer instâncias.

Por exemplo, suponha que, ao criar um cluster, o Amazon DocumentDB escolhe as Zonas de Disponibilidade {1A, 1B e 1C}. Se você tentar criar uma instância na Zona de Disponibilidade {1D}, a chamada de API não será bem-sucedida. Porém, se você optar por criar uma instância sem especificar uma Zona de Disponibilidade específica, o Amazon DocumentDB escolherá uma Zona de Disponibilidade em seu nome. O Amazon DocumentDB usa um algoritmo para balancear a carga das instâncias em todas as Zonas de Disponibilidade, para ajudar a alcançar alta disponibilidade. Por exemplo, se três instâncias forem fornecidas, por padrão, todas serão fornecidas em três Zonas de Disponibilidade, não em uma única Zona de Disponibilidade.

Recomendações:

- A menos que você tenha um motivo específico, sempre crie um grupo de sub-rede com três sub-redes. Isso ajuda a garantir que clusters com três ou mais instâncias alcancem maior disponibilidade, pois as instâncias são provisionadas em três Zonas de Disponibilidade.
- Sempre distribua instâncias em várias Zonas de Disponibilidade para obter alta disponibilidade. Nunca coloque todas as instâncias de um cluster em uma única Zona de Disponibilidade.
- Como eventos de failover podem acontecer a qualquer momento, você não deve presumir que uma instância primária ou de réplica estará sempre em uma determinada Zona de Disponibilidade.

Pré-requisitos adicionais

Veja a seguir alguns pré-requisitos adicionais para criar um cluster do Amazon DocumentDB:

- Se você estiver se conectando para AWS usar credenciais AWS Identity and Access Management (IAM), sua conta do IAM deve ter políticas do IAM que concedam as permissões necessárias para realizar as operações do Amazon DocumentDB.

Se você estiver usando uma conta do IAM para acessar o console do Amazon DocumentDB, primeiro faça login no Console de gerenciamento da AWS com sua conta do IAM. Em seguida, acesse o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.

- Se quiser personalizar os parâmetros de configuração do seu cluster, você deverá especificar um grupo de parâmetros do cluster e um grupo de parâmetros com as configurações de parâmetro obrigatórias. Para obter informações sobre como criar ou modificar um grupo de parâmetros de cluster ou um grupo de parâmetros, consulte Gerenciando grupos de parâmetros de cluster do Amazon DocumentDB.
- Você deve determinar o número da TCP/IP porta que deseja especificar para seu cluster. Em algumas empresas, os firewalls bloqueiam conexões com as portas padrão do Amazon DocumentDB. Se o firewall da sua empresa bloquear a porta padrão, escolha outra porta para o cluster. Todas as instâncias em um cluster usam a mesma porta.

Criação de um cluster e uma instância primária usando o Console de gerenciamento da AWS

Os procedimentos a seguir descrevem como usar o console para iniciar um cluster do Amazon DocumentDB com uma ou mais instâncias.

Criar um cluster: uso das configurações padrão

Para criar um cluster com instâncias usando as configurações padrão usando o Console de gerenciamento da AWS

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. Se você quiser criar seu cluster em uma região Região da AWS diferente da Leste dos EUA (Norte da Virgínia), escolha a região na lista na seção superior direita do console.
3. No painel de navegação, selecione Clusters e, em seguida, Criar.

   > **Tip**
   >
   > Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

4. Na página Criar cluster Amazon DocumentDB, complete o painel Configuração.

a. Identificador de cluster: aceite o nome fornecido pelo Amazon DocumentDB ou insira um nome para o seu cluster, como, por exemplo, **`sample-cluster`**.

   Restrições de nomeação de cluster:

   - O comprimento é de [1 a 63] letras, números ou hífens.
   - O primeiro caractere deve ser uma letra.
   - Não pode terminar com um hífen ou conter dois hifens consecutivos.
   - Deve ser exclusivo para todos os clusters no Amazon RDS, Neptune e Amazon DocumentDB por Conta da AWS região.

b. Versão do mecanismo — aceite a versão padrão do mecanismo 5.0.0 ou, opcionalmente, escolha 8.0.0, 4.0.0 ou 3.6.0.

c. Classe da instância: aceite o padrão `db.r5.large` ou escolha a classe da instância que deseja usar na lista.

d. Número de instâncias— na lista, escolha o número de instâncias que você deseja criar com esse cluster. A primeira instância é a instância primária e todas as outras instâncias são instâncias de réplica somente leitura. É possível adicionar e excluir instâncias posteriormente, se necessário. Por padrão, um cluster do Amazon DocumentDB é iniciado com três instâncias (uma primária e duas réplicas).

5. Preencha a seção Configuração de armazenamento do cluster.

   Escolha Amazon DocumentDB Standard (padrão) ou Amazon I/O-Optimized DocumentDB. Para obter mais informações, consulte Configurações de armazenamento de cluster do Amazon DocumentDB.

6. Complete o painel Autenticação.

   a. Nome de usuário: insira um nome para o usuário principal. Para fazer login no cluster, você deve usar o nome do usuário principal.

      Restrições de nomenclatura de usuário principal:

      - O comprimento é de [1–63] caracteres alfanuméricos.
      - O primeiro caractere deve ser uma letra.
      - Não pode ser uma palavra reservada pelo mecanismo de banco de dados.

   b. Escolha uma das opções de senha a seguir:

- Gerenciado em AWS Secrets Manager —Escolha essa opção se quiser AWS Secrets Manager gerenciar automaticamente sua senha de usuário principal.

  Se você escolher essa opção, configure a chave do KMS criando sua própria chave ou usando uma chave criada pelo Secrets Manager.

- Autogerenciado: escolha essa opção se quiser autogerenciar sua senha de usuário principal. Se você escolher essa opção, insira uma senha para o usuário principal e, em seguida, confirme-a. Para fazer login no cluster, você deve usar a senha do usuário principal.

  Restrições de senha:

  - O comprimento é de [8–100] caracteres ASCII imprimíveis.
  - Pode usar quaisquer caracteres ASCII imprimíveis, exceto:
    - **/** (barra)
    - **"** (aspas duplas)
    - **@** (arroba)

7. Na parte inferior da tela, escolha uma das seguintes opções:
   - Para criar o cluster agora, escolha Criar cluster.
   - Para não criar o cluster, escolha Cancelar.
   - Para continuar a configurar o cluster antes de criá-lo, escolha Mostrar configurações adicionais e, em seguida, Criar um cluster: configurações adicionais.

     As configurações incluídas na seção Configurações adicionais são as seguintes:

     - Configurações de rede— o padrão é usar o grupo de segurança da VPC `default`.
     - Opções do cluster— o padrão é utilizar a porta 27017 e o grupo parâmetro padrão.
     - Criptografia— o padrão é habilitar a criptografia usando a chave `(default) aws/rds`.

       > **Important**
       >
       > Depois que um cluster é criptografado, ele não pode ser descriptografado.

     - Backup— o padrão é reter os backups por 1 dia e permitir que o Amazon DocumentDB selecione a janela de backup.

- Exportações de registros — O padrão é não exportar registros de auditoria para CloudWatch registros.
- Manutenção— o padrão é permitir que o Amazon DocumentDB selecione a janela de manutenção.
- Proteção contra exclusão— protege o seu cluster contra a exclusão acidental. O padrão para cluster criado usando o console é habilitado.

Se você aceitar os configurações padrão agora, poderá alterar a maioria delas posteriormente modificando o cluster.

8. Habilite a conexão de entrada para o grupo de segurança do cluster.

Se você não alterou as configurações padrão do seu cluster, criou um cluster usando o grupo de segurança padrão para a VPC padrão na região especificada. Para conectar-se ao Amazon DocumentDB, você deve habilitar conexões de entrada na porta 27017 (ou na porta de sua escolha) para o grupo de segurança do cluster.

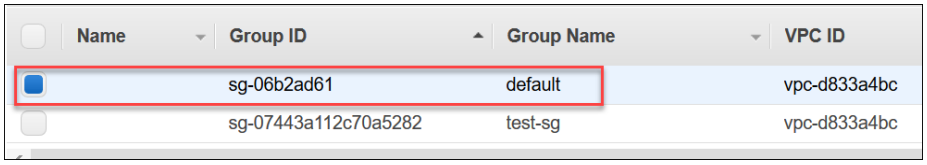
Para adicionar uma conexão de entrada ao seu grupo de segurança do cluster

a. Faça login no Console de gerenciamento da AWS e abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.

b. Na seção Recursos da janela principal, escolha Grupos de segurança.



c. Na lista de grupos de segurança, localize o grupo de segurança que você usou ao criar o cluster (é mais provável o grupo de segurança padrão) e escolha a caixa à esquerda do nome do grupo de segurança.



d. No menu Ações, escolha Editar regras de entrada e, em seguida, escolha ou insira as restrições de regras.

i. Tipo— na lista, escolha o protocolo para abrir para o tráfego de rede.

ii. Protocolo— na lista, escolha o tipo de protocolo.

iii. Intervalo de porta— para uma regra personalizada, insira um número ou um intervalo de porta. Certifique-se de que o número ou o intervalo de porta inclui a porta que você especificou quando criou o cluster (padrão: 27017).

iv. Origem— especifica o tráfego que pode alcançar sua instância. Na lista, escolha a origem do tráfego. Se você escolher Personalizado, especifique um único endereço IP ou um intervalo de endereços IP na notação CIDR (por exemplo, 203.0.113). 5/32).

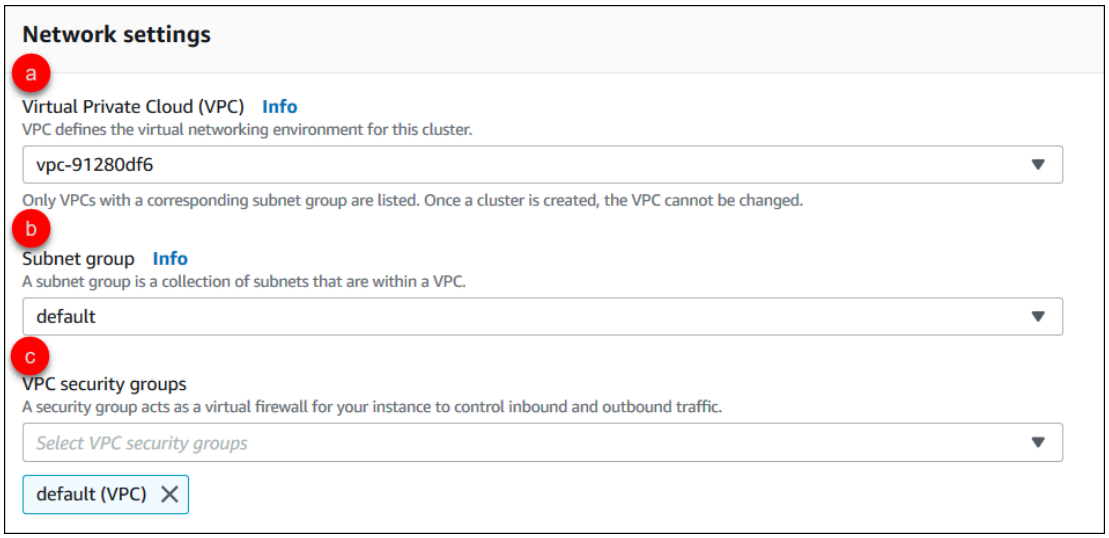
v. Descrição— insira uma descrição para essa regra.

vi. Quando terminar de criar a regra, escolha Salvar.

Criar um cluster: configurações adicionais

Se você deseja aceitar as configurações padrão para o seu cluster, pode ignorar as etapas a seguir e escolher Criar cluster.

1. Conclua o painel Configurações de rede.



a. Nuvem privada virtual (VPC)— na lista, escolha a nuvem privada virtual Amazon que deseja iniciar nesse cluster.

b. Grupo de sub-rede— na lista, escolha o grupo de sub-rede que deseja usar para esse cluster.

c. Grupos de segurança da VPC— na lista, escolha o grupo de segurança da VPC para esse cluster.

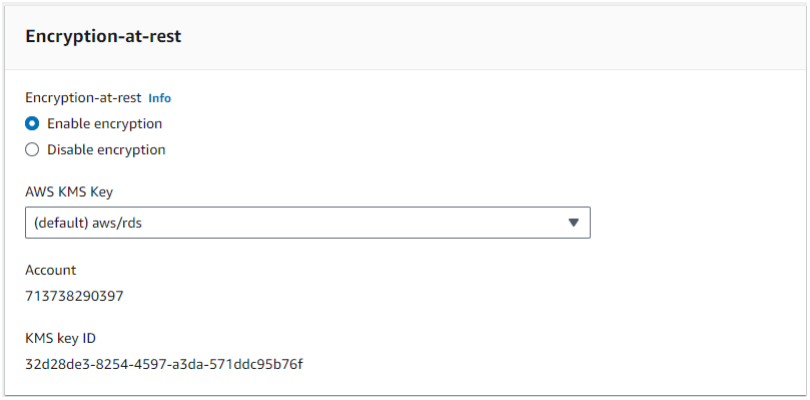
2. Conclua o painel Opções do cluster.



   a. Porta do banco de dados — use as setas para cima e para baixo para definir a TCP/IP porta que os aplicativos usarão para se conectar à sua instância.

   b. Grupo de parâmetros do cluster— na lista de grupos de parâmetros, escolha o grupo de parâmetros para esse cluster.

3. Preencha o painel Criptografia.



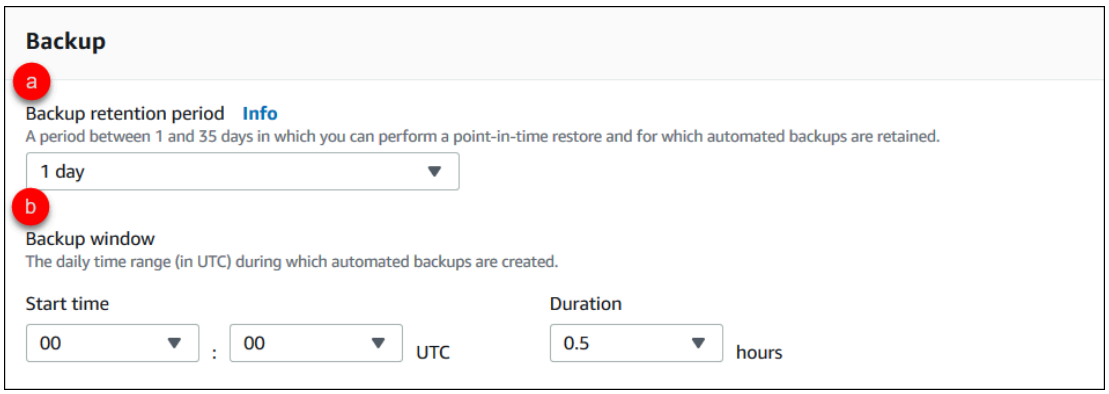
   a. Encryption-at-rest—Escolha uma das seguintes opções:

      - Habilitar criptografia— padrão. Todos os dados em repouso são criptografados. Se optar por criptografar seus dados, você não poderá desfazer essa ação.
      - Desabilitar criptografia— seus dados não estão criptografados.

   b. AWS Chave KMS — Isso só está disponível se você estiver criptografando seus dados. Na lista, escolha a chave que você deseja usar para criptografar os dados desse cluster. O padrão é `(default) aws/rds`.

      Se você escolher Inserir o ARN da chave, deverá inserir o nome do recurso da Amazon (ARN) da chave.

4. Preencha o painel Backup.



a. Período de retenção de backup— na lista, selecione o número de dias para manter os backups automáticos desse cluster antes de excluí-los.

b. Janela de backup— defina o tempo e duração diários durante os quais o Amazon DocumentDB fará backups desse cluster.

   i. Hora de início: na primeira lista, escolha a hora de início (UTC) para seus backups automáticos. Na segunda lista, escolha o minuto da hora em que você deseja que os backups automáticos sejam iniciados.

   ii. Duração— na lista, selecione a quantidade de horas a serem alocadas na criação de backups automáticos.

5. Preencha o painel Exportações de registros selecionando os tipos de registros que você deseja exportar para CloudWatch Registros.

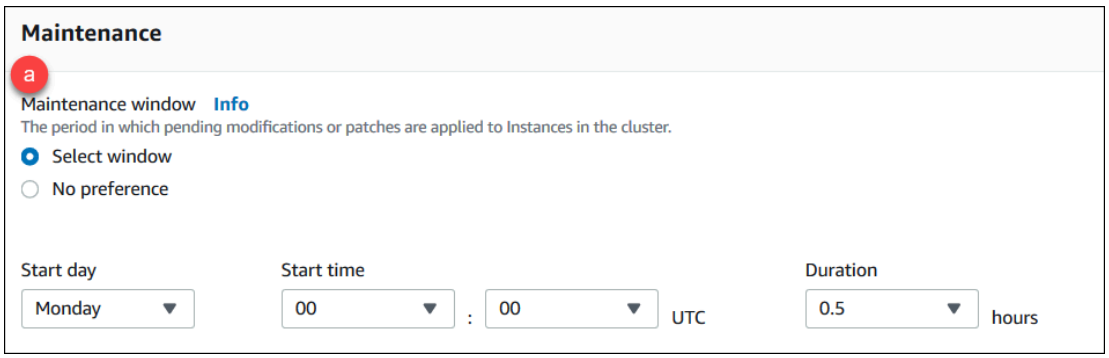


- Registros de auditoria — selecione essa opção para permitir a exportação de registros de auditoria para o Amazon CloudWatch Logs. Se você selecionar Logs de auditoria, deverá habilitar `audit_logs` no grupo de parâmetros personalizado do cluster. Para obter mais informações, consulte Auditar eventos do Amazon DocumentDB.
- Registros do Profiler — Selecione essa opção para permitir a exportação dos logs do Profiler Operation Profiler para o Amazon Logs. CloudWatch Se você selecionar Logs do profiler, deverá também modificar os seguintes parâmetros no grupo de parâmetros personalizado do cluster:
  - `profiler`— defina como `enabled`.

- `profiler_threshold_ms`— defina como um valor `[0-INT_MAX]` para configurar o limite para operações de criação de perfil.
- `profiler_sampling_rate`— defina como um valor `[0.0-1.0]` para configurar o percentual de operações lentas para perfilar.

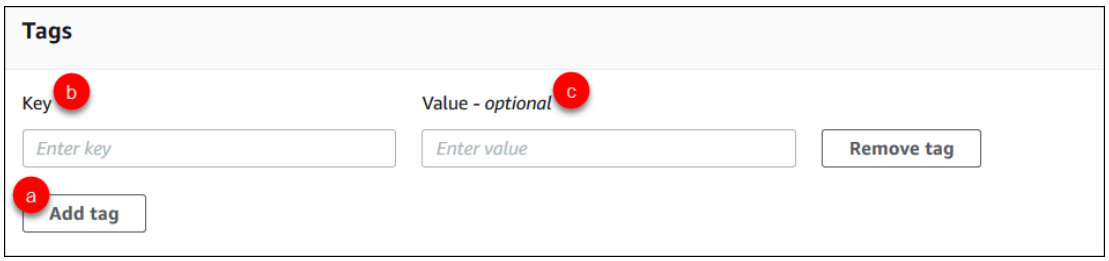
Para obter mais informações, consulte Definir o perfil das operações do Amazon DocumentDB.

6. Complete o painel Manutenção.



- Selecione uma das seguintes opções
  - Selecionar janela— você pode especificar o dia da semana, a hora de início em UTC e a duração do Amazon DocumentDB para executar a manutenção do seu cluster.
    a. Dia de início— na lista, escolha o dia da semana para início da manutenção do cluster.
    b. Hora de início: nas listas, escolha a hora e minuto (UTC) para iniciar a manutenção.
    c. Duração— na lista, escolha quanto tempo alocar para a manutenção do cluster. Se a manutenção não puder ser concluída no horário especificado, o processo de manutenção continuará após o horário especificado, até que seja concluída.
  - Sem preferência— o Amazon DocumentDB seleciona o dia da semana, o horário de início e a duração para executar a manutenção.

7. Se quiser adicionar uma ou mais tags a esse cluster, preencha o painel Tags.



Para cada tag que quiser adicionar ao cluster, repita as etapas a seguir. É possível ter até 10 em um cluster.

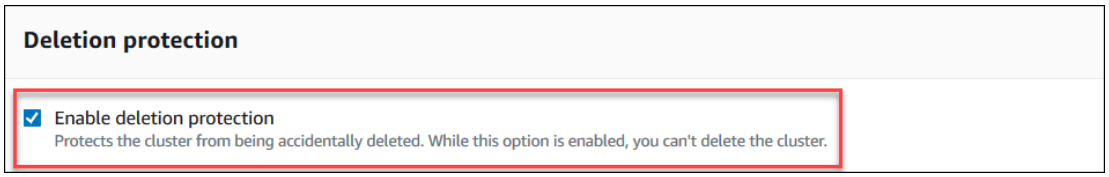
a. Selecione Adicionar tags.

b. Digite a Chave da tag.

c. Opcionalmente, digite o Valor da tag.

Para remover uma tag, selecione Remover tag.

8. Proteção contra exclusão está habilitada por padrão ao criar um cluster usando o console. Para desabilitar a proteção contra exclusão, desmarque Habilitar proteção contra exclusão. Quando habilitada, a proteção contra exclusão impede que um cluster seja excluído. Para excluir um cluster protegido contra a exclusão, primeiro é necessário modificar o cluster para desabilitar a proteção contra exclusão.



Para obter mais informações sobre a proteção contra exclusão, consulte Excluir um cluster do Amazon DocumentDB.

9. Para criar o cluster, escolha Criar cluster. Caso contrário, escolha Cancelar.

Criação de um cluster usando o AWS CLI

Os procedimentos a seguir descrevem como usar o AWS CLI para iniciar um cluster do Amazon DocumentDB e criar uma réplica do Amazon DocumentDB.

Parâmetros

- **`--db-cluster-identifier`**—Obrigatório. Uma string com letras minúsculas que identifica esse cluster.

Restrições de nomeação de cluster:

- O comprimento é de [1 a 63] letras, números ou hifens.
- O primeiro caractere deve ser uma letra.
- Não pode terminar com um hífen ou conter dois hifens consecutivos.
- Deve ser exclusivo para todos os clusters (no Amazon RDS, Amazon Neptune e Amazon DocumentDB) por AWS conta, por região.

- **`--engine`**—Obrigatório. Deve ser **`docdb`**.
- **`--deletion-protection | --no-deletion-protection`**— opcional. Quando a proteção contra exclusão estiver habilitada, ela impede que um cluster seja excluído. Quando você usa o AWS CLI, a configuração padrão é desativar a proteção contra exclusão.

  Para obter mais informações sobre a proteção contra exclusão, consulte Excluir um cluster do Amazon DocumentDB.
- **`--storage-type standard | iopt1`**—Opcional. Padrão: **`standard`**. Configuração de armazenamento do cluster. Os valores válidos são `standard` (Padrão) ou `iopt1` (I/O-optimized).
- **`--master-username`**—Obrigatório. O nome do usuário usado para autenticar o usuário.

  Restrições de nomenclatura do usuário principal:

  - O comprimento é de [1–63] caracteres alfanuméricos.
  - O primeiro caractere deve ser uma letra.
  - Não pode ser uma palavra reservada pelo mecanismo de banco de dados.
- **`--master-user-password`**—Opcional. A senha do usuário usada para autenticar o usuário.

  Restrições da senha mestre:

  - O comprimento é de [8–100] caracteres ASCII imprimíveis.
  - Pode usar quaisquer caracteres ASCII imprimíveis, exceto:
    - **`/`** (barra)
    - **`"`** (aspas duplas)
    - **`@`** (arroba)

- **`--manage-master-user-password`**— opcional. O Amazon DocumentDB gera a senha do usuário principal e a gerencia durante todo o ciclo de vida no Secrets Manager.

Para obter parâmetros adicionais, consulte CreateDBCluster.

Para iniciar um cluster do Amazon DocumentDB usando o AWS CLI

Para criar um cluster Amazon DocumentDB, chame o. `create-db-cluster` AWS CLI O AWS CLI comando a seguir cria um cluster Amazon DocumentDB chamado `sample-cluster` com a proteção contra exclusão ativada. Para obter mais informações sobre a proteção contra exclusão, consulte Excluir um cluster do Amazon DocumentDB.

Além disso, `--engine-version` é um parâmetro opcional padrão para a versão mais recente do mecanismo principal. A versão padrão atual do mecanismo é 5.0.0 (observação: o Amazon DocumentDB 8.0 está disponível, mas deve ser explicitamente especificado). Quando novas versões principais do mecanismo forem lançadas, a versão padrão `--engine-version` será atualizada para refletir a última versão principal do mecanismo. Como resultado, para cargas de trabalho de produção, especialmente aquelas que dependem de scripts, automação ou CloudFormation modelos, especifique-as explicitamente `--engine-version` para a versão principal pretendida.

> **Note**
>
> Se um `db-subnet-group-name` ou `vpc-security-group-id` não for especificado, o Amazon DocumentDB usará o grupo de sub-redes padrão e o grupo de segurança da Amazon VPC para a região em questão.

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante ao seguinte (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"
    }
}
```

Leva alguns minutos para criar o cluster. Você pode usar o Console de gerenciamento da AWS ou AWS CLI para monitorar o status do seu cluster. Para obter mais informações, consulte Monitoramento do status de um cluster do Amazon DocumentDB.

> **Important**
>
> Quando você usa o AWS CLI para criar um cluster do Amazon DocumentDB, nenhuma instância é criada. Consequentemente, é necessário criar explicitamente uma instância principal e qualquer instância de réplica de que precise. Você pode usar o console ou AWS CLI criar as instâncias. Para obter mais informações, consulte Adicionar uma instância do Amazon DocumentDB a um cluster.

Para obter mais informações, consulte CreateDBCluster na Amazon DocumentDB API Reference.

Descrever Clusters do Amazon DocumentDB

Você pode usar o Amazon DocumentDB Management Console ou o AWS CLI para ver detalhes como endpoints de conexão, grupos de segurança, VPCs e grupos de parâmetros relacionados aos seus clusters do Amazon DocumentDB.

Para saber mais, consulte:

- Monitoramento do status de um cluster do Amazon DocumentDB
- Localizar os endpoints de um cluster

Using the Console de gerenciamento da AWS

Use o procedimento a seguir para visualizar os detalhes de um cluster especificado do Amazon DocumentDB usando o console.

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Clusters.

> **Tip**
>
> Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

3. Na lista de clusters, escolha o nome do cluster do qual você deseja ver os detalhes. As informações sobre o cluster estão organizadas nos seguintes agrupamentos:

   - Resumo — informações gerais sobre o cluster, incluindo a versão do mecanismo, o status do cluster, a manutenção pendente e o status do grupo de parâmetros.
   - Conectividade e Segurança: a seção Conectar lista os endpoints de conexão para conectar-se a este cluster, com o shell do mongo ou uma aplicação. A seção Grupos de segurança lista os grupos de segurança associados a esse cluster, suas descrições e ID da VPC.
   - Configuração: a seção Detalhes do cluster lista detalhes sobre o cluster, como nome do recurso da Amazon (ARN), o endpoint e o grupo de parâmetros do cluster. Ela também lista as informações de backup do cluster, os detalhes de manutenção e as configurações de segurança e de rede. A seção Instâncias de cluster lista as instâncias que pertencem a esse cluster com a função de cada instância e o status do grupo de parâmetros do cluster.
   - Monitoramento — As métricas do Amazon CloudWatch Logs para esse cluster. Para obter mais informações, consulte Monitorando o Amazon DocumentDB com CloudWatch.
   - Eventos e tags — a seção Eventos recentes lista os eventos recentes desse cluster. O Amazon DocumentDB mantém um registro de eventos que se relacionam a seus clusters, instâncias, capturas de tela, grupos de segurança e grupos de parâmetros do cluster. Essas informações incluem a data, a hora e a mensagem associadas a cada evento. A seção Tags lista as tags anexadas a este cluster.

Using the AWS CLI

Para visualizar os detalhes dos seus clusters do Amazon DocumentDB usando o AWS CLI, use o `describe-db-clusters` comando conforme mostrado nos exemplos a seguir. Para obter mais informações, consulte DescribeDBClusters na Amazon DocumentDB Resource Management API Reference.

> **Note**
>
> Para determinados atributos de gerenciamento, como o gerenciamento do ciclo de vida de clusters e instâncias, o Amazon DocumentDB aproveita a tecnologia operacional compartilhada com o Amazon RDS. O filtro `filterName=engine,Values=docdb` retorna somente clusters do Amazon DocumentDB.

Example

Exemplo 1: listar todos os clusters do Amazon DocumentDB

O AWS CLI código a seguir lista os detalhes de todos os clusters do Amazon DocumentDB em uma região.

```
aws docdb describe-db-clusters --filter Name=engine,Values=docdb
```

A saída dessa operação é semelhante a seguinte.

```
{
    "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

Exemplo 2: listar todos os detalhes de um cluster especificado do Amazon DocumentDB

O AWS CLI código a seguir lista os detalhes do cluster`sample-cluster`.

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante à seguinte.

```
{
    "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

Exemplo 3: listar os detalhes específicos de um cluster Amazon DocumentDB

Para listar um subconjunto dos detalhes dos clusters usando o AWS CLI, adicione um `--query` que especifique quais membros do cluster a `describe-db-clusters` operação deve listar. O parâmetro `--db-cluster-identifier` é o identificador do cluster específico do qual você deseja exibir os detalhes. Para obter mais informações, consulte Como Filtrar a Saída com a Opção `--query` no Guia de Usuário AWS Command Line Interface .

O exemplo a seguir lista as instâncias em um cluster do Amazon DocumentDB.

Para Linux, macOS ou 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]'
```

A saída dessa operação é semelhante à seguinte.

```
[
    [
        [
            {
                "DBInstanceIdentifier": "sample-instance-1",
                "IsClusterWriter": true,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
            },
            {
                "DBInstanceIdentifier": "sample-instance-2",
                "IsClusterWriter": false,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
            }
        ]
    ]
]
```

Modificar um cluster do Amazon DocumentDB

Para modificar um cluster, o mesmo deve estar no estado disponível. Você não pode modificar um cluster interrompido. Se o cluster for interrompido, primeiro inicie-o, aguarde até que fique disponível e faça as modificações desejadas. Para obter mais informações, consulte Interromper e iniciar um cluster do Amazon DocumentDB.

Using the Console de gerenciamento da AWS

Use o procedimento a seguir para modificar um cluster Amazon DocumentDB específico usando o console.

Para modificar um cluster do Amazon DocumentDB

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Clusters.

> **Tip**
>
> Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

3. Especifique o cluster que você deseja modificar escolhendo o botão à esquerda do nome do cluster.
4. Escolha Ações e, em seguida, Modificar.
5. No painel Modificar cluster: <nome-do-cluster>, faça as alterações desejadas. É possível fazer alterações nas seguintes áreas:
   - Especificações do cluster: o nome, os grupos de segurança e o gerenciamento de credenciais do cluster.
   - Configuração de armazenamento do cluster — o modo de armazenamento de dados do cluster. Escolha entre Padrão e I/O-Optimized configuração.
   - Opções do cluster— a porta e o grupo de parâmetros do cluster.
   - Backup— o período de retenção e a janela de backup do cluster.
   - Exportações de log— habilite ou desabilite exportação dos logs de auditoria ou de profiler.
   - Manutenção— defina a janela de manutenção do cluster.
   - Proteção contra exclusão— habilite ou desabilite a proteção contra exclusão no cluster. Por padrão, a proteção contra exclusão está habilitada.
6. Ao concluir, escolha Continuar para ver um resumo das alterações.
7. Se estiver satisfeito com suas alterações, poderá escolher Modificar cluster para modificar o cluster. Como alternativa, você pode escolher Voltar ou Cancelar para editar ou cancelar as alterações, respectivamente.

Levará alguns minutos para que suas alterações sejam aplicadas. É possível usar o cluster somente quando seu status for disponível. É possível monitorar o status do cluster usando o

console ou a AWS CLI. Para obter mais informações, consulte Monitoramento do status de um cluster do Amazon DocumentDB.

Using the AWS CLI

Use a operação `modify-db-cluster` para modificar o cluster especificado usando a AWS CLI. Para obter mais informações, consulte ModifyDBCluster na Amazon DocumentDB API Reference.

Parâmetros

- **`--db-cluster-identifier`**—Obrigatório. O identificador do cluster do Amazon DocumentDB que você modificará.
- **`--backup-retention-period`**— opcional. O número de dias durante os quais os backups automatizados são retidos. Os valores válidos são 1–35.
- **`--storage-type`**— opcional. Configuração de armazenamento do cluster. Os valores válidos são `standard` (Padrão) ou `iopt1` (I/O-optimized).
- **`--db-cluster-parameter-group-name`**— opcional. O nome do grupo de parâmetros do cluster a ser usado.
- **`--manage-master-user-password`**— opcional. O Amazon DocumentDB gera a senha do usuário principal e a gerencia durante todo o ciclo de vida no Secrets Manager.
- **`--rotate-master-user-password`**—Opcional. O Secrets Manager gera uma nova versão secreta para o segredo existente. A nova versão do segredo contém a nova senha do usuário principal. O Amazon DocumentDB altera a senha do usuário principal do cluster para corresponder à senha para a nova versão do segredo.

  É necessário especificar a opção **`--apply-immediately`** ao alternar a senha principal.
- **`--master-user-password`**—Opcional. A nova senha para o usuário do banco de dados primário.

  Restrições de senha:

  - O comprimento é de [8–100] caracteres ASCII imprimíveis.
  - Pode usar quaisquer caracteres ASCII imprimíveis, exceto:
    - **`/`** (barra)
    - **`"`** (aspas duplas)
    - **`@`** (arroba)

- **`--new-db-cluster-identifier`**— opcional. O novo identificador do cluster durante a renomeação de um cluster. Esse valor é armazenado como uma string em minúsculas.

  Restrições de nomenclatura:

  - O comprimento é de [1 a 63] letras, números ou hifens.
  - O primeiro caractere deve ser uma letra.
  - Não pode terminar com um hífen ou conter dois hifens consecutivos.
  - Deve ser exclusivo para todos os clusters do Amazon RDS, Amazon Neptune e Amazon DocumentDB por Conta da AWS região.
- **`--preferred-backup-window`**— opcional. O período diário durante o qual backups automatizados são criados, em formato Tempo Universal Coordenado (UTC).
  - Formato: `hh24:mm-hh24:mm`.
- **`--preferred-maintenance-window`**— opcional. O intervalo de tempo semanal durante o qual pode ocorrer manutenção do sistema, em UTC.
  - Formato: `ddd:hh24:mm-ddd:hh24:mm`.
  - Dias válidos: `Sun`, `Mon`, `Tue`, `Wed`, `Thu`, `Fri` e `Sat`.
- **`--deletion-protection`** ou **`--no-deletion-protection`**—opcional. Se a proteção contra exclusão deve ser habilitada neste cluster. A proteção contra exclusão impede que um cluster seja excluído acidentalmente até que o cluster seja modificado para desabilitar a proteção contra exclusão. Para obter mais informações, consulte Excluir um cluster do Amazon DocumentDB.
- **`--apply-immediately`** ou **`--no-apply-immediately`**—use `--apply-immediately` para aplicar a alteração imediatamente. Use `--no-apply-immediately` para aplicar a alteração durante a próxima janela de manutenção do cluster.

Example

O código a seguir altera o período de retenção de backup para o cluster `sample-cluster`.

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante à seguinte.

```
{
    "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
    }
}
```

Levará alguns minutos para que suas alterações sejam aplicadas. É possível usar o cluster somente quando seu status for disponível. É possível monitorar o status do cluster usando o console ou a AWS CLI. Para obter mais informações, consulte Monitoramento do status de um cluster do Amazon DocumentDB.

Determinar manutenção pendente

A manutenção geralmente envolve atualizações nos seguintes recursos em seu cluster:

- Sistema operacional (SO) das instâncias no cluster
- Versão do mecanismo do cluster

Para obter mais informações sobre a manutenção do Amazon DocumentDB, consulteManutenção do Amazon DocumentDB.

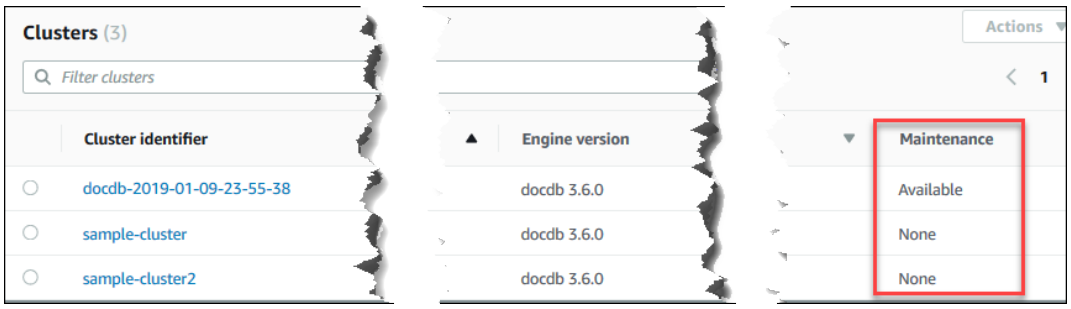
Using the Console de gerenciamento da AWS

Você pode usar o Console de gerenciamento da AWS para determinar se um cluster tem manutenção pendente.

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Clusters.

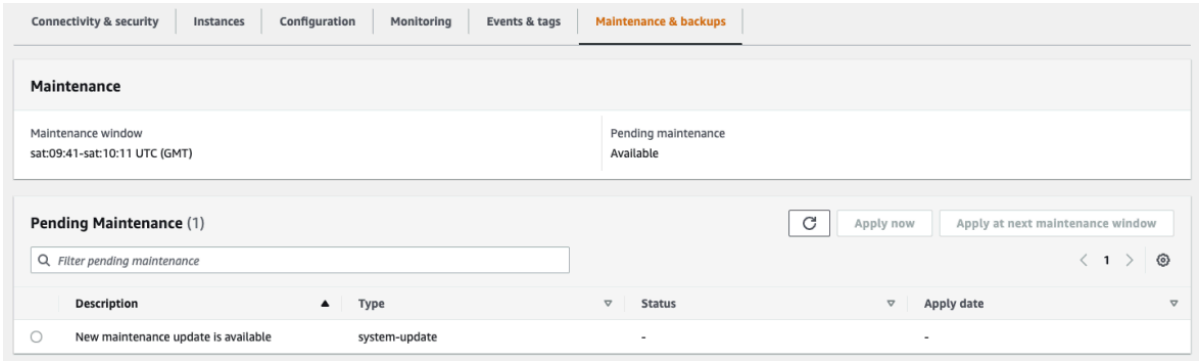
> **Tip**
>
> Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

3. Localize a coluna Manutenção para determinar se um cluster tem manutenção pendente.



Nenhum indica que o cluster está executando a versão mais recente do patch do mecanismo e que suas instâncias têm as atualizações mais recentes do sistema operacional. Disponível indica que o cluster tem manutenção pendente, o que pode significar que uma atualização do mecanismo ou do sistema operacional é necessária.

4. Se o cluster tiver manutenção pendente, você poderá visualizar as ações de manutenção pendentes escolhendo o nome do cluster e selecionando a guia Manutenção e backups. A seção Manutenção pendente lista as ações de manutenção pendentes.



Using the AWS CLI

Você pode usar o AWS CLI para determinar se um cluster tem a versão mais recente do mecanismo e as atualizações do sistema operacional usando a `describe-pending-maintenance-actions` operação com os parâmetros a seguir.

Parâmetros

- **`--resource-identifier`**— opcional. O ARN do recurso (cluster). Se esse parâmetro for omitido, ações de manutenção pendentes para todos os clusters serão listadas.
- **`--region`**— opcional. A Região da AWS na qual você deseja executar essa operação, por exemplo, `us-east-1`.

Example

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante à seguinte.

```
{
    "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"
                }
```

```
            ]
        }
    ]
}
```

Os seguintes tipos de ação de manutenção se aplicam aos clusters do Amazon DocumentDB:

- `system-update`— Atualize o patch do mecanismo para o cluster Amazon DocumentDB.

  Se seu cluster tiver uma atualização de patch de mecanismo pendente disponível, continue com as etapas emExecutar uma atualização de patch para a versão do mecanismo de um cluster.
- `os-upgrade`— Atualize os sistemas operacionais de todas as instâncias de banco de dados no cluster Amazon DocumentDB usando atualizações contínuas.

  Para obter mais informações, consulte Atualizações do sistema operacional do Amazon DocumentDB.

Executar uma atualização de patch para a versão do mecanismo de um cluster

Nesta seção, explicaremos como implantar uma atualização de patch usando o Console de gerenciamento da AWS ou AWS CLI o. Uma atualização de patch é uma atualização dentro da mesma versão do mecanismo (por exemplo, uma versão do mecanismo 3.6 para uma versão mais recente). É possível atualizá-la imediatamente ou durante a próxima janela de manutenção do cluster. Para determinar se seu mecanismo precisa de uma atualização, consulte Determinar manutenção pendente. Observe que, ao aplicar a atualização, seu cluster passará por algum tempo de inatividade.

> **Note**
>
> Se você estiver tentando atualizar de uma versão principal do mecanismo para outra, como 3.6 para 5.0 ou 5.0 para 8.0, consulte Atualização da versão principal implementada do Amazon DocumentDB no local ouAtualizando seu cluster Amazon DocumentDB usando AWS Database Migration Service.

Há dois requisitos de configuração para obter as atualizações de patch mais recentes para a versão do mecanismo de um cluster:

- O status do cluster precisa ser disponível.
- O cluster deve estar executando uma versão de mecanismo mais antiga.

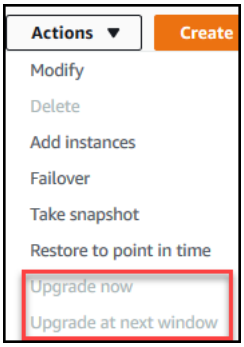
Using the Console de gerenciamento da AWS

O procedimento a seguir atualiza a versão de mecanismo do seu cluster para uma versão mais recente usando o console. Você tem a opção de atualizar imediatamente ou durante a próxima janela de manutenção do cluster.

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Clusters. Na lista de clusters, escolha o botão à esquerda do cluster que você deseja atualizar. O status do cluster precisa ser disponível.

   > **Tip**
   >
   > Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

3. No menu Ações, escolha uma das opções a seguir. Essas opções de menu são selecionáveis somente se o cluster que você escolheu não está executando a versão mais recente do mecanismo.

   - Atualizar agora— inicia imediatamente o processo de atualização. Seu cluster ficará offline por um momento, enquanto é atualizado para a versão mais recente do mecanismo.
   - Atualizar na próxima janela— inicia o processo de atualização durante a próxima janela de manutenção do cluster. O cluster ficará offline por um momento, enquanto é atualizado para a versão mais recente do mecanismo.

4. Quando a janela de confirmação for aberta, escolha uma das seguintes opções:
   - Atualizar— para atualizar seu cluster para a versão mais recente do mecanismo, de acordo com a programação escolhida na etapa anterior.
   - Cancelar— para cancelar a atualização do mecanismo do cluster e continuar com a versão atual.

Using the AWS CLI

Você pode aplicar atualizações de patch ao seu cluster usando a `apply-pending-maintenance-action` operação AWS CLI e com os parâmetros a seguir.

Parâmetros

- **`--resource-identifier`**—Obrigatório. O ARN do cluster do Amazon DocumentDB que você vai atualizar.
- **`--apply-action`**—Obrigatório. Use `system-update` para aplicar uma atualização de patch do mecanismo ao cluster.
- **`--opt-in-type`**—Obrigatório. Os valores a seguir são permitidos.
  - `immediate`— aplique a ação de manutenção imediatamente.
  - `next-maintenance`— aplique a ação de manutenção durante a próxima janela de manutenção.
  - `undo-opt-in`— cancele quaisquer solicitações de inclusão `next-maintenance` existentes.

Example

O exemplo a seguir aplica a atualização do patch do mecanismo `sample-cluster` a.

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante à seguinte.

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

Atualizando o sistema operacional de um cluster

Você pode aplicar atualizações do sistema operacional (SO) em todas as instâncias do seu cluster usando a ação de manutenção no nível do cluster`os-upgrade`. O Amazon DocumentDB atualiza as instâncias de forma contínua, algumas de cada vez, e atualiza a instância primária por último para minimizar os failovers. A atualização é executada durante a janela de manutenção do cluster. Depois que uma instância recebe uma atualização do sistema operacional, seu cache de buffer começa vazio e as consultas nessa instância podem ter maior latência até que o conjunto de trabalho seja repreenchido a partir do volume de armazenamento.

Para determinar se seu cluster tem uma atualização de sistema operacional pendente, consulteDeterminar manutenção pendente. Para obter detalhes completos sobre atualizações do sistema operacional, incluindo procedimentos de console e CLI, consulteAtualizações do sistema operacional do Amazon DocumentDB.

Interromper e iniciar um cluster do Amazon DocumentDB

Interromper e iniciar os clusters do Amazon DocumentDB pode ajudar a gerenciar os custos dos ambientes de teste e desenvolvimento. Em vez de criar e excluir clusters e instâncias toda vez que você usar o Amazon DocumentDB, você pode interromper temporariamente todas as instâncias em seu cluster quando elas não forem necessárias. Depois, é possível iniciá-las novamente, ao retomar os testes.

Tópicos

- Visão geral de como interromper e iniciar um cluster
- Operações que você pode realizar em um cluster interrompido

Visão geral de como interromper e iniciar um cluster

Durante os períodos nos quais não precisar de um cluster do Amazon DocumentDB, você pode interromper todas as instâncias nesse cluster de uma só vez. Depois, você pode iniciar o cluster novamente a qualquer momento, sempre que precisar usá-lo. Iniciar e interromper simplifica os processos de configuração e destruição dos clusters usados em desenvolvimento, teste ou atividades afins que não exijam disponibilidade contínua. Você pode parar e iniciar um cluster usando o Console de gerenciamento da AWS ou o AWS CLI com uma única ação, independentemente de quantas instâncias estejam no cluster.

Enquanto o cluster estiver interrompido, o volume de armazenamento do cluster permanece inalterado. Serão cobrados somente o armazenamento, as captura de telas manuais e o armazenamento do backup automatizado dentro da janela de retenção especificada. Não haverá cobrança por horas de instância. O Amazon DocumentDB iniciará automaticamente seu cluster depois de sete dias para garantir que ele não perca nenhuma atualização de manutenção necessária. Quando o cluster reiniciar após sete dias, você será cobrado pelas instâncias novamente. Enquanto o cluster é interrompido, não é possível consultar o volume de armazenamento porque a consulta requer que as instâncias estejam no estado disponível.

Quando um cluster do Amazon DocumentDB é interrompido, nem o cluster nem suas instâncias podem ser modificados, de maneira alguma. Isso inclui adicionar ou remover instâncias, ou excluir o cluster.

Using the Console de gerenciamento da AWS

O procedimento a seguir mostra como interromper um cluster com uma ou mais instâncias no estado disponível ou iniciar um cluster interrompido.

Para interromper ou iniciar um cluster do Amazon DocumentDB

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Clusters.

> **Tip**
>
> Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

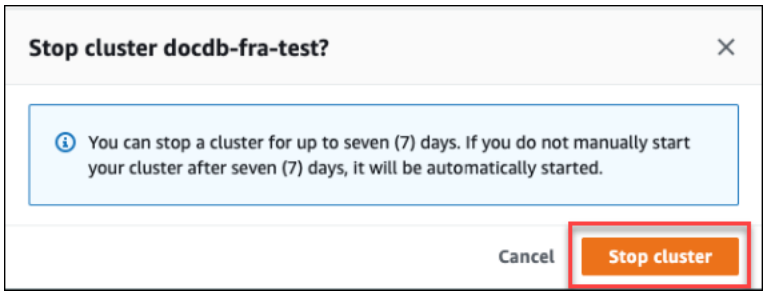
3. Na lista de clusters, escolha o botão à esquerda do nome do cluster que você deseja interromper ou iniciar.
4. Escolha Ações e selecione a ação que deseja executar no cluster.
   - Se você quiser interromper o cluster e ele estiver disponível:

     a. Escolha Parar.

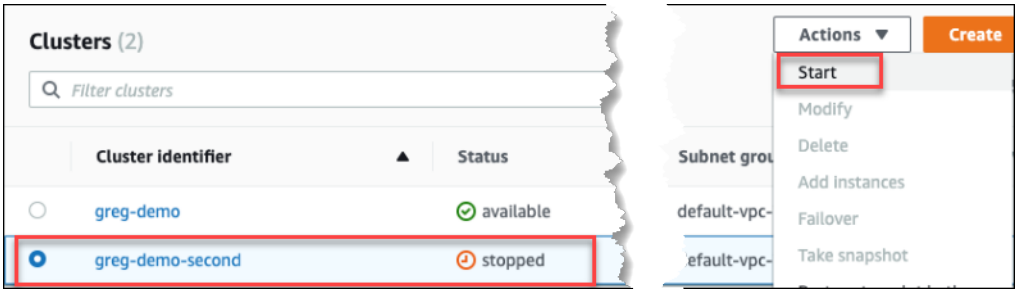


     Para evitar a ativação do mecanismo de failover, a operação de interrupção interrompe primeiro as instâncias de réplica e, depois, a instância primária.

     b. Na caixa de diálogo de confirmação, confirme se deseja interromper o cluster, escolhendo Interromper cluster ou, para manter o cluster em execução, escolha Cancelar.



   - Se quiser iniciar o cluster e o cluster estiver interrompido, escolha Iniciar.



5. Monitore o status do cluster e suas instâncias. Se você iniciou o cluster, poderá retomar o uso do cluster quando ele e suas instâncias estiverem disponíveis. Para obter mais informações, consulte Determinar o status de um cluster.



Using the AWS CLI

Os exemplos de código a seguir mostram como interromper um cluster com uma ou mais instâncias no estado disponível ou iniciar um cluster interrompido.

Para interromper um cluster com uma ou mais instâncias disponíveis usando o AWS CLI, use a `stop-db-cluster` operação. Para iniciar um cluster interrompido, use a operação `start-db-cluster`. Ambas as operações usam o parâmetro `--db-cluster-identifier`.

Parâmetro

- **`--db-cluster-identifier`**—Obrigatório. O nome do cluster a ser interrompido ou iniciado.

Example— Para interromper um cluster usando o AWS CLI

O código a seguir interrompe o cluster `sample-cluster`. O cluster deve ter uma ou mais instâncias no estado disponível.

Para Linux, macOS ou 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 um cluster usando o AWS CLI

O código a seguir inicia o cluster `sample-cluster`. O cluster deve estar interrompido no momento.

Para Linux, macOS ou Unix:

```
aws docdb start-db-cluster \
    --db-cluster-identifier sample-cluster
```

Para Windows:

```
aws docdb start-db-cluster ^
    --db-cluster-identifier sample-cluster
```

Operações que você pode realizar em um cluster interrompido

Com um cluster de banco de dados do Amazon DocumentDB interrompido, você pode executar uma restauração pontual em qualquer ponto dentro da janela de retenção especificada para o backup automático. Para obter detalhes sobre como realizar uma restauração pontual, consulte Restauração para um ponto no tempo.

Não é possível modificar a configuração de um cluster do Amazon DocumentDB, ou de qualquer uma de suas instâncias, enquanto o cluster estiver interrompido. Também não é possível adicionar ou remover instâncias do cluster ou excluir o cluster, caso ainda tenha alguma instância associada. Será necessário iniciar o cluster antes de realizar uma dessas ações administrativas.

O Amazon DocumentDB aplicará as manutenções programadas em seu cluster interrompido somente depois que for reiniciado. Após sete dias, o Amazon DocumentDB inicia automaticamente um cluster interrompido, para que não fique muito atrasado em seu status de manutenção. Quando o cluster for reiniciado, você começará a ser cobrado pelas instâncias no cluster novamente.

Enquanto um cluster estiver interrompido, o Amazon DocumentDB não executa quaisquer backups automatizados, nem estende o período de retenção de backup.

Excluir um cluster do Amazon DocumentDB

Você pode excluir um cluster do Amazon DocumentDB usando o Console de gerenciamento da AWS ou o. AWS CLI Para excluir um cluster, o cluster deve estar no estado disponível e não deve ter nenhuma instância associada a ele. Se o cluster foi interrompido, primeiro inicie-o, aguarde até que fique disponível e então, exclua-o. Para obter mais informações, consulte Interromper e iniciar um cluster do Amazon DocumentDB.

Deletion protection (Proteção contra exclusão)

Para proteger seu cluster contra a exclusão acidental, você pode habilitar a proteção contra exclusão. A proteção contra exclusão está habilitada por padrão ao criar um cluster usando o console. No entanto, a proteção contra exclusão será desabilitada por padrão, se você criar um cluster usando a AWS CLI.

O Amazon DocumentDB aplicará a proteção contra exclusão para um cluster se você executar a operação de exclusão usando o console ou AWS CLI. Se a proteção contra exclusão estiver habilitada, não será possível excluir um cluster. Para excluir um cluster com a projeção contra exclusão habilitada, primeiro é necessário modificar o cluster e desabilitar a proteção contra exclusão.

Ao usar o console com a proteção contra exclusão habilitada em um cluster, não será possível excluir a última instância do cluster, pois isso também excluirá o cluster. É possível excluir a última instância de um cluster protegido contra exclusão usando a AWS CLI. No entanto, o cluster em si ainda existirá e seus dados serão preservados. É possível acessar os dados criando novas instâncias para o cluster. Para obter mais informações sobre como habilitar e desabilitar a proteção contra exclusão, consulte:

- Criar um cluster do Amazon DocumentDB
- Modificar um cluster do Amazon DocumentDB

Using the Console de gerenciamento da AWS

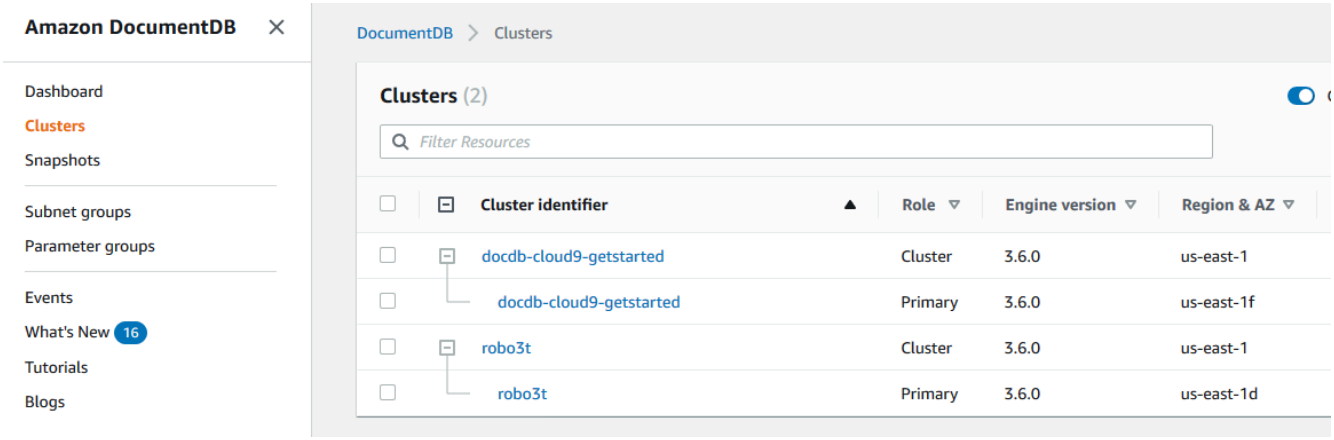
> Para excluir um cluster usando o Console de gerenciamento da AWS, a proteção contra exclusão deve ser desativada.

Para determinar se um cluster está com a proteção contra exclusão habilitada:

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Clusters.

> **Tip**
>
> Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

3. Observe que, na caixa de navegação Clusters, a coluna Identificador do cluster mostra tanto os clusters quanto as instâncias. As instâncias são listadas abaixo dos clusters, de forma semelhante à captura de tela a seguir.



4. Escolha o nome do cluster e selecione a guia Configuração. Na seção Detalhes do cluster, localize Proteção contra exclusão. Se a proteção contra exclusão estiver habilitada, modifique o cluster para desabilitar a proteção contra exclusão. Para obter informações sobre como modificar um cluster, consulte Modificar um cluster do Amazon DocumentDB.

   Após desabilitar a Proteção contra exclusão, você estará pronto para excluir o cluster.

Para excluir um cluster:

1. No painel de navegação, escolha Clusters.
2. Determine se o cluster tem instâncias verificando a coluna Identificador do cluster para as instâncias lidas abaixo dele. Antes de excluir um cluster, é necessário excluir todas

as instâncias. Para obter mais informações, consulte Excluir uma instância do Amazon DocumentDB.

3. Dependendo de o cluster ter ou não instâncias, realize uma das seguintes etapas.
   - Se o cluster não tiver instâncias, selecione o botão à esquerda do nome do cluster e escolha Ações. No menu suspenso, escolha Excluir. Preencha a caixa de diálogo Excluir <nome-cluster> e escolha Excluir.
   - Se o cluster tiver uma ou mais instâncias, faça o seguinte:

     a. No painel de navegação, escolha Clusters.

     b. Exclua cada uma das instâncias do cluster escolhendo a caixa de seleção à esquerda do nome do cluster. Selecione Ações e escolha Excluir. Preencha a caixa de diálogo Excluir <nome-cluster> e escolha Excluir.

        Ao excluir a última instância, o cluster também será excluído. Para obter informações sobre a exclusão de instâncias, consulte Excluir uma instância do Amazon DocumentDB.

A exclusão do cluster demora alguns minutos. Para monitorar o status do cluster, consulte Monitoramento do status de um cluster do Amazon DocumentDB.

Using the AWS CLI

Não é possível excluir um cluster que tenha instâncias associadas a ele. Para determinar quais instâncias estão associadas ao cluster, execute o comando `describe-db-clusters` e exclua todas as instâncias do cluster. Depois, se necessário, desabilite a proteção contra exclusão no cluster e, finalmente, exclua o cluster.

1. Primeiro, exclua todas as instâncias do cluster.

   Para determinar quais instâncias você precisa excluir, execute o comando a seguir.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].
[DBClusterIdentifier,DBClusterMembers[*].DBInstanceIdentifier]'
```

   A saída dessa operação é semelhante ao seguinte (formato JSON).

```
[
```

```
    [
        "sample-cluster",
        [
            "sample-instance-1",
            "sample-instance-2"
        ]
    ]
]
```

Se o cluster que você deseja excluir tiver instâncias, exclua-as conforme abaixo.

```
aws docdb delete-db-instance \
    --db-instance-identifier sample-instance
```

2. Em segundo lugar, desabilite a proteção contra exclusão.

Usar o AWS CLI para excluir todas as instâncias de um cluster não exclui o cluster. Também é necessário excluir o cluster, mas isso só poderá ser feito se a proteção contra exclusão estiver desabilitada.

Para determinar se o cluster está com a proteção contra exclusão habilitada, execute o comando a seguir.

> **Tip**
>
> Para ver o status da proteção contra exclusão de todos os clusters do Amazon DocumentDB, omita o parâmetro `--db-cluster-identifier`.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].[DBClusterIdentifier,DeletionProtection]'
```

A saída dessa operação é semelhante à seguinte.

```
[
    [
        "sample-cluster",
        "true"
```

```
    ]
]
```

Se o cluster estiver com a proteção contra exclusão habilitada, modifique o cluster e desabilite a proteção contra exclusão. Para desabilitar a proteção contra exclusão no cluster, use o comando a seguir.

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --no-deletion-protection \
    --apply-immediately
```

3. Para finalizar, exclua o cluster.

   Após desabilitar a proteção contra exclusão, você estará pronto para excluir o cluster. Para excluir um cluster, use a operação `delete-db-cluster` com os parâmetros a seguir.

   - **`--db-cluster-identifier`**—Obrigatório. O identificador do cluster que deseja excluir.
   - **`--final-db-snapshot-identifier`**— opcional. Se você quiser uma captura de tela final, inclua esse parâmetro com um nome para a captura de tela final. É necessário incluir `--final-db-snapshot-identifier` ou `--skip-final-snapshot`.

     Restrições de nomenclatura:

     - O comprimento é de [1 a 63] letras, números ou hifens.
     - O primeiro caractere deve ser uma letra.
     - Não pode terminar com um hífen ou conter dois hifens consecutivos.
     - Deve ser exclusivo para todos os clusters do Amazon RDS, Amazon Neptune e Amazon DocumentDB por Conta da AWS região.
   - **`--skip-final-snapshot`**— opcional. Use esse parâmetro somente se não quiser obter um captura de tela final antes de excluir seu cluster. A configuração padrão é obter uma captura de tela final. É necessário incluir `--final-db-snapshot-identifier` ou `--skip-final-snapshot`.

O AWS CLI código a seguir exclui o cluster `sample-cluster` com um instantâneo final. A operação falhará se houver instâncias associadas ao cluster ou se a proteção contra exclusão estiver desabilitada.

Example

Para Linux, macOS ou 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

O AWS CLI código a seguir exclui o cluster `sample-cluster` sem tirar um instantâneo final.

Para Linux, macOS ou 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
```

A saída da operação `delete-db-cluster` é o cluster que você está excluindo.

A exclusão do cluster demora alguns minutos. Para monitorar o status do cluster, consulte Monitorar o status de um cluster.

Escalar clusters do Amazon DocumentDB

O Amazon DocumentDB permite escalar o armazenamento e a computação nos clusters com base em suas necessidades. Esta seção descreve como você pode escalar armazenamento, instância e leitura, para gerenciar o desempenho e a escala dos clusters e das instâncias do Amazon DocumentDB.

Tópicos

- Escalabilidade de armazenamento
- Escalabilidade de instâncias
- Escalabilidade de leitura
- Gravar escalabilidade

Escalabilidade de armazenamento

O armazenamento do Amazon DocumentDB escala automaticamente com os dados no volume do cluster. À medida que seus dados crescem, o volume de armazenamento do cluster cresce em incrementos de 10 GiB, até 256 TiB para o Engine versão 8.0 e superior (128 TiB para versões anteriores do mecanismo).

Escalabilidade de instâncias

É possível escalar o cluster do Amazon DocumentDB conforme a necessidade e modificar a classe da instância para cada instância do cluster. O Amazon DocumentDB oferece suporte a várias classes de instância otimizadas.

Para obter mais informações, consulte Modificar uma instância do Amazon DocumentDB.

Escalabilidade de leitura

É possível obter uma escala de leitura para o cluster do Amazon DocumentDB criando até 15 réplicas do Amazon DocumentDB no cluster. Cada réplica do Amazon DocumentDB retornará os mesmos dados do volume de cluster com atraso de réplica mínimo — geralmente inferior a 100 milissegundos após a instância principal ter gravado uma atualização. Conforme o tráfego de leitura aumenta, você pode criar réplicas adicionais do Amazon DocumentDB e conectar-se a elas para distribuir a carga de leitura para o seu cluster. As réplicas do Amazon DocumentDB não precisam ser da mesma classe da instância primária.

Para obter mais informações, consulte Adicionar uma instância do Amazon DocumentDB a um cluster.

Para escalar a leitura com o Amazon DocumentDB, conecte-se ao seu cluster como um conjunto de réplicas e distribua leituras para instâncias de réplica usando os recursos integrados de preferência de leitura do seu driver. Para obter mais informações, consulte Conectar-se ao Amazon DocumentDB como um conjunto de réplicas.

Gravar escalabilidade

É possível escalar a capacidade de gravação no cluster do Amazon DocumentDB aumentando o tamanho da instância principal do cluster. Esta seção fornece dois métodos para escalar a instância principal do cluster com base em suas necessidades. A primeira opção procura minimizar o impacto na aplicação, mas exige que mais etapas sejam concluídas. A segunda opção otimiza visando simplicidade, pois tem menos etapas, mas, em compensação, representa potencial impacto maior na aplicação.

De acordo com a aplicação, você pode escolher qual abordagem abaixo é a melhor para você. Para obter mais informações sobre custos e tamanhos de instância disponíveis, consulte a página Definição de preço do Amazon DocumentDB Pricing.

1. Otimize para alta disponibilidade e desempenho: se estiver se conectando ao cluster em modo de conjunto de réplica (recomendado), você pode usar o processo a seguir para minimizar o impacto na aplicação ao escalar a instância principal. Esse método minimiza o impacto porque mantém o cluster em alta disponibilidade ou acima, e os destinos em escala de leitura são adicionados ao cluster como instâncias, em vez de atualizados no local.

    a. Adicione uma ou mais réplicas do tipo de instância maior ao seu cluster (consulte ???). Certifique-se de que todas as réplicas sejam do mesmo tipo de instância ou maior que a primária. Isso evita que uma redução não intencional no desempenho de gravação execute failover para um tipo de instância menor. Na maioria dos casos, isso significa dobrar temporariamente o número de instâncias em seu cluster e, em seguida, remover as réplicas menores após a conclusão do escalonamento.

    b. Defina o nível de failover em todas as réplicas novas como prioridade zero, de forma a garantir que uma réplica do tipo de instância menor tenha a maior prioridade de failover. Para obter mais informações, consulte ???.

    c. Inicie um failover manual, o que promoverá uma das novas réplicas a instância principal. Para obter mais informações, consulte ???.

> **Note**
>
> Isso incorrerá em aproximadamente 30 segundos de tempo de inatividade para o cluster. Planeje adequadamente.

d. Remova do cluster todas as réplicas de um tipo de instância menor que a nova principal.

e. Defina o nível de failover de todas as instâncias de volta para a mesma prioridade (geralmente, isso significa defini-las de volta a 1).

Como exemplo, vamos supor que você tenha um cluster com três instâncias `r5.large` (uma principal e duas réplicas), e que deseje escalar para um tipo de instância `r5.xlarge`. Para fazê-lo, primeiro adicione três instâncias de réplica `r5.xlarge` ao seu cluster e defina o nível de failover das novas réplicas `r5.xlarge` como zero. Depois, inicie um failover manual (considerando que a aplicação terá aproximadamente 30 segundos de tempo de inatividade). Quando o failover for concluído, remova todas as três instâncias `r5.large` do cluster deixando-o escalado para instâncias `r5.xlarge`.

Para ajudar a otimizar os custos, as instâncias do Amazon DocumentDB são cobradas em incrementos de um segundo, com uma cobrança mínima de dez minutos após uma alteração no status de faturamento, como criação, modificação ou exclusão de uma instância. Para obter mais informações, consulte Otimização de custos na documentação de práticas recomendadas.

2. Otimize para simplificar — essa abordagem otimiza para simplicidade. Ela não expande nem contrai o cluster, mas pode reduzir temporariamente sua capacidade de leitura.

   É possível que a alteração da classe da instância de uma réplica faça com que a mesma não atenda as solicitações por um breve período, de alguns segundos a menos de 30 segundos. Se você estiver se conectando ao seu cluster no modo de conjunto de réplicas (recomendado), isso reduzirá sua capacidade de leitura em uma réplica (por exemplo, para 66% da capacidade em um cluster de 3 nós, ou 75% da capacidade em um cluster de 4 nós, etc.) durante a operação em escala.

   a. Escale uma das instâncias de réplica no seu cluster. Para obter mais informações, consulte Gerenciar classes de instância.

   b. Espere até que a instância esteja disponível (consulte Monitoramento do status de uma instância do Amazon DocumentDB).

> **Note**
>
> Isso incorrerá em aproximadamente 30 segundos de tempo de inatividade para o cluster. Planeje adequadamente.

c. Continue executando as etapas 1 e 2 até que todas as instâncias de réplicas tenham sido escaladas, uma a uma.

d. Iniciar um failover manual. Isso promoverá uma das réplica à instância principal. Para obter mais informações, consulte Failover do Amazon DocumentDB.

> **Note**
>
> Isso resultará em até 30 segundos de inatividade para seu cluster, mas, geralmente leva menos tempo do que isso. Planeje adequadamente.

e. Escale a antiga instância primária (agora, uma réplica).

Clonar um volume para um cluster Amazon DocumentDB

Com a clonagem do Amazon DocumentDB, você pode criar um cluster que use o mesmo volume de cluster do Amazon DocumentDB e contenha os mesmos dados do original. O processo foi projetado para ser rápido e econômico. O novo cluster e seu volume de dados associado é chamado de clone.. Criar um clone é mais rápido e eficiente em termos de espaço do que copiar fisicamente os dados usando outras técnicas, como restauração ou captura de tela.

O Amazon DocumentDB oferece suporte à criação de um clone provisionado do Amazon DocumentDB por meio de um cluster provisionado do Amazon DocumentDB. Quando você cria um clone usando uma configuração de implantação diferente da origem, o clone é criado usando a versão mais recente do mecanismo Amazon DocumentDB de origem.

Quando você cria clones a partir de seus clusters do Amazon DocumentDB, os clones são criados em sua conta AWS — a mesma conta que possui o cluster de origem do Amazon DocumentDB.

Tópicos

- Visão geral da clonagem do Amazon DocumentDB
- Notas de uso da clonagem do Amazon DocumentDB
- Como funciona a clonagem do Amazon DocumentDB

- Criar um clone do Amazon DocumentDB

Visão geral da clonagem do Amazon DocumentDB

O Amazon DocumentDB usa um protocolo copiar ao escrever para criar um clone. Esse mecanismo usa um espaço adicional mínimo para criar um clone inicial. Quando o clone é criado pela primeira vez, o Amazon DocumentDB mantém uma única cópia dos dados usados pelo cluster de origem do Amazon DocumentDB e pelo novo cluster (clonado) do Amazon DocumentDB. O armazenamento adicional é alocado somente quando as alterações são feitas nos dados (no volume de armazenamento do Amazon DocumentDB) pelo cluster de origem do Amazon DocumentDB, ou pelo clone do cluster do Amazon DocumentDB. Para saber mais sobre o protocolo copiar ao escrever, consulte Como funciona a clonagem do Amazon DocumentDB.

A clonagem do Amazon DocumentDB é útil principalmente para configurar rapidamente ambientes de teste usando seus dados de produção, sem o risco de corromper dados. É possível utilizar clones para vários tipos de aplicações, como:

- Experimente possíveis alterações (de esquema e de grupos de parâmetros, por exemplo) para avaliar todos os impactos.
- Execute operações com workloads intensivas, como exportar dados ou executar consultas analíticas no clone.
- Crie uma cópia do cluster de banco de dados de produção para desenvolvimento, teste ou outras finalidades.

É possível criar mais de um clone do mesmo cluster Amazon DocumentDB. Também é possível criar vários clones a partir de outro.

Depois de criar um clone do Amazon DocumentDB, você pode configurar suas instâncias de modo diferente do cluster de origem. Por exemplo, talvez você não precise que um clone para fins de desenvolvimento atenda aos mesmos requisitos de alta disponibilidade que o cluster de produção de origem Amazon DocumentDB. Nesse caso, é possível configurar o clone com uma única instância Amazon DocumentDB em vez de várias instâncias DB usadas pelo cluster Amazon DocumentDB.

Ao concluir o uso do clone para seus testes, desenvolvimento, ou outras finalidades, será possível excluí-lo.

Notas de uso da clonagem do Amazon DocumentDB

Atualmente, a clonagem do Amazon DocumentDB; tem as seguintes limitações:

- É possível criar quantos clones quiser, até o número máximo de clusters DB permitido na Região da AWS. No entanto, depois 15 clones, o próximo será uma cópia completa. A operação de clonagem age como uma recuperação pontual.
- Você não pode criar um clone em uma AWS região diferente do cluster Amazon DocumentDB de origem.
- Não é possível criar um clone a partir de um cluster Amazon DocumentDB sem instâncias. Só é possível clonar clusters Amazon DocumentDB que tenham pelo menos uma instância DB.
- É possível criar um clone em uma nuvem privada virtual (VPC) diferente daquela do cluster Amazon DocumentDB. Nesse caso, as sub-redes dessas VPCs devem ser mapeadas mas mesmas Zonas de Disponibilidade.

Como funciona a clonagem do Amazon DocumentDB

A clonagem do Amazon DocumentDB funciona na camada de armazenamento de um cluster do Amazon DocumentDB. Ela utiliza um protocolo copiar ao escrever rápido e eficiente em termos de espaço, no que diz respeito à mídia durável suportando o volume de armazenamento Amazon DocumentDB. Saiba mais sobre os volumes de cluster do Amazon DocumentDB em Gerenciar clusters do Amazon DocumentDB.

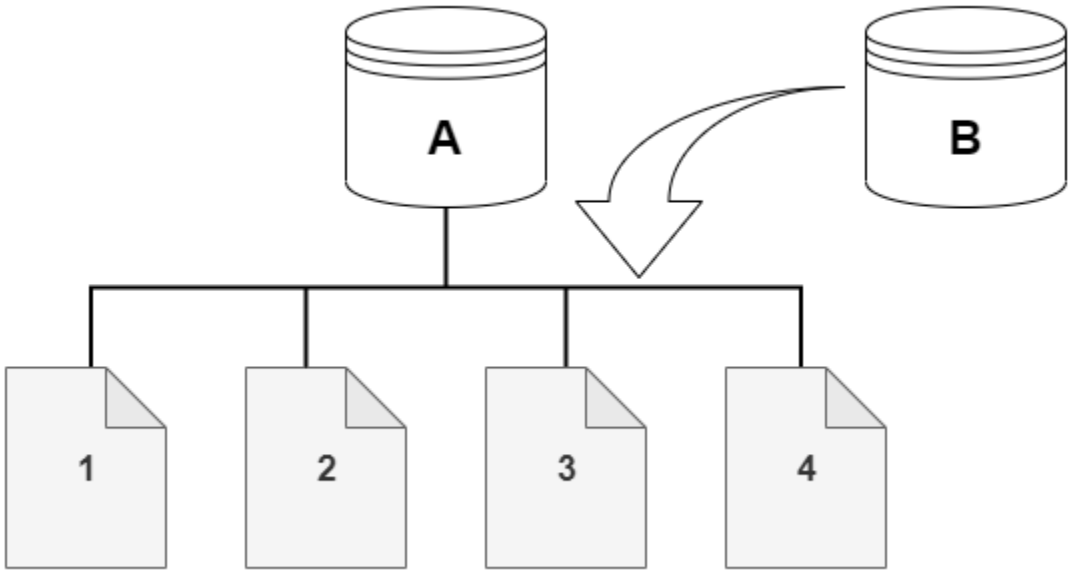
Tópicos

- Noções sobre o protocolo copiar ao escrever
- Excluir um volume de cluster de origem

Noções sobre o protocolo copiar ao escrever

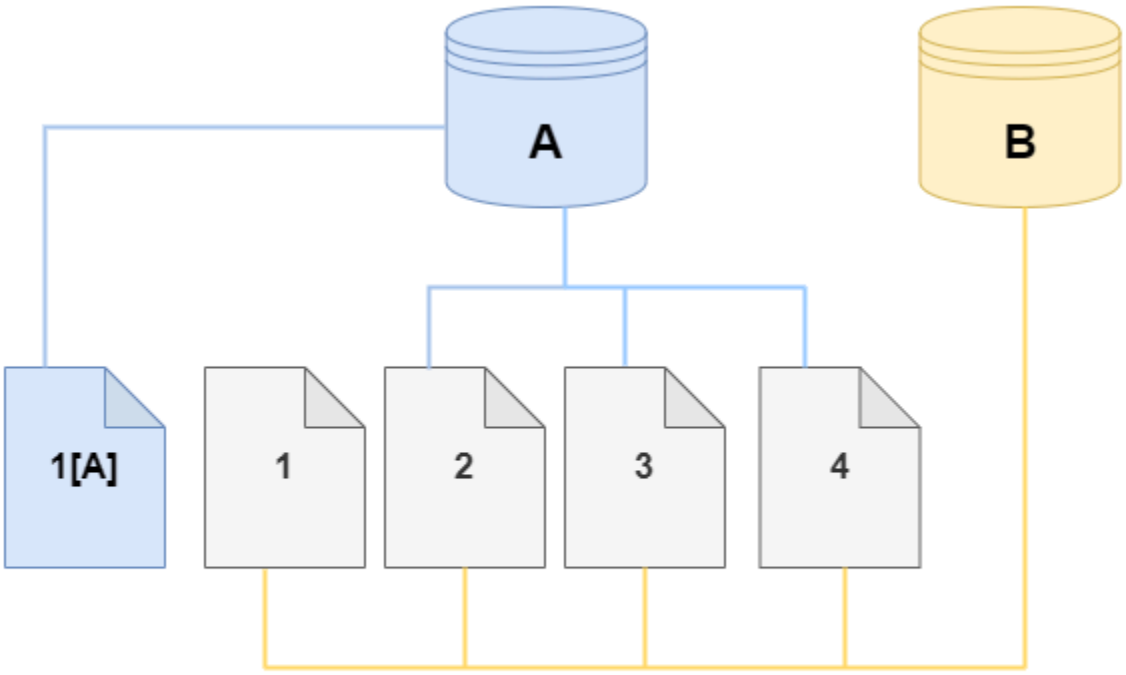
Um cluster do Amazon DocumentDB armazena dados em páginas do volume de armazenamento subjacente Amazon DocumentDB.

Por exemplo, no diagrama a seguir, você descobre um cluster Amazon DocumentDB (A) com quatro páginas de dados: 1, 2, 3 e 4. Imagine que um clone, B, é criado a partir do cluster do Amazon DocumentDB. Quando o clone é criado, nenhum dado é copiado. Em vez disso, o clone aponta para o mesmo conjunto de páginas que o cluster Amazon DocumentDB de origem.

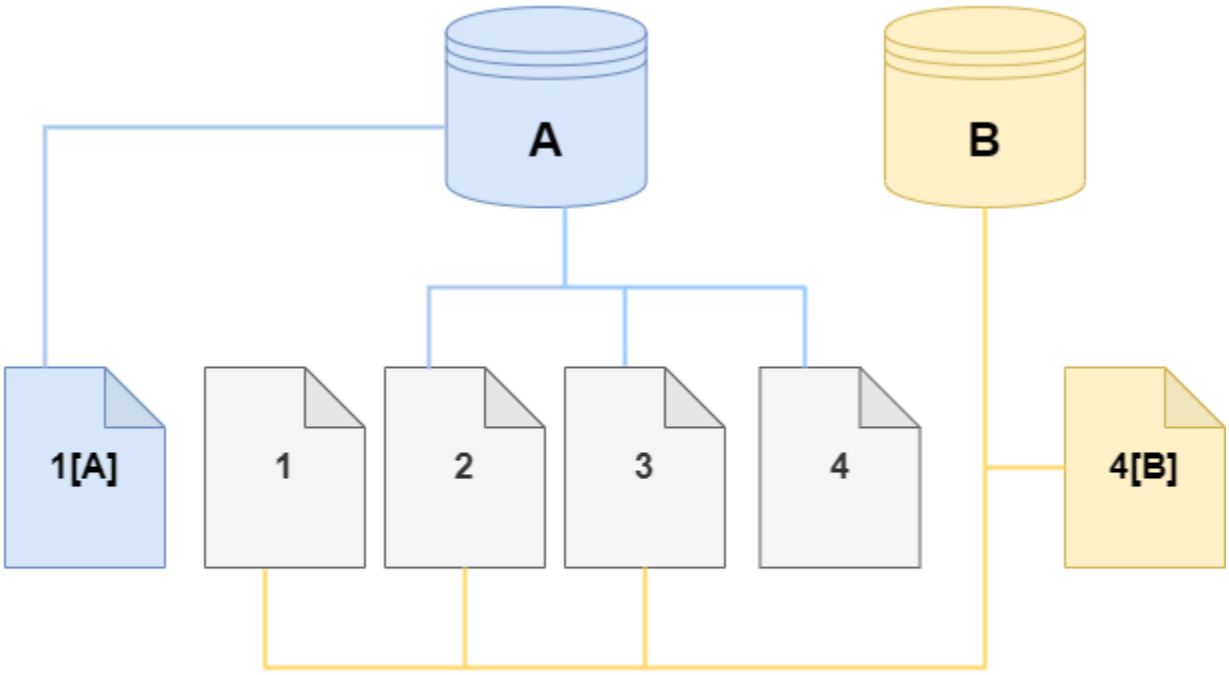


Quando o clone é criado, geralmente não é necessário armazenamento adicional. O protocolo copiar ao escrever usa o mesmo segmento na mídia de armazenamento física que no segmento de origem. O armazenamento adicional é necessário somente se a capacidade do segmento de origem não for suficiente para todo o segmento do clone. Se for esse o caso, o segmento de origem será copiado para outro dispositivo físico.

Nos diagramas a seguir, você descobre um exemplo do protocolo copiar a escrever em ação usando o mesmo cluster A e seu clone B, conforme mostrado anteriormente. Digamos que você faça uma alteração no cluster do Amazon DocumentDB (A) que resulte em uma alteração nos dados mantidos na página 1. Em vez de gravar na página 1 original, o Amazon DocumentDB cria uma nova página, 1[A]. O volume do cluster Amazon DocumentDB para o cluster (A) agora aponta para a página 1[A], 2, 3 e 4, enquanto o clone (B) ainda faz referência às páginas originais.



No clone, uma alteração é feita na página 4, no volume de armazenamento. Em vez de gravar na página 4 original, o Amazon DocumentDB cria uma nova página, 4[B]. O clone agora aponta para as páginas 1, 2, 3 e para a página 4[B], enquanto o cluster (A) continua apontando para 1[A], 2, 3 e 4.



À medida que ocorrerem mais alterações ao longo do tempo no volume do cluster do Amazon DocumentDB original e no clone, será necessário mais armazenamento incremental para capturar e armazenar as alterações.

Excluir um volume de cluster de origem

Quando você exclui um volume do cluster de origem com um ou mais clones associados a ele, os clones não são afetados. Os clones continuam a apontar para as páginas que pertenciam anteriormente ao volume do cluster de origem.

Criar um clone do Amazon DocumentDB

Você pode criar um clone na mesma AWS conta do cluster Amazon DocumentDB de origem. Para fazer isso, você pode usar o Console de gerenciamento da AWS ou o AWS CLI e os procedimentos a seguir.

Ao usar a clonagem do Amazon DocumentDB, você pode criar um clone provisionado por meio de um cluster provisionado do Amazon DocumentDB.

Using the Console de gerenciamento da AWS

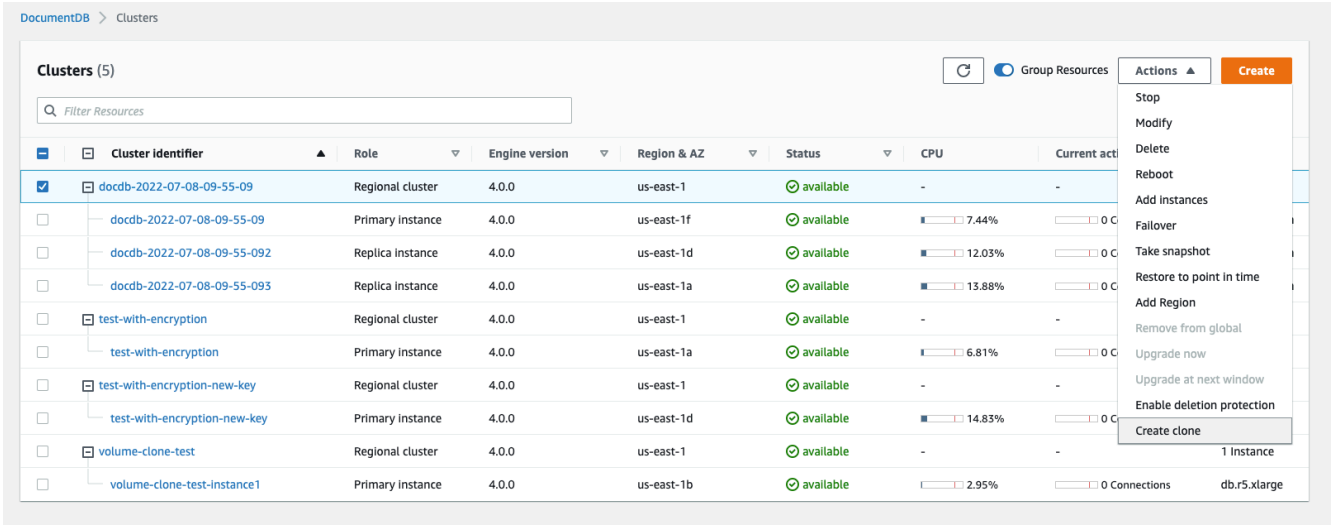
O procedimento a seguir descreve como clonar um cluster Amazon DocumentDB usando Console de gerenciamento da AWS.

Criação de um clone usando os Console de gerenciamento da AWS resultados em um cluster do Amazon DocumentDB com uma instância do Amazon DocumentDB.

Essas instruções se aplicam a clusters de banco de dados pertencentes à mesma AWS conta que está criando o clone. O cluster de banco de dados deve pertencer à mesma AWS conta, pois a clonagem entre contas não é suportada no Amazon DocumentDB.

Para criar um clone de um cluster de banco de dados de propriedade de seu AWS conta usando o Console de gerenciamento da AWS

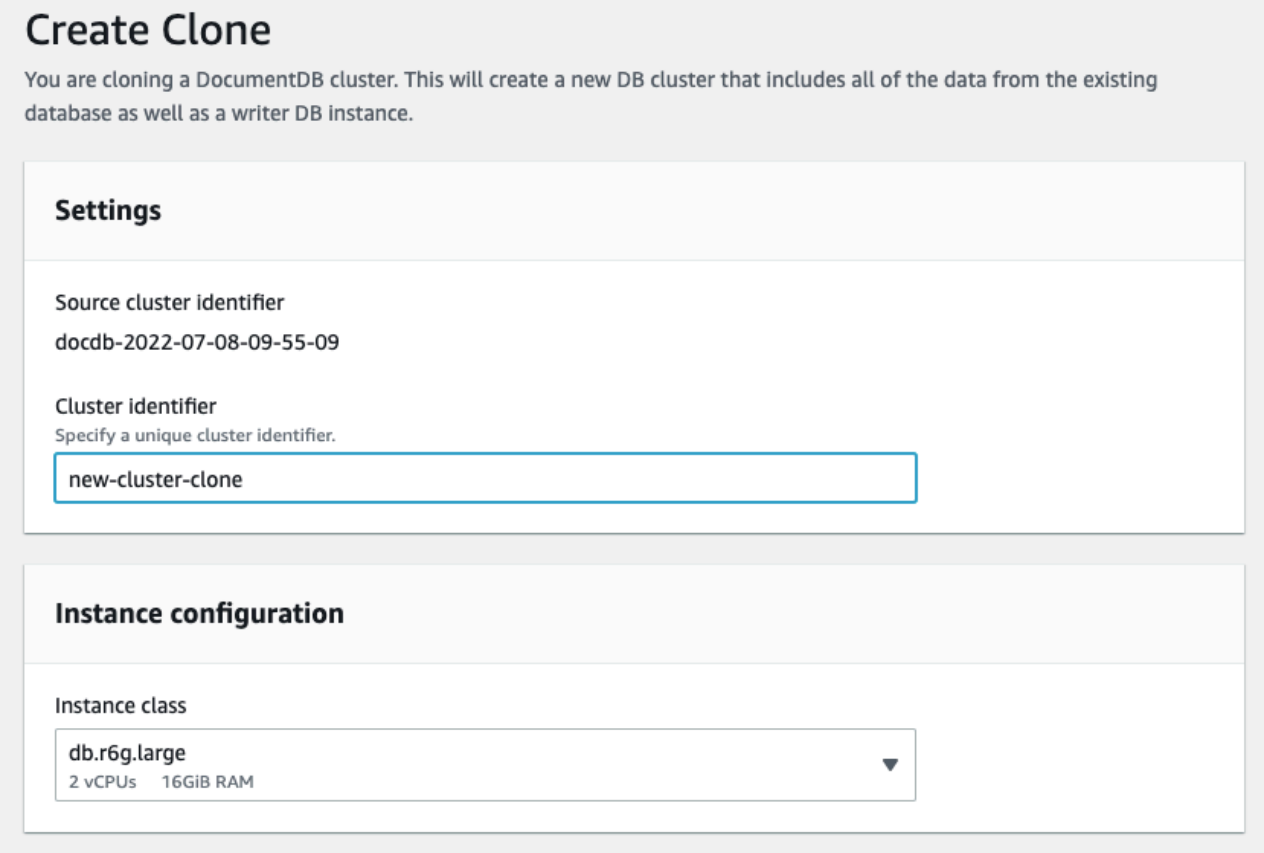
1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Clusters.
3. Escolha seu cluster Amazon DocumentDB da lista e em Ações, escolha Criar clone.



Na página Criar clone aberta, é possível configurar um Identificador de Cluster e uma Classe de instância, além de outras opções para o clone de cluster do Amazon DocumentDB.

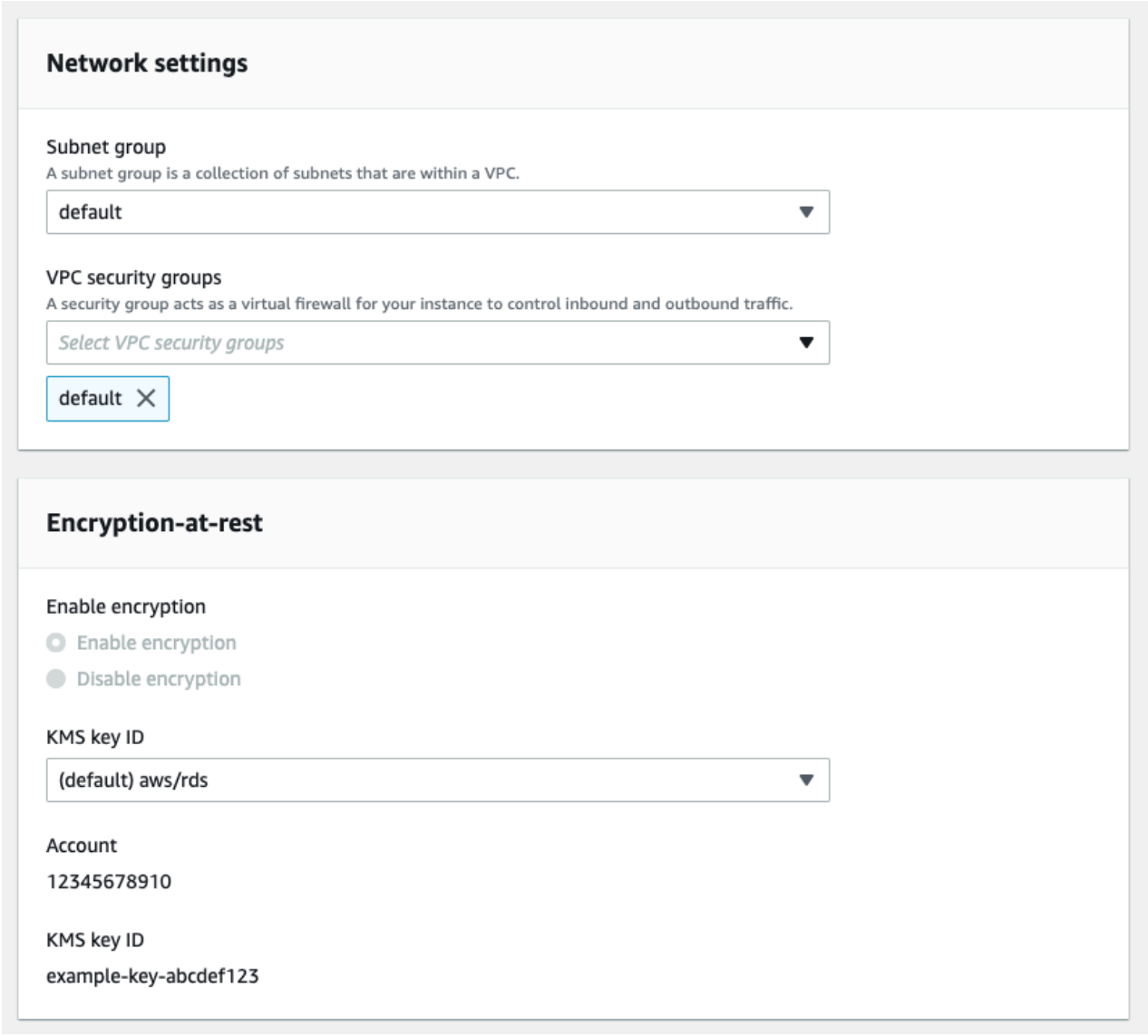
4. Na seção Configurações, faça o seguinte:
   a. Em Identificador de cluster, insira o nome que deseja dar ao cluster Amazon DocumentDB clonado.

b. Em Configuração da instância, selecione uma Classe de instância apropriada para seu cluster Amazon DocumentDB clonado.

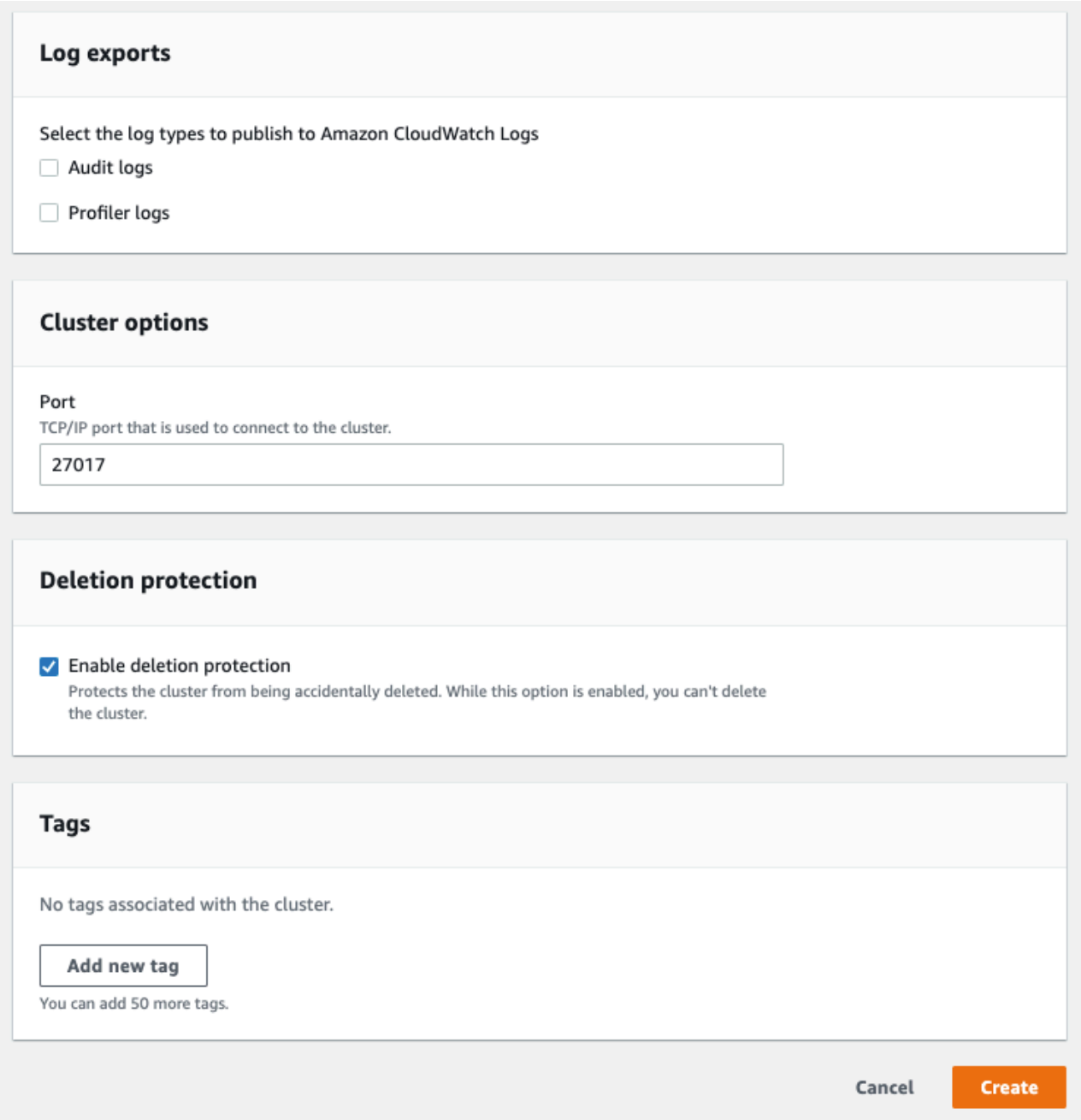


c. Em Configurações de rede, escolha um Grupo de sub-rede para seu caso de uso e os grupos de segurança VPC associados.

d. Pois Encryption-at-rest, se o cluster de origem (o cluster que está sendo clonado) tiver a criptografia habilitada, o cluster clonado também deverá ter a criptografia habilitada. Se esse cenário for verdadeiro, as opções Ativar criptografia permanecerão na cor cinza (desativadas), mas com a opção Ativar criptografia selecionada. Por outro lado, se o cluster de origem não estiver com a criptografia habilitada, as opções Ativar criptografia estarão disponíveis e será possível optar por ativar ou desativar a criptografia.



e. Conclua a nova configuração do clone do cluster selecionando os tipos de logs a serem exportados (opcional) e inserindo uma porta específica, usada para se conectar ao cluster e ativar a proteção contra a exclusão acidental do cluster (ativada por padrão).



f. Termine de inserir todas as configurações do clone do cluster Amazon DocumentDB. Para saber mais sobre as configurações de cluster e instância Amazon DocumentDB, consulte Gerenciar clusters do Amazon DocumentDB.

5. Escolha Criar clone para iniciar o clone do Amazon DocumentDB do cluster Amazon DocumentDB escolhido.

Ao ser criado, o clone é listado com seus outros clusters Amazon DocumentDB na seção Bancos de dados do console e exibe seu estado atual. O clone estará pronto para uso quando o estado for Disponível.

Using the AWS CLI

Usar o AWS CLI para clonar seu cluster Amazon DocumentDB envolve algumas etapas.

O `restore-db-cluster-to-point-in-time` AWS CLI comando que você usa resulta em um cluster vazio do Amazon DocumentDB com 0 instâncias do Amazon DocumentDB. Ou seja, o comando restaura apenas o cluster Amazon DocumentDB, não as instâncias desse cluster. Faça isso separadamente depois que o clone estiver disponível. As duas etapas do processo são:

1. Crie o clone usando o comando restore-db-cluster-to-pontual da CLI. Os parâmetros usados com esse comando controlam o tipo de capacidade e outros detalhes do cluster Amazon DocumentDB vazio (clone) sendo criado.
2. Crie a instância Amazon DocumentDB para o clone usando o comando create-db-instance da CLI, para recriar a instância no cluster restaurado.

Os comandos a seguir pressupõem que o AWS CLI esteja configurado com sua AWS região como padrão. Essa abordagem evita que você passe pelo nome `--region` em cada um dos comandos. Para obter mais informações, consulte Configurar a AWS CLI. Também é possível especificar `--region` em cada um dos comandos da CLI seguintes.

Criar o clone

Os parâmetros específicos que você passa para o comando `restore-db-cluster-to-point-in-time` da CLI variam. O que será transmitido depende do tipo de clone que você quer criar.

Use o procedimento a seguir para criar um clone provisionado do Amazon DocumentDB de um cluster provisionado.

Para criar um clone no mesmo modo de mecanismo que o cluster Amazon DocumentDB original

- Use o comando `restore-db-cluster-to-point-in-time` da CLI e especifique valores para os seguintes parâmetros:
  - `--db-cluster-identifier` — escolha um nome significativo para o clone. Nomeie o clone ao usar o comando restore-db-cluster-to-pontual da CLI.

- `--restore-type` — use `copy-on-write` para criar um clone do cluster DB de origem. Sem esse parâmetro, `restore-db-cluster-to-point-in-time` restaura o cluster Amazon DocumentDB em vez de criar um clone. O padrão para `restore-type` é `full-copy`.
- `--source-db-cluster-identifier` — use o nome do cluster Amazon DocumentDB de origem que deseja clonar.
- `--use-latest-restorable-time` — esse valor aponta para dados de volume restauráveis mais recentes para o clone. Esse parâmetro é necessário para `restore-type copy-on-write`. No entanto, você não pode usar o `restore-to-time parameter` com ele.

O exemplo a seguir mostra a criação de um clone chamado `my-clone` a partir de um cluster chamado `my-source-cluster`.

Para Linux, macOS ou 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
```

O comando retorna o objeto JSON que contém detalhes do clone. Verifique se o cluster clonado está disponível antes de tentar criar a instância DB para o seu clone. Para obter mais informações, consulte Verificar o status e obter detalhes do clone abaixo:

Verificar o status e obter detalhes do clone

É possível usar o comando a seguir para verificar o status do cluster dDB vazio recém-criado.

```
$ aws docdb describe-db-clusters --db-cluster-identifier my-clone --query '*[].
[Status]' --output text
```

Ou você pode obter o status e os outros valores necessários para criar a instância de banco de dados para seu clone usando a seguinte AWS CLI consulta:

Para Linux, macOS ou 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}"
```

Essa consulta retorna uma saída semelhante à saída abaixo.

```
[
  {
        "Status": "available",
        "Engine": "docdb",
        "EngineVersion": "4.0.0",
    }
]
```

Criar a instância do Amazon DocumentDB para seu clone

Use o comando create-db-instance da CLI para criar a instância primária DB para seu clone.

O parâmetro `--db-instance-class` é usado para provisionar apenas clusters Amazon DocumentDB.

Para Linux, macOS ou 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
```

Parâmetros a serem usados para clonagem

A tabela a seguir resume os vários parâmetros usados com `restore-db-cluster-to-point-in-time` para clonar clusters Amazon DocumentDB.

Parâmetro	Descrição
--source-db-cluster-identifier	Use o nome do cluster Amazon DocumentDB original que deseja clonar.
--db-cluster-identifier	Escolha um nome significativo para o clone. Nomeie seu clone com o comando `restore-db-cluster-to-point-in-time` . Em seguida, passe esse nome para o comando `create-db-instance` .
--restore-type	Especifique `copy-on-write` como `--restore-type` para criar um clone do cluster Amazon DocumentDB original em vez de restaurar o cluster original.
--use-latest-restorable-time	Esse valor aponta para os dados de volume restauráveis mais recentes para o clone.

Entender a tolerância a falhas de cluster do Amazon DocumentDB

Os clusters do Amazon DocumentDB são tolerantes a falhas por design. O volume de cada cluster abrange várias zonas de disponibilidade em uma única Região da AWS, e cada zona de disponibilidade contém uma cópia dos dados do volume do cluster. Esta funcionalidade significa que seu cluster pode tolerar a falha de uma Zona de Disponibilidade sem perder dados, com apenas uma breve interrupção do serviço.

Se a instância primária em um cluster falhar, o Amazon DocumentDB fará failover automaticamente para uma nova instância primária, de uma dessas duas maneiras:

- Ao promover uma réplica existente do Amazon DocumentDB para a nova instância principal e, em seguida, criar uma substituição para a réplica anterior. Um failover para a instância de réplica normalmente leva menos de 30 segundos. As operações de leitura e gravação poderão sofrer uma breve interrupção durante esse período. Para aumentar a disponibilidade do seu cluster, crie pelo menos uma ou mais réplicas do Amazon DocumentDB em duas ou mais zonas de disponibilidade diferentes.
- Criar uma nova instância principal. Isso só acontecerá se você não tiver uma instância de réplica em seu cluster, e pode levar alguns minutos para ser concluída.

Se o cluster tiver uma ou mais réplicas do Amazon DocumentDB, uma réplica do Amazon DocumentDB será promovida à instância principal durante um evento de falha. Um evento de falha resulta em uma breve interrupção, durante a qual as operações de leitura e gravação falham com uma exceção. No entanto, o serviço é restaurado normalmente em menos de 120 segundos, muitas vezes, em menos de 60 segundos. Para aumentar a disponibilidade do seu cluster, crie pelo menos uma ou mais réplicas do Amazon DocumentDB em duas ou mais zonas de disponibilidade diferentes.

É possível personalizar a ordem na qual suas réplicas do Amazon DocumentDB são promovidas à instância primária após uma falha atribuindo uma prioridade a cada réplica. As prioridades variam de 0, para a prioridade mais alta, a 15, para a prioridade mais baixa. Se a instância principal falhar, a réplica do Amazon DocumentDB com a prioridade mais alta será promovida à nova instância principal. É possível modificar a prioridade de uma réplica do Amazon DocumentDB a qualquer momento. Modificar a prioridade não desencadeia um failover. É possível usar a operação `modify-db-instance` com o parâmetro `--promotion-tier`. Para obter mais informações sobre como personalizar a prioridade de failover de uma instância, consulte Failover do Amazon DocumentDB.

A mesma prioridade pode ser compartilhada por mais de uma réplica do Amazon DocumentDB, o que resulta em níveis de promoção. Se duas ou mais réplicas do Amazon DocumentDB compartilharem a mesma prioridade, a réplica maior será promovida à principal. Se duas ou mais réplicas do Amazon DocumentDB compartilharem a mesma prioridade e o mesmo tamanho, uma réplica arbitrária no mesmo nível de promoção será promovida.

Se o cluster não contiver quaisquer réplicas do Amazon DocumentDB, a instância principal será recriada durante um evento de falha. Um evento de falha resulta em uma interrupção, durante a qual as operações de leitura e gravação falharão com uma exceção. O serviço é reestabelecido

quando a nova instância primária é criada, o que normalmente leva menos de 10 minutos. Promover uma réplica do Amazon DocumentDB à instância primária é muito mais rápido que criar uma nova instância primária.

Gerenciar instâncias do Amazon DocumentDB

Os tópicos a seguir fornecem informações para ajudar a gerenciar as instâncias do Amazon DocumentDB. Eles incluem detalhes sobre classes e status de instâncias e como criar, excluir e modificar uma instância.

Tópicos

- Determinar o status de uma instância
- Ciclo de vida da instância do Amazon DocumentDB
- Gerenciar classes de instância
- NVMe-backed instâncias

Determinar o status de uma instância

Para ver os status válidos da instância, seus significados e como determinar o status de suas instâncias, consulte Monitoramento do status de uma instância do Amazon DocumentDB.

Ciclo de vida da instância do Amazon DocumentDB

O ciclo de vida de uma instância do Amazon DocumentDB inclui a criação, a modificação, a manutenção e a atualização, a realização de backups e restaurações, a reinicialização e a exclusão dessa instância. Esta seção fornece informações sobre como concluir esses processos.

Tópicos

- Adicionar uma instância do Amazon DocumentDB a um cluster
- Descrever instâncias do Amazon DocumentDB
- Modificar uma instância do Amazon DocumentDB
- Reinicializar uma instância do Amazon DocumentDB
- Excluir uma instância do Amazon DocumentDB

Você pode criar uma nova instância do Amazon DocumentDB usando o. Console de gerenciamento da AWS ou o. AWS CLI Para adicionar uma instância a um cluster, o cluster deve estar no estado

disponível. Não é possível adicionar uma instância a um cluster que foi interrompido. Se o cluster foi interrompido, primeiro inicie-o, aguarde até que fique disponível e adicione uma instância. Para obter mais informações, consulte Interromper e iniciar um cluster do Amazon DocumentDB.

> **Note**
>
> Se você criar um cluster do Amazon DocumentDB usando o console, uma instância será criada automaticamente para você ao mesmo tempo. Se você deseja criar instâncias adicionais, use um dos procedimentos a seguir.

Adicionar uma instância do Amazon DocumentDB a um cluster

Using the Console de gerenciamento da AWS

Use o procedimento a seguir para criar uma instância para o cluster usando o console do Amazon DocumentDB.

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em. https://console.aws.amazon.com/docdb
2. No painel de navegação, escolha Clusters.

   > **Tip**
   >
   > Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

3. Para escolher o cluster ao qual você deseja adicionar uma instância, selecione o botão à esquerda do nome do cluster.
4. Escolha Ações e escolha Adicionar instâncias.
5. Na página Adicionar instância a: <nome-do-cluster>, repita as etapas a seguir para cada instância a ser adicionada ao cluster. É possível ter até 15.

   a. Identificador da instância: você pode inserir um identificador exclusivo para essa instância ou permitir que o Amazon DocumentDB forneça o identificador da instância com base no identificador do cluster.

      Restrições de nomenclatura da instância:

- O comprimento é de [1 a 63] letras, números ou hífens.
- O primeiro caractere deve ser uma letra.
- Não pode terminar com um hífen ou conter dois hifens consecutivos.
- Deve ser exclusivo para todas as instâncias do Amazon RDS, Neptune e Amazon DocumentDB por região. Conta da AWS

b. Classe da instância: na lista suspensa, escolha o tipo de instância desejado para essa instância.

c. Nível de promoção: na lista suspensa, escolha o nível de promoção da instância ou selecione Nenhuma preferência para permitir que o Amazon DocumentDB defina o nível de promoção para a instância. Números mais baixos significam maior prioridade. Para obter mais informações, consulte Controlar o destino de failover.

d. Para adicionar mais instâncias, escolha Adicionar outras instâncias e repita as etapas a, b e c.

6. Conclua a operação.

- Para adicionar instâncias ao seu cluster, escolha Criar.
- Para cancelar a operação, escolha Cancelar.

Leva alguns minutos para criar uma instância. Você pode usar o console ou AWS CLI visualizar o status da instância. Para obter mais informações, consulte Monitorar o status de uma instância.

Using the AWS CLI

Use a `create-db-instance` AWS CLI operação com os parâmetros a seguir para criar a instância primária para seu cluster.

- **`--db-instance-class`** — Obrigatório. A capacidade de computação e memória da instância, por exemplo, `db.m4.large`. Nem todas as classes de instâncias estão disponíveis em todas as Regiões da AWS.
- **`--db-instance-identifier`** — Obrigatório. Uma string que identifica a instância.

  Restrições de nomenclatura da instância:

  - O comprimento é de [1 a 63] letras, números ou hífens.
  - O primeiro caractere deve ser uma letra.
  - Não pode terminar com um hífen ou conter dois hifens consecutivos.

  - Deve ser exclusivo para todas as instâncias do Amazon RDS, Neptune e Amazon DocumentDB por região. Conta da AWS
- **`--engine`** — Obrigatório. Deve ser `docdb`.
- **`--availability-zone`** — Opcional. A zona de disponibilidade na qual você deseja que essa instância seja criada. Use esse parâmetro para localizar suas instâncias em diferentes zonas de disponibilidade para aumentar a tolerância a falhas. Para obter mais informações, consulte Alta disponibilidade e replicação do Amazon DocumentDB.
- **`--promotion-tier`** — Opcional. O nível de prioridade de failover dessa instância. Deve estar entre 0 e 15, com números mais baixos sendo prioridade mais alta. Para obter mais informações, consulte Controlar o destino de failover.

1. Primeiro, determine em quais zonas de disponibilidade você pode criar a instância.

   Se quiser especificar a zona de disponibilidade antes de criar a instância, execute o seguinte comando para determinar quais zonas de disponibilidade estão disponíveis para o cluster do Amazon DocumentDB.

   Para Linux, macOS ou Unix:

   ```
   aws docdb describe-db-clusters \
         --query 'DBClusters[*].[DBClusterIdentifier,AvailabilityZones[*]]'
   ```

   Para Windows:

   ```
   aws docdb describe-db-clusters ^
         --query 'DBClusters[*].[DBClusterIdentifier,AvailabilityZones[*]]'
   ```

   A saída dessa operação é semelhante à seguinte.

   ```
   [
       [
           "sample-cluster",
           [
               "us-east-1c",
               "us-east-1b",
               "us-east-1a"
           ]
       ]
   ```

```
]
```

2. Segundo, determine quais classes de instância você pode criar na região.

   Para determinar quais classes de instância estão disponíveis na sua região, execute o seguinte comando. Na saída, escolha uma classe de instância para a instância que deseja adicionar ao cluster do Amazon DocumentDB.

   Para Linux, macOS ou 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'
```

   A saída dessa operação é semelhante à seguinte.

```
[
    "db.r5.16xlarge",
    "db.r5.2xlarge",
    "db.r5.4xlarge",
    "db.r5.8xlarge",
    "db.r5.large",
    "db.r5.xlarge"
]
```

3. Por último, adicione uma instância ao cluster do Amazon DocumentDB.

   Para adicionar uma instância ao cluster do Amazon DocumentDB, execute o seguinte comando.

   Para Linux, macOS ou 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
```

A saída dessa operação é semelhante à seguinte.

```
{
    "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"
    }
}
```

Leva alguns minutos para criar a instância. Você pode usar o console ou AWS CLI visualizar o status da instância. Para obter mais informações, consulte Monitoramento do status de uma instância do Amazon DocumentDB.

Descrever instâncias do Amazon DocumentDB

É possível usar o Console de gerenciamento do Amazon DocumentDB ou a AWS CLI para ver detalhes como endpoints de conexão, VPCs de grupos de segurança, autoridade de certificação e grupos de parâmetros pertencentes às suas instâncias do Amazon DocumentDB.

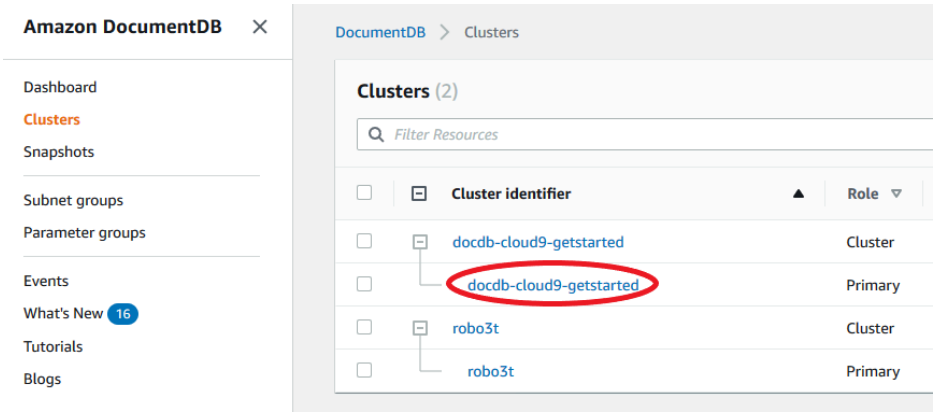
Using the Console de gerenciamento da AWS

Para visualizar os detalhes de suas instâncias usando o Console de gerenciamento da AWS, siga as etapas abaixo.

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em. https://console.aws.amazon.com/docdb
2. No painel de navegação, escolha Clusters.

> **Tip**
>
> Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

3. Na caixa de navegação Clusters, você verá a coluna Identificador do Cluster. Suas instâncias estão listadas em clusters, semelhante à captura de tela a seguir.



4. Na lista de instâncias, escolha o nome da instância da qual você deseja ver os detalhes. As informações sobre a instância são organizadas nos seguintes agrupamentos:

   - Resumo: informações gerais sobre a instância, incluindo a versão do mecanismo, a classe, o status e todas as manutenções pendentes.
   - Conectividade e segurança: a seção Conecte-se lista os endpoints de conexão para se conectar a essa instância com o shell do mongo ou com uma aplicação. A seção Grupos de segurança lista os grupos de segurança associados a essa instância e suas descrições e ID da VPC.
   - Configuração: a seção Detalhes lista as configurações e o status da instância, incluindo o nome do recurso da Amazon (ARN), o endpoint, o perfil, a classe e a autoridade

certificadora da instância. Ela também lista as configurações de segurança e de rede da instância e as informações de backup. A seção Detalhes do cluster lista os detalhes do cluster ao qual essa instância pertence. A seção Instâncias de cluster lista todas as instâncias que pertencem ao cluster com cada função e status de grupo de parâmetros de cluster de cada instância.

> **Note**
>
> É possível modificar o cluster associado à sua instância selecionando Modificar ao lado do cabeçalho Detalhes do cluster. Para obter mais informações, consulte Modificar um cluster do Amazon DocumentDB.

- Monitoramento — as métricas de CloudWatch registros dessa instância. Para obter mais informações, consulte Monitorando o Amazon DocumentDB com CloudWatch.
- Eventos e tags: a seção Eventos recentes lista os eventos recentes dessa instância. O Amazon DocumentDB mantém um registro de eventos que se relacionam a seus clusters, instâncias, snapshots, grupos de segurança e grupos de parâmetros do cluster. Essas informações incluem a data, a hora e a mensagem associadas a cada evento. A seção Tags lista as tags anexadas a este cluster. Para obter mais informações, consulte Marcação de recursos do Amazon DocumentDB.

Using the AWS CLI

Para visualizar os detalhes de suas instâncias do Amazon DocumentDB usando o AWS CLI, use o `describe-db-clusters` comando conforme mostrado nos exemplos a seguir. Para obter mais informações, consulte `DescribeDBInstances` na Referência de API para gerenciamento de recursos do Amazon DocumentDB.

> **Note**
>
> Para determinados atributos de gerenciamento, como o gerenciamento do ciclo de vida de cluster e instâncias, o Amazon DocumentDB aproveita a tecnologia operacional compartilhada com o Amazon RDS. O parâmetro de filtro `filterName=engine,Values=docdb` só retorna clusters do Amazon DocumentDB.

1. Liste todas as instâncias do Amazon DocumentDB.

O AWS CLI código a seguir lista os detalhes de todas as instâncias do Amazon DocumentDB em uma região.

Para Linux, macOS ou Unix:

```
aws docdb describe-db-instances \
    --filter Name=engine,Values=docdb
```

Para Windows:

```
aws docdb describe-db-instances \
    --filter Name=engine,Values=docdb
```

2. Listar todos os detalhes de uma instância especificada do Amazon DocumentDB

O código a seguir lista os detalhes para `sample-cluster-instance`. Incluir o parâmetro `--db-instance-identifier` com o nome de uma instância restringe a saída às informações sobre essa instância específica.

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante à seguinte.

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

Modificar uma instância do Amazon DocumentDB

Você pode modificar sua instância do Amazon DocumentDB usando o Console de gerenciamento da AWS ou o. AWS CLI Para modificar uma instância, ela deve estar no estado disponível. Você não pode modificar uma instância que foi interrompida. Se o cluster foi interrompido, primeiro inicie-o, aguarde até que a instância fique disponível e faça as modificações desejadas. Para obter mais informações, consulte Interromper e iniciar um cluster do Amazon DocumentDB.

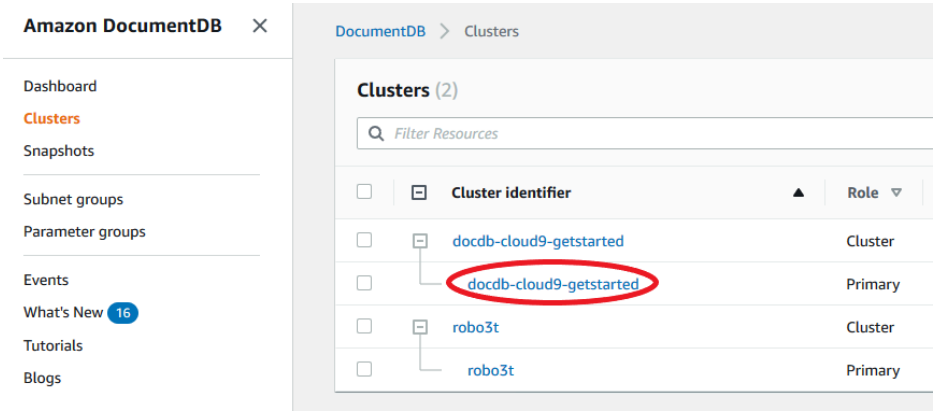
Using the Console de gerenciamento da AWS

Para modificar uma instância específica do Amazon DocumentDB usando o console, siga as etapas a seguir.

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em. https://console.aws.amazon.com/docdb
2. No painel de navegação, escolha Clusters.

> **Tip**
>
> Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

3. Na caixa de navegação Clusters, você verá a coluna Identificador do Cluster. Suas instâncias estão listadas em clusters, semelhante à captura de tela a seguir.



4. Marque a caixa à esquerda da instância que você deseja modificar.
5. Escolha Ações e, em seguida, Modificar.
6. No painel Modificar instância: <nome-da-instância>, faça as alterações desejadas. É possível fazer as alterações a seguir:

   - Especificações da instância: o identificador e a classe da instância. Restrições de nomenclatura de identificador de instância:
     - Identificador da instância — insira um nome exclusivo para todas as instâncias de sua propriedade Conta da AWS na região atual. O identificador da instância deve conter [1 a 63] caracteres alfanuméricos ou hifens, ter uma letra como primeiro caractere e não pode terminar com um hífen nem conter dois hifens consecutivos.

- Classe de instância: no menu suspenso, selecione uma classe de instância para sua instância do Amazon DocumentDB. Para obter mais informações, consulte Gerenciar classes de instância.

- Autoridade de certificação: certificado do servidor para essa instância. Para obter mais informações, consulte Atualizar seus certificados TLS do Amazon DocumentDB.
- Failover: durante o failover, a instância com o nível de promoção mais alto será promovida para primária. Para obter mais informações, consulte Failover do Amazon DocumentDB.
- Manutenção: a janela de manutenção na qual modificações ou patches pendentes são aplicados às instâncias no cluster.

7. Ao concluir, escolha Continuar para ver um resumo das alterações.
8. Depois de verificar suas alterações, é possível aplicá-las imediatamente, ou durante a próxima janela de manutenção em Programação de modificações. Selecione Modificar instância para salvar as alterações. Como alternativa, você pode escolher Cancelar para descartar as alterações.

Levará alguns minutos para que suas alterações sejam aplicadas. É possível usar a instância somente quando seu status for disponível. É possível monitorar o status da instância usando o console ou a AWS CLI. Para obter mais informações, consulte Monitoramento do status de uma instância do Amazon DocumentDB.

Using the AWS CLI

Para modificar uma instância específica do Amazon DocumentDB usando o AWS CLI, use o `modify-db-instance` com os seguintes parâmetros. Para obter mais informações, consulte ModifyDBInstance. O código a seguir modifica a classe de instância para `db.r5.large` para a instância `sample-instance`.

Parâmetros

- **`--db-instance-identifier`** — Obrigatório. O identificador da instância a ser modificada.
- **`--db-instance-class`** — Opcional. A nova capacidade de computação e memória da instância. Por exemplo, `db.r5.large`. Nem todas as classes de instância estão disponíveis em todos os Regiões da AWS. Se você modificar a classe da instância, ocorrerá uma interrupção durante a alteração. A alteração será aplicada durante a próxima janela de manutenção, a menos que `ApplyImmediately` seja especificado como verdadeiro para essa solicitação.

- **`--apply-immediately`** ou **`--no-apply-immediately`**: opcional. Especifica se essa modificação deve ser aplicada imediatamente ou se deve aguardar até a próxima janela de manutenção. Se esse parâmetro for omitido, a modificação será executada durante a próxima janela de manutenção.

Example

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante à seguinte.

```
{
    "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"
```

```
            ]
        }
    ]
}
```

Leva alguns minutos para que suas modificações sejam aplicadas. É possível usar a instância somente quando seu status for disponível. Você pode monitorar o status da instância usando o Console de gerenciamento da AWS ou AWS CLI. Para obter mais informações, consulte Monitoramento do status de uma instância do Amazon DocumentDB.

Reinicializar uma instância do Amazon DocumentDB

Ocasionalmente, você pode precisar reinicializar sua instância do Amazon DocumentDB, geralmente por motivos de manutenção. Se você fizer determinadas alterações, como alterar o grupo de parâmetros de cluster associado a um cluster, será necessário reinicializar as instâncias no cluster para que as alterações sejam implantadas. Você pode reinicializar uma instância específica usando o. Console de gerenciamento da AWS ou o. AWS CLI

Reinicializar uma instância reinicia o serviço de mecanismo de banco de dados. A reinicialização resulta em uma interrupção momentânea, durante a qual o status da instância é definido como `rebooting`. Um evento do Amazon DocumentDB é criado quando a reinicialização é concluída.

A reinicialização de uma instância não resulta em um failover. Para fazer o failover de um cluster do Amazon DocumentDB, use Console de gerenciamento da AWS a ou AWS CLI a operação. `failover-db-cluster` Para obter mais informações, consulte Failover do Amazon DocumentDB.

Você não poderá reinicializar sua instância se ela não estiver no estado disponível. Seu banco de dados pode se tornar indisponível por vários motivos, como uma modificação solicitada anteriormente ou uma ação no intervalo de manutenção. Para obter mais informações sobre os estados da instância, consulte Monitoramento do status de uma instância do Amazon DocumentDB.

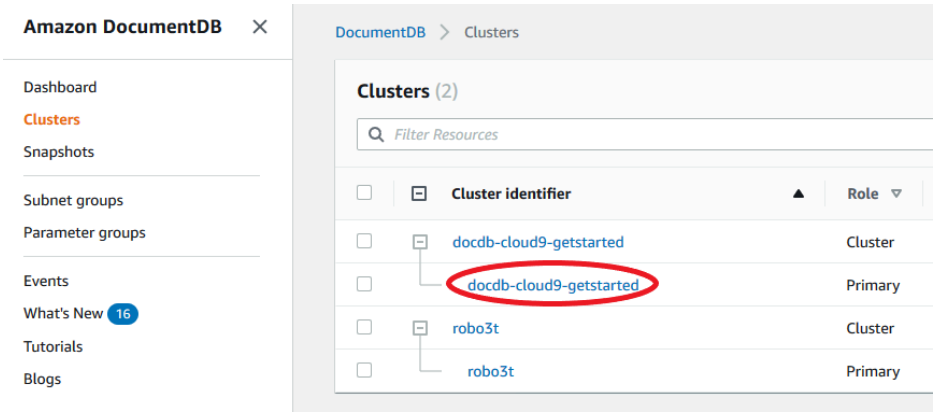
Using the Console de gerenciamento da AWS

O procedimento a seguir reinicializa uma instância especificada usando o console.

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em. https://console.aws.amazon.com/docdb
2. No painel de navegação, escolha Clusters.

> **Tip**
>
> Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

3. Na caixa de navegação Clusters, você verá a coluna Identificador do Cluster. Suas instâncias estão listadas em clusters, semelhante à captura de tela a seguir.



4. Marque a caixa à esquerda da instância que você deseja reinicializar.
5. Escolha Ações, escolha Reinicializar e depois Reinicializar para confirmar a reinicialização.

Demora alguns minutos para sua instância reinicializar. É possível usar a instância somente quando seu status for disponível. É possível monitorar o status da instância usando o console ou a AWS CLI. Para obter mais informações, consulte Monitoramento do status de uma instância do Amazon DocumentDB.

Using the AWS CLI

Para reinicializar uma instância do Amazon DocumentDB, use a operação `reboot-db-instance` com o parâmetro `--db-instance-identifier`. Esse parâmetro especifica o identificador da instância a ser reinicializada.

O código a seguir reinicializa a instância `sample-instance`.

Example

Para Linux, macOS ou Unix:

```
aws docdb reboot-db-instance \
```

```
       --db-instance-identifier sample-instance
```

Para Windows:

```
aws docdb reboot-db-instance ^
       --db-instance-identifier sample-instance
```

A saída dessa operação é semelhante à seguinte.

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

Demora alguns minutos para sua instância reinicializar. É possível usar a instância somente quando seu status for disponível. É possível monitorar o status da instância usando o console ou a AWS CLI. Para obter mais informações, consulte Monitoramento do status de uma instância do Amazon DocumentDB.

Excluir uma instância do Amazon DocumentDB

Você pode excluir sua instância do Amazon DocumentDB usando o Console de gerenciamento da AWS ou o. AWS CLI Para excluir uma instância, ela deve estar no estado disponível. Você não pode excluir uma instância que foi interrompida. Se o cluster do Amazon DocumentDB que contém a instância for interrompido, primeiro inicie o cluster, aguarde até que a instância fique disponível e, depois, exclua-a. Para obter mais informações, consulte Interromper e iniciar um cluster do Amazon DocumentDB.

> **Note**
>
> O Amazon DocumentDB armazena todos os dados no volume do cluster. Os dados persistem nesse volume de cluster, mesmo se você remover todas as instâncias do cluster. Se precisar acessar os dados novamente, adicione uma instância ao cluster a qualquer momento e continue de onde parou.

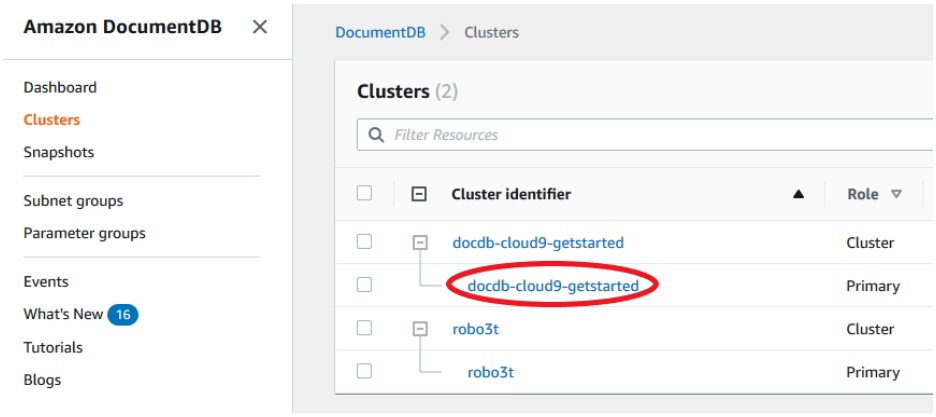
Using the Console de gerenciamento da AWS

O procedimento a seguir excluir uma instância do Amazon DocumentDB especificada usando o console.

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em. https://console.aws.amazon.com/docdb
2. No painel de navegação, escolha Clusters.

   > **Tip**
   >
   > Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

3. Na caixa de navegação Clusters, você verá a coluna Identificador do Cluster. Suas instâncias estão listadas em clusters, semelhante à captura de tela a seguir.



4. Marque a caixa à esquerda da instância que você deseja excluir.
5. Selecione Ações e escolha Excluir.

   1. Se você estiver excluindo a última instância no cluster:

- Criar snapshot final do cluster? — Escolha Sim se você quiser criar um snapshot final antes do cluster ser excluído. Caso contrário, escolha Não.
- Nome do snapshot final: se você escolher criar um snapshot final, insira o identificador do novo snapshot do cluster.
- Excluir instância <nome da instância>? — Insira a frase excluir cluster inteiro no campo para confirmar a exclusão.

2. Se você não estiver excluindo a última instância no cluster:
   - Excluir instância <nome da instância>? — Insira a frase exclua-me no campo para confirmar a exclusão.

6. Selecione Excluir para excluir a instância.

Leva alguns minutos para uma instância ser excluída. Para monitorar o status de uma instância, consulte Monitoramento do status de uma instância do Amazon DocumentDB.

Using the AWS CLI

O procedimento a seguir exclui uma instância do Amazon DocumentDB usando a AWS CLI.

1. Primeiro, determine quantas instâncias estão no cluster do Amazon DocumentDB:

   Para determinar quantas instâncias estão no cluster, execute o comando `describe-db-clusters` da seguinte forma.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].
[DBClusterIdentifier,DBClusterMembers[*].DBInstanceIdentifier]'
```

   A saída dessa operação é semelhante à seguinte.

```
[
    [
        "sample-cluster",
        [
            "sample-instance-1",
            "sample-instance-2"
        ]
    ]
```

```
]
```

2. Se houver mais de uma instância no cluster do Amazon DocumentDB:

   Para excluir uma instância do Amazon DocumentDB especificada, use o comando `delete-db-instance` com o parâmetro `--db-instance-identifier`, conforme mostrado abaixo. Leva alguns minutos para uma instância ser excluída. Para monitorar o status de uma instância, consulte Monitoramento do status de uma instância do Amazon DocumentDB.

```
aws docdb delete-db-instance \
       --db-instance-identifier sample-instance-2
```

   A saída dessa operação é semelhante à seguinte.

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "sample-instance-2",
        "DBInstanceClass": "db.r5.large",
        "Engine": "docdb",
        "DBInstanceStatus": "deleting",
        "Endpoint": {
            "Address": "sample-instance-2.node.us-east-1.docdb.amazonaws.com",
            "Port": 27017,
            "HostedZoneId": "ABCDEFGHIJKLM"
        },
        "InstanceCreateTime": "2020-03-27T08:05:56.314Z",
        "PreferredBackupWindow": "02:00-02:30",
        "BackupRetentionPeriod": 1,
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-abcd0123",
                "Status": "active"
            }
        ],
        "AvailabilityZone": "us-east-1c",
        "DBSubnetGroup": {
            "DBSubnetGroupName": "default",
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-6242c31a",
            "SubnetGroupStatus": "Complete",
            "Subnets": [
                {
```

```
                    "SubnetIdentifier": "subnet-abcd0123",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1a"
                    },
                    "SubnetStatus": "Active"
                },
                {
                    "SubnetIdentifier": "subnet-wxyz0123",
                    "SubnetAvailabilityZone": {
                        "Name": "us-east-1b"
                    },
                    "SubnetStatus": "Active"
                }
            ]
        },
        "PreferredMaintenanceWindow": "sun:06:53-sun:07:23",
        "PendingModifiedValues": {},
        "EngineVersion": "3.6.0",
        "AutoMinorVersionUpgrade": true,
        "PubliclyAccessible": false,
        "DBClusterIdentifier": "sample-cluster",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:<accountID>:key/sample-key",
        "DbiResourceId": "db-ABCDEFGHIJKLMNOPQRSTUVWXYZ",
        "CACertificateIdentifier": "rds-ca-2019",
        "PromotionTier": 1,
        "DBInstanceArn": "arn:aws:rds:us-east-1:<accountID>:db:sample-instance-2",
        "EnabledCloudwatchLogsExports": [
            "profiler"
        ]
    }
}
```

3. Se a instância que deseja excluir for a última instância no cluster do Amazon DocumentDB:

   Se você excluir a última instância em um cluster do Amazon DocumentDB, você também excluirá esse cluster e os snapshots automáticos e backups contínuos associados a ele.

   Para excluir a última instância no cluster, é possível excluí-lo e opcionalmente criar um snapshot final. Para obter mais informações, consulte Excluir um cluster do Amazon DocumentDB.

Proteção contra exclusão

A exclusão da última instância em um cluster do Amazon DocumentDB também excluirá o cluster, os snapshots automáticos e os backups contínuos associados a ele. O Amazon DocumentDB impõe proteção contra exclusão para um cluster, independentemente de você realizar a operação de exclusão usando o. ou o. Console de gerenciamento da AWS AWS CLI Se a proteção contra exclusão estiver habilitada, não será possível excluir um cluster.

Para excluir um cluster com a projeção contra exclusão habilitada, primeiro é necessário modificar o cluster e desabilitar a proteção contra exclusão. Para obter mais informações, consulte Excluir um cluster do Amazon DocumentDB.

Gerenciar classes de instância

A classe de instância determina a capacidade de computação e de memória de uma instância do Amazon DocumentDB (compatível com MongoDB). A classe de instância da qual você precisa depende dos requisitos de energia e memória de processamento.

O Amazon DocumentDB oferece suporte às famílias de classes de instância R4, R5, R6G, R8G, T3 e T4G. Essas classes são de instâncias da geração atual que são otimizadas para aplicações com uso intensivo de memória. Para obter as especificações dessas classes, consulte Especificações de classe de instância.

Tópicos

- Determinar uma classe de instância
- Alterar a classe de uma instância
- Classes de instância compatíveis por região
- Especificações de classe de instância

Determinar uma classe de instância

Para determinar a classe de uma instância, você pode usar a `describe-db-instances` AWS CLI operação Console de gerenciamento da AWS ou.

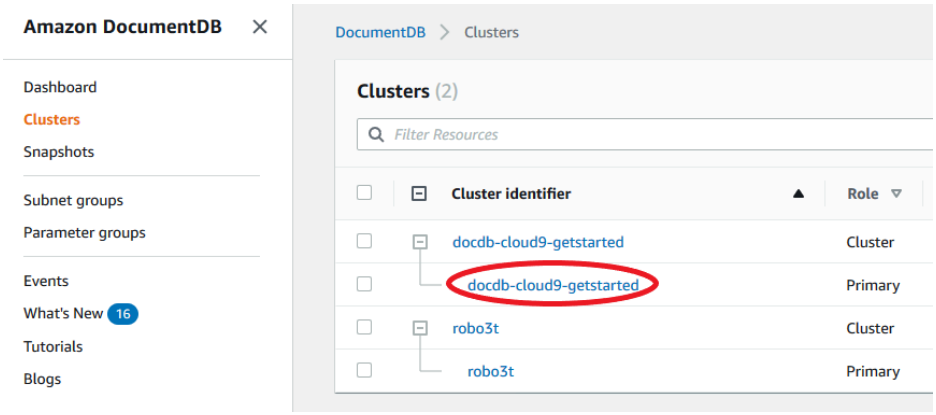
Using the Console de gerenciamento da AWS

Para determinar a classe de instância para as instâncias do seu cluster, execute as etapas a seguir no console.

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Clusters para encontrar a instância do seu interesse.

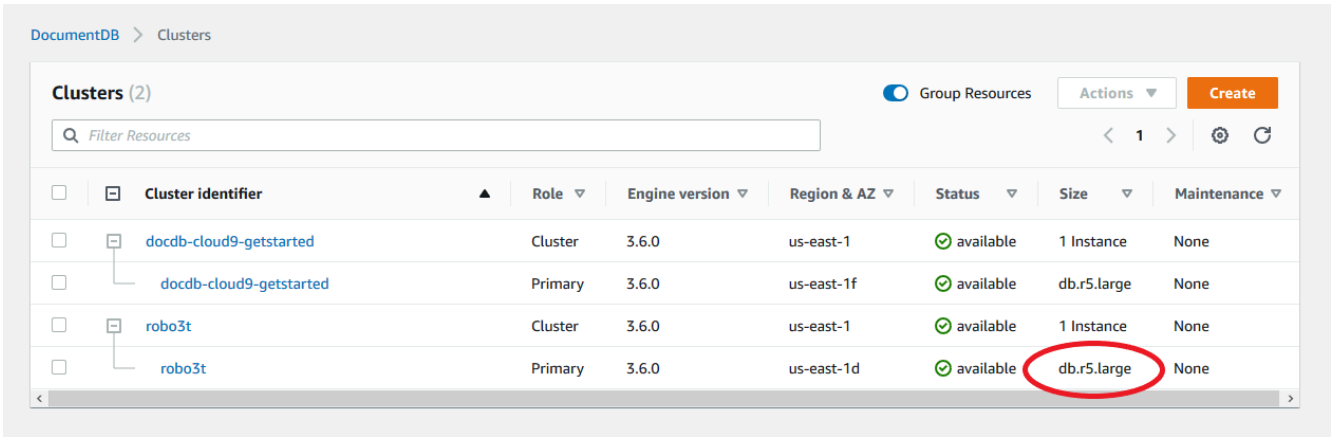
> **Tip**
>
> Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

3. Na caixa de navegação Clusters, você verá a coluna Identificador do Cluster. Suas instâncias estão listadas em clusters, de forma semelhante à captura de tela a seguir.



4. Na lista de instâncias, expanda o cluster para encontrar as instâncias de seu interesse. Encontre a instância desejada. Em seguida, verifique a coluna Tamanho da linha da instância para visualizar a classe de instância.

   Na imagem a seguir, a classe da instância `robo3t` é `db.r5.4xlarge`.



Using the AWS CLI

Para determinar a classe de uma instância usando o AWS CLI, use a `describe-db-instances` operação com os parâmetros a seguir.

- **`--db-instance-identifier`**: opcional. Especifica a instância para a qual você deseja localizar a classe da instância. Se o parâmetro for omitido, `describe-db-instances` retornará uma descrição para até 100 das suas instâncias.
- **`--query`**: opcional. Especifica os membros da instância a serem incluídos nos resultados. Se esse parâmetro for omitido, todos os membros da instância serão retornados.

Example

O exemplo a seguir localiza o nome e a classe da instância para a instância `sample-instance-1`.

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante à seguinte.

```
[
    [
        "sample-instance-1",
        "db.r5.large"
    ]
```

Example

O exemplo a seguir localiza o nome e a classe de até 100 instâncias do Amazon DocumentDB.

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante à seguinte.

```
[
    [
        "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 obter mais informações, consulte Descrever instâncias do Amazon DocumentDB.

Alterar a classe de uma instância

Você pode alterar a classe de instância da sua instância usando o Console de gerenciamento da AWS ou AWS CLI o. Para obter mais informações, consulte Modificar uma instância do Amazon DocumentDB.

Classes de instância compatíveis por região

O Amazon DocumentDB é compatível com as classes de instância a seguir:

- R8G—Última geração de instâncias otimizadas para memória alimentadas por processadores Arm-based AWS Graviton4 que oferecem desempenho até 30% melhor em relação às instâncias R6G.
- R6G— Memory-optimized instâncias alimentadas por processadores Arm-based AWS Graviton2 que oferecem desempenho até 30% melhor em relação às instâncias R5 a um custo 5% menor.
- R6GD— instâncias Memory-optimized R6G com armazenamento em Solid-State drive (SSD) local baseado em memória não volátil expressa (NVMe) para dados efêmeros.
- R5— Memory-optimized instâncias que oferecem desempenho até 100% melhor em relação às instâncias R4 pelo mesmo custo de instância.
- R4: geração anterior de instâncias otimizadas para memória.
- T4G— tipo de instância de uso geral intermitente de Latest-generation baixo custo, alimentado por processadores Arm-based AWS Graviton2, que fornece um nível básico de desempenho da CPU, oferecendo uma relação preço/desempenho até 35% melhor em relação às instâncias T3 e ideal para executar aplicativos com uso moderado da CPU que apresentam picos temporários de uso.
- T3: instância de uso geral de baixo custo com capacidade de expansão que fornecem um nível de linha de base de performance de CPU com a capacidade de expansão para uso de CPU a qualquer momento e pelo tempo necessário.

Para obter especificações dessas classes de instância, consulte Especificações de classe de instância.

> **Note**
>
> I/O-Optimized o armazenamento está disponível somente nas versões 5.0 e 8.0 do mecanismo (clusters baseados em instâncias).

Uma classe de instância específica pode ou não ser compatível em uma região. A tabela a seguir especifica quais classes de instância são compatíveis com o Amazon DocumentDB em cada região.

Classes de instância compatíveis por região

Região	R8G	R6GD	R6G	R5	R4	T4G	T3	Sem servidor
Leste dos EUA (Ohio)	Compatível	Compatível	Compatível	Compatível	Compatível	Compatível	Compatível	Compatível
Leste dos EUA (Norte da Virgínia)	Compatível	Compatível	Compatível	Compatível	Compatível	Compatível	Compatível	Compatível
Oeste dos EUA (Oregon)	Compatível	Compatível	Compatível	Compatível	Compatível	Compatível	Compatível	Compatível
África (Cidade do Cabo)			Compatível	Compatível		Compatível	Compatível	Compatível
América do Sul (São Paulo)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
Ásia-Pacífico (Hong Kong)	Compatível		Compatível	Compatível		Compatível	Compatível	Compatível
Ásia-Pacífico (Hyderabad)			Compatível	Compatível		Compatível	Compatível	Compatível
Ásia-Pacífico (Malásia)			Compatível			Compatível	Compatível	Compatível
Ásia-Pacífico (Mumbai)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
Ásia-Pacífico (Osaka)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
Ásia-Pacífico (Seul)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível

Região	R8G	R6GD	R6G	R5	R4	T4G	T3	Sem servidor
Ásia-Pacífico (Sydney)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
Ásia-Pacífico (Jacarta)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
Ásia-Pacífico (Melbourne)			Compatível	Compatível		Compatível	Compatível	Compatível
Ásia-Pacífico (Singapura)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
Ásia-Pacífico (Tailândia)			Compatível			Compatível	Compatível	Compatível
Ásia-Pacífico (Tóquio)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
Canadá (Central)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
Oeste do Canadá (Calgary)			Compatível	Compatível		Compatível	Compatível	Compatível
Europa (Frankfurt)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
Europa (Zurique)		Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
Europa (Irlanda)	Compatível	Compatível	Compatível	Compatível	Compatível	Compatível	Compatível	Compatível
Europa (Londres)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível

Região	R8G	R6GD	R6G	R5	R4	T4G	T3	Sem servidor
Europa (Milão)			Compatível	Compatível		Compatível	Compatível	Compatível
Europa (Paris)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
Europa (Espanha)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
Europa (Estocolmo)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
México (Centro)			Compatível			Compatível	Compatível	Compatível
Oriente Médio (Emirados Árabes Unidos)			Compatível	Compatível		Compatível	Compatível	Compatível
China (Pequim)	Compatível	Compatível	Compatível	Compatível		Compatível	Compatível	Compatível
China (Ningxia)	Compatível		Compatível	Compatível		Compatível	Compatível	Compatível
Israel (Tel Aviv)			Compatível	Compatível		Compatível	Compatível	Compatível
AWS GovCloud (US-West)	Compatível	Compatível	Compatível	Compatível			Compatível	Compatível
AWS GovCloud (US-East)		Compatível	Compatível	Compatível		Compatível	Compatível	Compatível

Especificações de classe de instância

A tabela a seguir fornece detalhes das classes de instância do Amazon DocumentDB, incluindo quais tipos de instância são aceitos em cada classe. É possível encontrar explicações para cada coluna da tabela abaixo da tabela.

Classe de instância	vCPU[1]	Memória (GiB)[2]	Cache hierárquico SSD NVMe (GiB) [3]	Temperatura máxima de armazenamento (GiB) 4	Largura de banda de linha de base / rajada (Gbps) [5]	Motores de apoio [6]
R8G — Classe de Memory-Optimized instância de geração atual baseada em Graviton4 O R8G é a mais nova família de instâncias (Graviton4), disponível somente nas versões 5.0 e 8.0 do motor.						
`db.r8g.large`	2	16	-	30	0.937/12.5	5.0.0 e 8.0.0
`db.r8g.xlarge`	4	32	-	60	1.875/12.5	5.0.0 e 8.0.0
`db.r8g.2xlarge`	8	64	-	121	3.75/15.0	5.0.0 e 8.0.0
`db.r8g.4xlarge`	16	128	-	243	7.5/15.0	5.0.0 e 8.0.0
`db.r8g.8xlarge`	32	256	-	488	15	5.0.0 e 8.0.0
`db.r8g.12xlarge`	48	384	-	732	22	5.0.0 e 8.0.0
`db.r8g.16xlarge`	64	512	-	987	30	5.0.0 e 8.0.0

Classe de instância	vCPU[1]	Memória (GiB)[2]	Cache hierárquico SSD NVMe (GiB) [3]	Temperatura máxima de armazenamento (GiB) 4	Largura de banda de linha de base / rajada (Gbps) [5]	Motores de apoio [6]
db.r8g.24xlarge	96	768	-	1484	40	5.0.0 e 8.0.0
db.r8g.48xlarge	192	1536	-	2967	50	5.0.0 e 8.0.0
R6G — Classe de Memory-Optimized instância de geração atual baseada em Graviton2						
db.r6g.large	2	16	-	32	0,75/10	4.0.0, 5.0.0 e 8.0.0
db.r6g.xlarge	4	32	-	63	1,25/10	4.0.0, 5.0.0 e 8.0.0
db.r6g.2xlarge	8	64	-	126	2,5/10	4.0.0, 5.0.0 e 8.0.0
db.r6g.4xlarge	16	128	-	252	5,0/10	4.0.0, 5.0.0 e 8.0.0
db.r6g.8xlarge	32	256	-	504	12	4.0.0, 5.0.0 e 8.0.0
db.r6g.12xlarge	48	384	-	756	20	4.0.0, 5.0.0 e 8.0.0
db.r6g.16xlarge	64	512	-	1008	25	4.0.0, 5.0.0 e 8.0.0
R6GD — Classe de NVMe-backed instância de geração atual baseada em Graviton2						

Classe de instância	vCPU[1]	Memória (GiB)[2]	Cache hierárquico SSD NVMe (GiB) [3]	Temperatura máxima de armazenamento (GiB) 4	Largura de banda de linha de base / rajada (Gbps) [5]	Motores de apoio [6]
db.r6gd.xlarge	4	32	173	64	1,25/10	5.0.0 e 8.0.0
db.r6gd.2xlarge	8	64	346	128	2,5/10	5.0.0 e 8.0.0
db.r6gd.4xlarge	16	128	694	256	5,0/10	5.0.0 e 8.0.0
db.r6gd.8xlarge	32	256	1388	512	12	5.0.0 e 8.0.0
db.r6gd.12xlarge	48	384	2082	768	20	5.0.0 e 8.0.0
db.r6gd.16xlarge	64	512	2776	1024	25	5.0.0 e 8.0.0
R5 — Classe de Memory-Optimized instância da geração anterior						
db.r5.large	2	16	-	31	0,75/10	3.6.0, 4.0.0, 5.0.0 e 8.0.0
db.r5.xlarge	4	32	-	62	1,25/10	3.6.0, 4.0.0, 5.0.0 e 8.0.0
db.r5.2xlarge	8	64	-	124	2,5/10	3.6.0, 4.0.0, 5.0.0 e 8.0.0

Classe de instância	vCPU[1]	Memória (GiB)[2]	Cache hierárquico SSD NVMe (GiB) [3]	Temperatura máxima de armazenamento (GiB) 4	Largura de banda de linha de base / rajada (Gbps) [5]	Motores de apoio [6]
db.r5.4xlarge	16	128	-	249	5,0/10	3.6.0, 4.0.0, 5.0.0 e 8.0.0
db.r5.8xlarge	32	256	-	504	10	3.6.0, 4.0.0, 5.0.0 e 8.0.0
db.r5.12xlarge	48	384	-	748	12	3.6.0, 4.0.0, 5.0.0 e 8.0.0
db.r5.16xlarge	64	512	-	1008	20	3.6.0, 4.0.0, 5.0.0 e 8.0.0
db.r5.24xlarge	96	768	-	1500	25	3.6.0, 4.0.0, 5.0.0 e 8.0.0
R4 — Classe de Memory-Optimized instância da geração anterior As instâncias R4 são suportadas somente na versão 3.6 do mecanismo. Observe que o Amazon DocumentDB 3.6 atinge o fim do suporte padrão em 30 de março de 2026. O suporte estendido (pago) está disponível até março de 2029.						
db.r4.large	2	15.25	-	30	0,75/10	3.6.0 somente
db.r4.xlarge	4	30.5	-	60	1,25/10	3.6.0 somente

Classe de instância	vCPU[1]	Memória (GiB)[2]	Cache hierárquico SSD NVMe (GiB) [3]	Temperatura máxima de armazenamento (GiB) 4	Largura de banda de linha de base / rajada (Gbps) [5]	Motores de apoio [6]
db.r4.2xlarge	8	61	-	120	2,5/10	3.6.0 somente
db.r4.4xlarge	16	122	-	240	5,0 /10	3.6.0 somente
db.r4.8xlarge	32	244	-	480	10	3.6.0 somente
db.r4.16xlarge	64	488	-	960	25	3.6.0 somente
T4G: classes de instância de desempenho expansível de última geração com base no Graviton2 T-series as instâncias são executadas no modo de intermitência de CPU ilimitada. O uso contínuo além da linha de base é cobrado a mais. Não há suporte para clusters globais.						
db.t4g.medium	2	4	-	8,13	0,256/5	4.0.0, 5.0.0 e 8.0.0
T3: classes de instância de desempenho expansível de geração anterior						
db.t3.medium	2	4	-	7,5	0,256/5	3.6.0, 4.0.0, 5.0.0 e 8.0.0

Classe de instância	vCPU[1]	Memória (GiB)[2]	Cache hierárquico SSD NVMe (GiB) [3]	Temperatura máxima de armazenamento (GiB) 4	Largura de banda de linha de base / rajada (Gbps) [5]	Motores de apoio [6]

1. vCPU: o número de unidades de processamento central (CPUs) virtuais. CPU virtual é uma unidade de capacidade que pode ser usada para comparar classes de instância. Em vez de comprar ou alugar um determinado processador para usar durante vários meses ou anos, você está alugando a capacidade de acordo com a hora. Nossa meta é fornecer uma quantidade consistente de capacidade da CPU independentemente do hardware subjacente real.
2. Memória (GiB): a memória RAM, em gigabytes, que é atribuída à instância. Geralmente, há uma proporção consistente entre a memória e a vCPU.
3. Cache hierárquico SSD NVMe: o espaço no volume de SSD, medido em gigabytes, alocado como cache estendido para armazenar dados efêmeros. Esse cache só está disponível em NVMe-backed instâncias.
4. Armazenamento em temperatura máxima (GiB): o espaço, medido em gigabytes, alocado à instância para armazenamento não persistente de arquivos temporários. Por NVMe-backed exemplo, esse armazenamento é hospedado em um volume NVMe-based SSD. Em todas as outras instâncias, ele está hospedado no Amazon Elastic Block Store (EBS).
5. Largura de banda de linha de base/rajada (Gbps): a largura de banda de rajada representa a largura de banda máxima em gigabits por segundo. Divida por oito para obter o throughput esperado em gigabytes por segundo. Instâncias de tamanho 4xlarge e menores têm uma largura de banda de linha de base. Para atender à demanda adicional, eles podem usar um mecanismo I/O de crédito de rede para ultrapassar sua largura de banda básica. As instâncias podem usar largura de banda expansível por um tempo limitado, geralmente de 5 a 60 minutos, dependendo do tamanho da instância.
6. Mecanismos de suporte — os mecanismos do Amazon DocumentDB que oferecem suporte à classe de instância.

NVMe-backed instâncias

NVMe-backed as instâncias oferecem desempenho de consulta até 7 vezes mais rápido para aplicativos com grandes conjuntos de dados que excedem a memória de uma instância comum.

Essas instâncias aproveitam o armazenamento em Solid-State drive (SSD) local não volátil baseado em memória expressa (NVMe) disponível em instâncias r6gd para armazenar dados efêmeros, reduzir o acesso ao armazenamento baseado em rede e melhorar a latência e a taxa de transferência de leitura.

O espaço em SSD local é dividido em duas seções:

- Cache hierárquico: aproximadamente 73% do SSD local é alocado como cache de banco de dados, permitindo que o sistema armazene até cinco vezes mais páginas do banco de dados do que somente a memória principal. O SSD local atua como um cache de segundo nível, enquanto o cache de buffer existente na memória continua sendo o cache de primeiro nível. A consulta só acessa o armazenamento externo se o cache do buffer e o cache de SSD falharem.
- Armazenamento temporário: os 27% restantes são reservados para armazenamento não persistente de arquivos temporários, usado para consultas complexas envolvendo classificações ou operações que consomem muitos recursos, como compilações de índices. Em instâncias regulares, o espaço temporário reside em um volume de armazenamento do Amazon Elastic Block Store (EBS). O armazenamento temporário hospedado localmente no SSD reduz a latência de consultas envolvendo classificações em até duas vezes e acelera as operações que consomem muitos recursos, como a criação de índices.

Os detalhes sobre o tipo de NVMe-backed instâncias suportadas e suas especificações podem ser encontrados emEspecificações de classe de instância.

Tópicos

- Casos de uso recomendados para NVMe-backed instâncias
- Usando NVMe-backed instâncias com o Amazon DocumentDB
- NVMe-backed Instâncias de monitoramento

Casos de uso recomendados para NVMe-backed instâncias

Use NVMe-backed instâncias nos seguintes cenários:

- Read-heavy cargas de trabalho — se sua carga de trabalho exige muita leitura e seu conjunto de dados é maior do que o cache de buffer, indicado por `ReadIOPS` métricas baixas `BufferCacheHitRatio` e altas, as instâncias podem oferecer benefícios de desempenho. NVMe-backed

- Update-heavy cargas de trabalho — se sua carga de trabalho exige muitas atualizações e a coleta de lixo não consegue acompanhar o ritmo devido à latência de leitura no armazenamento da rede, as NVMe-backed instâncias podem ajudar a mitigar o problema.

NVMe-backed as instâncias podem beneficiar vários casos de uso, incluindo:

- Internet-scale aplicativos — Aplicativos como processamento de pagamentos, cobrança e comércio eletrônico com contratos de nível de serviço (SLAs) de desempenho rigoroso podem aproveitar as vantagens de desempenho das NVMe-backed instâncias.
- Real-time painéis de relatórios — painéis que executam centenas de consultas para metrics/data coleta podem se beneficiar da baixa latência e do alto rendimento das instâncias. NVMe-backed
- Aplicativos generativos de IA — Os aplicativos que usam a pesquisa vetorial para encontrar vizinhos exatos ou mais próximos em milhões de incorporações vetoriais podem aproveitar o alto desempenho das NVMe-backed instâncias.

Usando NVMe-backed instâncias com o Amazon DocumentDB

Para usar NVMe-backed instâncias do Amazon DocumentDB:

- Crie um cluster Amazon DocumentDB e adicione uma das classes de NVMe-backed instância. Para obter mais informações, consulte Criar um cluster do Amazon DocumentDB.
- Como alternativa, modifique um cluster existente do Amazon DocumentDB para usar uma das classes de NVMe-backed instância. Para obter mais informações, consulte Modificar um cluster do Amazon DocumentDB.

Para verificar a disponibilidade de NVMe-backed instâncias em diferentes AWS regiões, consulteClasses de instância compatíveis por região.

Se você quiser voltar de uma NVMe-backed instância para uma instância regular, modifique a classe de instância de banco de dados da sua instância do Amazon DocumentDB para uma classe de instância similar sem o armazenamento NVMe. Por exemplo, se a sua classe de instância atual for db.r6gd.4xlarge, selecione ‘db.r6g.4xlarge’ para alternar de volta. Para obter mais informações, consulte Modificar um cluster do Amazon DocumentDB.

NVMe-backed Instâncias de monitoramento

Além das métricas de instância regulares disponíveis na Amazon CloudWatch, as NVMe-backed instâncias emitem métricas adicionais específicas para armazenamento NVMe-based SSD, IOPS e taxa de transferência.

```
NVMeStorageCacheHitRatio
FreeNVMeStorage
ReadIOPSNVMeStorage
ReadLatencyNVMeStorage
ReadThroughputNVMeStorage
WriteIOPSNVMeStorage
WriteLatencyNVMeStorage
WriteThroughputNVMeStorage
```

Para obter mais informações sobre essas métricas, consulte NVMe-backed métricas de instância

Gerenciamento de grupos de sub-rede do Amazon DocumentDB

Uma nuvem privada virtual (VPC) é uma rede virtual dedicada ao seu Conta da AWS. Ele é logicamente isolado de outras redes virtuais na AWS nuvem. Você pode iniciar seus AWS recursos, como clusters do Amazon DocumentDB, em sua Amazon VPC. Você pode especificar um intervalo de endereços IP para a VPC, adicionar sub-rede, associar security groups e configurar tabelas de rota.

Uma sub-rede é uma gama de endereços IP na sua Amazon VPC. Você pode iniciar AWS recursos em uma sub-rede especificada. Use uma sub-rede pública para recursos que devem ser conectados à internet. Use uma sub-rede privada para recursos que não serão conectados à internet. Para obter mais informações sobre sub-redes públicas e privadas, consulte Noções básicas de VPC e sub-rede no Guia do usuároio da Amazon Virtual Private Cloud.

Um grupo de sub-redes de banco de dados é uma coleção de sub-redes que você cria em uma VPC e designa para seus clusters. Um grupo de sub-redes permite que você especifique uma VPC específica ao criar clusters. Se você usar o grupo de sub-redes `default`, ele abrangerá todas as sub-redes na VPC.

Cada grupo de sub-redes de banco de dados deve ter sub-redes em pelo menos duas zonas de disponibilidade em uma determinada região. Ao criar um cluster de banco de dados na VPC, você deve selecionar um grupo de sub-rede de banco de dados. O Amazon DocumentDB usa esse grupo

de sub-rede de banco de dados e sua zona de disponibilidade preferida para selecionar uma sub-rede e um endereço IP dentro dessa sub-rede para associar ao seu cluster. Se a instância principal falhar, o Amazon DocumentDB poderá promover uma instância de réplica correspondente para ser a nova instância principal. Ela pode criar uma nova instância de réplica usando um endereço IP da sub-rede em que a instância principal anterior foi localizada.

Quando o Amazon DocumentDB cria uma instância em uma VPC, ele atribui uma interface de rede ao seu cluster usando um endereço IP selecionado do seu grupo de sub-redes de banco de dados. Sempre use o nome DNS porque o endereço IP subjacente pode mudar durante o failover. Para obter mais informações, consulte Endpoints do Amazon DocumentDB.

Para obter informações sobre como criar sua própria VPC e sub-redes, consulte Como trabalhar com VPCs e sub-redes no Guia do usuário do Amazon Virtual Private Cloud.

Tópicos

- Criação de um grupo de sub-redes do Amazon DocumentDB
- Descrevendo um grupo de sub-redes do Amazon DocumentDB
- Modificar um grupo de sub-redes do Amazon DocumentDB
- Excluir um grupo de sub-redes do Amazon DocumentDB

Criação de um grupo de sub-redes do Amazon DocumentDB

Ao criar um cluster do Amazon DocumentDB, você deve escolher uma Amazon VPC e um grupo de sub-rede correspondente dentro dessa Amazon VPC para iniciar o seu cluster. As sub-redes determinam a zona de disponibilidade e intervalo de IP dentro da zona de disponibilidade que você deseja usar para iniciar uma instância.

Um grupo de sub-redes permite que você especifique as sub-redes (ou AZs) que permite a você especificar as zonas de disponibilidade que deseja usar para iniciar instâncias do Amazon DocumentDB. Por exemplo, em um cluster com três instâncias, é recomendável que cada uma dessas instâncias seja provisionada em AZs separadas - fazendo isto, otimiza para alta disponibilidade. Assim, se uma única AZ falhar, ela afetará apenas uma única instância.

As instâncias do Amazon DocumentDB podem ser provisionadas em até três zonas de disponibilidade. Mesmo que um grupo tenha mais de três sub-redes, você só pode usar três dessas sub-redes para criar um cluster do Amazon DocumentDB. Ao criar um grupo de sub-redes, escolha somente as três sub-redes nas quais você deseja implantar suas instâncias.

Por exemplo: um cluster é criado e o Amazon DocumentDB escolhe AZs {1A, 1B e 1C}. Se você tentar criar uma instância na AZ {1D} a chamada de API não será bem-sucedida. No entanto, se você optar por criar uma instância, sem especificar a AZ específica, o Amazon DocumentDB escolherá uma AZ por você. O Amazon DocumentDB usa um algoritmo para balancear a carga das instâncias em todas as AZs para ajudar você a alcançar alta disponibilidade. Por exemplo, se três instâncias são provisionadas, por padrão, elas serão provisionadas em três zonas de disponibilidade e não serão provisionadas todas em uma única AZ.

Práticas recomendadas

- A menos que você tenha um motivo específico, sempre crie um grupo de sub-rede com três sub-redes. Isso garantirá que clusters com três ou mais instâncias alcancem maior disponibilidade à medida que as instâncias sejam provisionadas em três zonas de disponibilidade.
- Sempre distribua instâncias em vários AZs para obter alta disponibilidade. Nunca coloque todas as instâncias de um cluster em uma única AZ.
- Como os eventos de failover podem acontecer a qualquer momento, você não deve presumir que uma instância primária ou de réplica sempre estará em uma determinada AZ.

Como criar um grupo de sub-redes

Você pode usar o Console de gerenciamento da AWS ou AWS CLI para criar um grupo de sub-redes do Amazon DocumentDB:

Using the Console de gerenciamento da AWS

Use as seguintes etapas para criar um grupo de sub-redes do Amazon DocumentDB.

Para criar um grupo de sub-redes do Amazon DocumentDB

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, selecione Subnet groups (Grupos de sub-redes) e, depois, escolha Create (Criar).

> **Tip**
>
> Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

3. Na página Criar grupo de sub-redes:

   a. Na seção Detalhes do grupo de sub-redes:

      i. Nome—Digite um nome significativo para o grupo de sub-redes.

      ii. Descrição—insira uma descrição para o grupo de sub-redes.

   b. Na seção Adicionar sub-redes:

      i. VPC—Na lista, escolha uma VPC para esse grupo de sub-redes.

      ii. Execute um destes procedimentos:

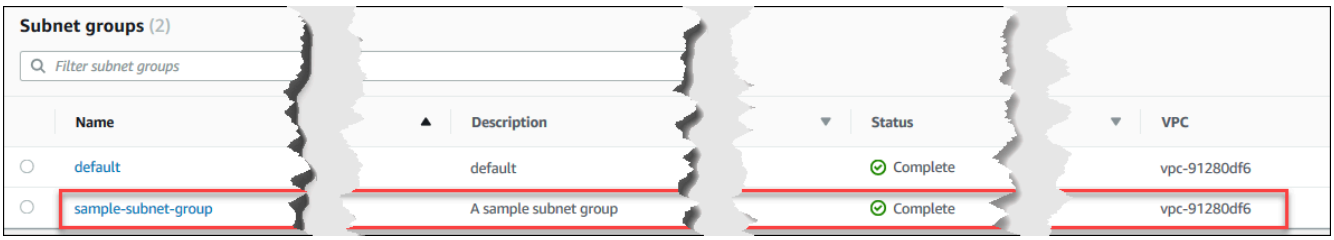
      - Para incluir todas as sub-redes na VPC escolhida, selecione Add all the subnets related to this VPC (Adicionar todas as sub-redes relacionadas a essa VPC).
      - Para especificar sub-redes para esse grupo de sub-redes, faça o seguinte para cada zona de disponibilidade na qual você deseja incluir sub-redes. Você deve incluir pelo menos duas zonas de disponibilidade.

        A. Zona de disponibilidade—Na lista, escolha uma zona de disponibilidade.

        B. Sub-rede—Na lista, escolha uma sub-rede na zona de disponibilidade escolhida para esse grupo de sub-redes.

        C. Escolha Adicioinar sub-rede.

4. Escolha Criar. Quando o grupo de sub-rede é criado, ele é listado com seus outros grupos de sub-redes.



Using the AWS CLI

Antes de criar um grupo de sub-redes usando o AWS CLI, você deve primeiro determinar quais sub-redes estão disponíveis. Execute a AWS CLI operação a seguir para listar as zonas de disponibilidade e suas sub-redes.

Parâmetros:

- **`--db-subnet-group`**: opcional. A especificação de um determinado grupo de sub-redes lista as zonas de disponibilidade e as sub-redes para esse grupo. Omitir esse parâmetro lista zonas de disponibilidade e as sub-redes para todos os seus grupos de sub-redes. A especificação do grupo de sub-redes `default` lista todas as sub-redes da VPC.

Example

Para Linux, macOS ou 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]]'
```

A saída dessa operação é semelhante ao seguinte (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"
            ]
        ]
    ]
]
```

Usando a saída da operação anterior, você pode criar um novo grupo de sub-redes. O novo grupo de sub-redes deve incluir sub-redes em pelo menos duas zonas de disponibilidade.

Parâmetros:

- **`--db-subnet-group-name`**—Obrigatório. O nome para esse grupo de sub-redes.
- **`--db-subnet-group-description`**—Obrigatório. A descrição do grupo de sub-redes.
- **`--subnet-ids`**—Obrigatório. Uma lista de sub-redes a serem incluídas no grupo de sub-redes. Exemplo: `subnet-53ab3636`.
- Tags: opcionais. Uma lista de tags (pares de chave/valor) a serem anexadas a esse grupo de sub-redes.

O código a seguir cria o grupo de sub-redes `sample-subnet-group` com três sub-redes, `subnet-4e26d263`, `subnet-afc329f4` e `subnet-b3806e8f`.

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante ao seguinte (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"
    }
}
```

Descrevendo um grupo de sub-redes do Amazon DocumentDB

Você pode usar o Console de gerenciamento da AWS ou o AWS CLI para obter os detalhes de um grupo de sub-redes do Amazon DocumentDB.

Using the Console de gerenciamento da AWS

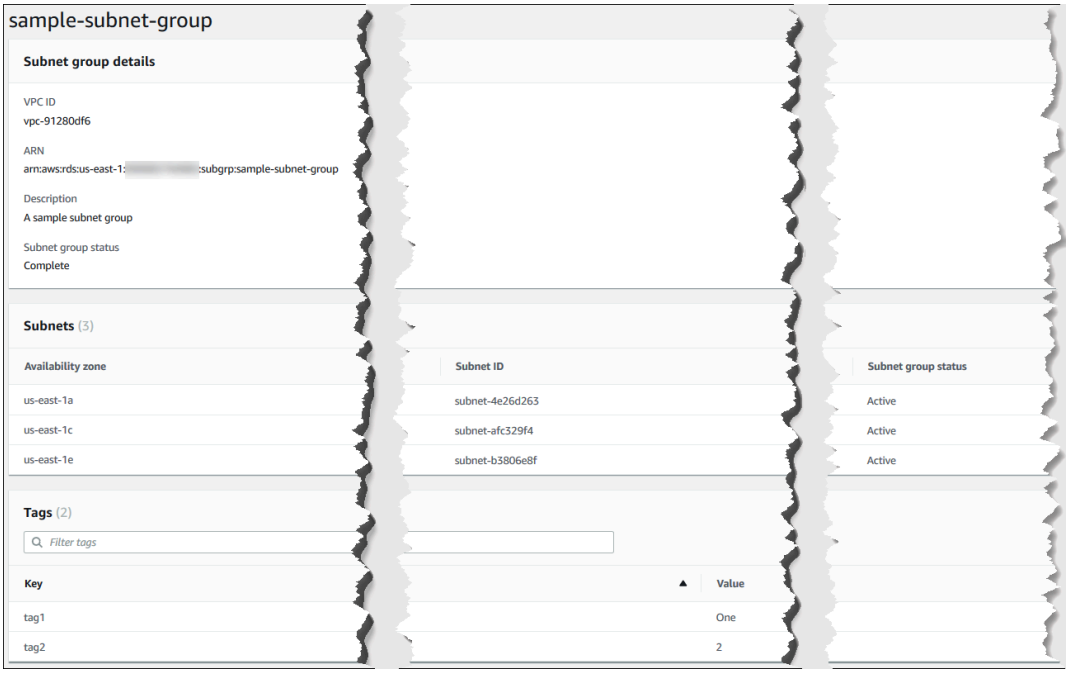
O procedimento a seguir mostra como obter os detalhes de um grupo de sub-redes do Amazon DocumentDB.

Para localizar os detalhes de um grupo de sub-redes

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Grupos de sub-redes.

   > **Tip**
   >
   > Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

3. Para ver os detalhes de um grupo de sub-redes, escolha o nome do grupo de sub-redes.



Using the AWS CLI

Para localizar os detalhes de um grupo de sub-redes do Amazon DocumentDB, use a operação `describe-db-subnet-groups` com o seguinte parâmetro.

Parâmetro

- `--db-subnet=group-name`: opcional. Se incluído, os detalhes do grupo de sub-redes chamado serão listados. Se omitido, os detalhes para até 100 grupos de sub-redes serão listados.

Example

O código a seguir lista os detalhes para o grupo de sub-redes `sample-subnet-group` que criamos na seção Criação de um grupo de sub-redes do Amazon DocumentDB.

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante ao seguinte (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"
    }
}
```

Modificar um grupo de sub-redes do Amazon DocumentDB

Você pode usar o Console de gerenciamento da AWS ou AWS CLI para modificar a descrição de um grupo de sub-redes ou para adicionar ou remover sub-redes de um grupo de sub-redes do Amazon DocumentDB. No entanto, você não pode modificar o grupo de sub-redes `default`.

Using the Console de gerenciamento da AWS

Você pode usar o Console de gerenciamento da AWS para alterar a descrição de um grupo de sub-redes ou para adicionar e remover sub-redes. Lembre-se de que, ao terminar, você deve ter pelo menos duas zonas de disponibilidade associadas ao seu grupo de sub-redes.

Para modificar o grupo de sub-redes

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Grupos de sub-redes. Em seguida, escolha o botão à esquerda do nome do grupo de sub-redes. Lembre-se de que você não pode modificar o grupo de sub-redes `default`.

   > **Tip**
   >
   > Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

3. Escolha Ações e, em seguida, Modificar.
4. Descrição-Para alterar a descrição de seu grupo de sub-redes, insira uma nova descrição.
5. Para alterar as sub-redes associadas ao grupo de sub-redes, na seção Adicionar sub-redes, siga um ou mais destes procedimentos:
   - Para remover todas as sub-redes desse grupo, escolha Remover todos.
   - Para remover sub-redes específicas desse grupo, escolha Remover para cada sub-rede que você deseja remover.
   - Para adicionar todas as sub-redes associadas a essa VPC, escolha Adicionar todas as sub-redes relacionadas a essa VPC.
   - Para adicionar sub-redes específicas a esse grupo de sub-redes, faça o seguinte para cada zona de disponibilidade à qual você deseja adicionar uma sub-rede.

a. Zona de disponibilidade-Na lista, escolha uma nova zona de disponibilidade.

b. Sub-rede—Na lista, escolha uma sub-rede na zona de disponibilidade escolhida para esse grupo de sub-redes.

c. Escolha Adicioinar sub-rede.

6. Na caixa de diálogo de confirmação:
   - Para fazer essas alterações no grupo de sub-redes, escolha Modificar.
   - Para manter o grupo de sub-redes inalterado, escolha Cancelar.

Using the AWS CLI

Você pode usar o AWS CLI para alterar a descrição de um grupo de sub-redes ou para adicionar e remover sub-redes. Lembre-se de que, ao terminar, você deve ter pelo menos duas zonas de disponibilidade associadas ao seu grupo de sub-redes. Você não pode modificar o grupo de sub-redes `default`.

Parâmetros:

- `--db-subnet-group-name`—Obrigatório. O nome do grupo de sub-redes do Amazon DocumentDB que você está modificando.
- `--subnet-ids`—Obrigatório. Uma lista de todas as sub-redes que você deseja no grupo de sub-redes após essa alteração é feita.

> **Important**
>
> Todas as sub-redes que estiverem no grupo, mas não estiverem nessa lista serão removidas do grupo de sub-redes. Se você desejar manter qualquer uma das sub-redes que estiverem atualmente no grupo, inclua-as nessa lista.

- `--db-subnet-group-description`: opcional. A descrição do grupo de sub-redes.

Example

O código a seguir modifica a descrição e substitui as sub-redes existentes pelas sub-redes `subnet-991cb8d0`, `subnet-53ab3636` e `subnet-29ab1025`.

Para Linux, macOS ou 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"
```

A saída dessa operação é semelhante ao seguinte (formato JSON). Observe que esse é o mesmo grupo de sub-rede que foi criado na seção Criação de um grupo de sub-redes do Amazon DocumentDB. No entanto, as sub-redes no grupo de sub-rede são substituídas pelas listadas na operação `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"
    }
}
```

Excluir um grupo de sub-redes do Amazon DocumentDB

Você pode usar o Console de gerenciamento da AWS ou AWS CLI para excluir um grupo de sub-redes do Amazon DocumentDB. No entanto, você não pode excluir o grupo de sub-redes `default`.

Using the Console de gerenciamento da AWS

Você pode usar o Console de gerenciamento da AWS para excluir um grupo de sub-redes. No entanto, você não pode excluir o grupo de sub-redes `default`.

Para excluir um grupo de sub-redes

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Grupos de sub-redes. Em seguida, escolha o botão à esquerda do nome do grupo de sub-redes. Lembre-se de que você não pode excluir o grupo de sub-redes `default`.

   > **Tip**
   >
   > Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

3. Escolha Ações e, em seguida, escolha Excluir.
4. Na caixa de diálogo de confirmação:
   - Para excluir o grupo de sub-redes, escolha Excluir.
   - Para manter o grupo de sub-redes, escolha Cancelar.

Using the AWS CLI

Para excluir um grupo de sub-redes do Amazon DocumentDB usando o AWS CLI, use a `delete-db-subnet-group` operação com o seguinte parâmetro.

Parâmetro

- `--db-subnet-group-name`—Obrigatório. O nome do grupo de sub-redes do Amazon DocumentDB a ser excluído. Lembre-se de que você não pode excluir o grupo de sub-redes `default`.

Example

O código a seguir exclui `sample-subnet-group`.

Para Linux, macOS ou 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
```

A operação não produzirá uma saída.

Alta disponibilidade e replicação do Amazon DocumentDB

Você pode obter alta disponibilidade e escalabilidade de leitura no Amazon DocumentDB (compatível com MongoDB) usando instâncias de réplica. Um único cluster do Amazon DocumentDB oferece suporte a uma única instância primária e até 15 instâncias de réplica. Essas instâncias podem ser distribuídas em zonas de disponibilidade dentro da região do cluster. A instância principal aceita o tráfego de leitura e de gravação e as instâncias de réplica aceitam somente solicitações de leitura.

O volume do cluster é composto por várias cópias dos dados do cluster. Contudo, os dados no volume do cluster são representados como um volume lógico e único para a instância primária e para réplicas do Amazon DocumentDB no cluster. No final, as instâncias de réplica tornam-se

consistentes. Elas retornam os resultados da consulta com atraso de réplica mínimo, geralmente muito inferior a 100 milissegundos após a instância principal ter gravado uma atualização. O atraso da réplica varia de acordo com a taxa de mudança no banco de dados. Ou seja, durante os períodos em que um grande número de operações de gravação ocorre para o banco de dados, você pode ver um aumento no atraso da réplica.

Escalabilidade de leitura

As réplicas do Amazon DocumentDB funcionam bem para a escalabilidade de leitura porque são totalmente dedicadas a operações de leitura no seu volume de cluster. As operações de gravação são gerenciadas pela instância principal. O volume do cluster é compartilhado com todas as instâncias em seu cluster. Portanto, você não precisa replicar e manter uma cópia dos dados para cada réplica do Amazon DocumentDB.

Alta disponibilidade

Quando você cria um cluster do Amazon DocumentDB, dependendo do número de zonas de disponibilidade no grupo de sub-redes (deve haver pelo menos duas) o Amazon DocumentDB provisiona instâncias entre as zonas de disponibilidade. Quando você cria instâncias no cluster, o Amazon DocumentDB distribui automaticamente as instâncias entre as zonas de disponibilidade em um grupo de sub-redes para balancear o cluster. Essa ação também impede que todas as instâncias estejam localizadas na mesma zona de disponibilidade.

Exemplo

Para ilustrar, considere um exemplo no qual você criou um cluster que tem um grupo de sub-redes com três zonas de disponibilidade: AZ1, AZ2 e AZ3.

Quando a primeira instância no cluster é criada, é a instância principal e é localizada em uma das zonas de disponibilidade. Neste exemplo, está AZ1. A segunda instância criada é uma instância de réplica e é localizada em uma das outras duas zonas de disponibilidade, digamos, AZ2. A terceira instância criada é uma instância de réplica e está localizada na zona de disponibilidade remanescente, ou seja, a AZ3. Se você criar mais instâncias, elas serão distribuídas entre as zonas de disponibilidade para alcançar o equilíbrio no cluster.

Se ocorrer uma falha na instância principal (AZ1), um failover será acionado, e uma das réplicas existentes será promovida a principal. Quando a instância principal antiga for recuperada, ela se tornará uma réplica na mesma zona de disponibilidade na qual ela foi provisionada (AZ1). Quando

você provisiona um cluster de três instâncias, o Amazon DocumentDB continua preservando esse cluster de três instâncias. O Amazon DocumentDB gerencia automaticamente a detecção, o failover e a recuperação de falhas de instância sem qualquer intervenção manual.

Quando o Amazon DocumentDB executar um failover e recuperar uma instância, a instância recuperada permanecerá na zona de disponibilidade na qual ela foi originalmente provisionada. No entanto, a função da instância pode mudar de principal para réplica. Isso impede o cenário em que uma série de failovers pode resultar em todas as instâncias estando na mesma zona de disponibilidade.

Você pode especificar réplicas do Amazon DocumentDB como destinos de failover. Ou seja, se ocorrer uma falha na instância principal, a réplica especificada do Amazon DocumentDB ou a réplica de uma camada será promovida a instância principal. Há uma breve interrupção durante a qual as solicitações de leitura e gravação feitas na instância principal falharão com uma exceção. Se o cluster do Amazon DocumentDB não incluir réplicas do Amazon DocumentDB, quando a instância principal falhar, ela será recriada. Promover uma réplica do Amazon DocumentDB é muito mais rápido do que recriar a instância principal.

Para alta disponibilidade, crie uma ou mais réplicas do Amazon DocumentDB da mesma classe de instância da instância primária em diferentes zonas de disponibilidade.

Para saber mais, consulte:

- Entender a tolerância a falhas de cluster do Amazon DocumentDB
- Failover do Amazon DocumentDB
  - Controlar o destino de failover

Nas versões 5.0 e 8.0 do Amazon DocumentDB, as instâncias de réplica não são reiniciadas quando a instância primária é reiniciada. Eles continuam atendendo às solicitações de leitura durante a reinicialização da instância primária, mantendo a disponibilidade aprimorada.

Alta disponibilidade com clusters globais

Para alta disponibilidade em vários Regiões da AWS, você pode configurar clusters globais do Amazon DocumentDB. Cada cluster global abrange várias regiões, permitindo leituras globais de baixa latência e recuperação de desastres em caso de interrupções em todas as regiões. O Amazon DocumentDB gerencia automaticamente a replicação de todos os dados e atualizações da região primária para cada uma das regiões secundárias.

Adicionar réplicas do

A primeira instância adicionada ao cluster é a instância principal. Todas instância adicionada após a primeira instância é uma instância de réplica. Um cluster pode ter até 15 instâncias de réplica, além da instância principal.

Quando você cria um cluster usando o Console de gerenciamento da AWS, uma instância primária é criada automaticamente ao mesmo tempo. Para criar uma réplica ao mesmo tempo em que cria o cluster e a instância principal, escolha Criar réplica em zona diferente. Para obter mais informações, consulte a etapa 4.d em Criar um cluster do Amazon DocumentDB. Para adicionar mais réplicas a um cluster do Amazon DocumentDB, consulte Adicionar uma instância do Amazon DocumentDB a um cluster.

Ao usar o AWS CLI para criar seu cluster, você deve criar explicitamente suas instâncias primárias e de réplica. Para obter mais informações, consulte a seção “Usando o AWS CLI” nos tópicos a seguir:

- Criar um cluster do Amazon DocumentDB
- Adicionar uma instância do Amazon DocumentDB a um cluster

Failover do Amazon DocumentDB

Em alguns casos, como determinados tipos de manutenção programada, ou no caso improvável de uma falha de nó primário ou zona de disponibilidade, o Amazon DocumentDB (compativel com MongoDB) detecta a falha e substitui o nó primário. Durante um failover, o tempo de gravação é minimizado. Isso ocorre porque a função do nó primário fará failover em uma das réplicas de leitura, em vez de ter que criar e provisionar um novo nó primário. A detecção de falhas e a promoção de réplica garantem que você possa continuar a gravar no novo primário assim que a promoção estiver concluída.

Para que o failover funcione, o cluster deve ter pelo menos duas instâncias — uma instância principal e pelo menos uma réplica.

> **Note**
>
> Esse tópico se aplica somente aos clusters baseados em instâncias originais do Amazon DocumentDB. Ele não se aplica aos clusters elásticos ou globais.

Controlar o destino de failover

O Amazon DocumentDB fornece níveis de failover para controlar qual instância de réplica é promovida a instância principal quando ocorre um failover.

Níveis de failover

Cada instância de réplica é associada a um nível de failover (0–15). Quando ocorre um failover devido à manutenção ou a uma falha de hardware improvável, a instância principal executa failover em uma réplica com o maior nível de prioridade (o nível numerado mais baixo). Se várias réplicas tiverem o mesmo nível de prioridade, a principal executará failover na réplica do nível que é mais próximo em tamanho da prévia principal.

Ao definir o nível de failover de um grupo de réplicas selecionadas como `0` (a prioridade mais alta), você pode garantir que um failover promova uma das réplicas desse grupo. Você pode impedir efetivamente que as réplicas específicas sejam promovidas a principal no caso de um failover, atribuindo um nível de baixa prioridade (número alto) a essas réplicas. Isso é útil em casos em que réplicas específicas são muito usadas por uma aplicação, e o failover em um deles teria um impacto negativo em uma aplicação crítica.

Recomendamos que as réplicas que são possíveis alvos de failover usem a mesma classe de instância da instância primária. Isso evita a degradação do desempenho de gravação se um failover promover uma réplica durante atualizações do sistema operacional ou outras manutenções.

Você pode definir o nível de failover de uma instância ao criá-la ou posteriormente, modificando-o. Definir um nível de failover da instância modificando a instância não aciona um failover. Para obter mais informações, consulte os tópicos a seguir:

- Adicionar uma instância do Amazon DocumentDB a um cluster
- Modificar uma instância do Amazon DocumentDB

Ao iniciar manualmente um failover, você tem dois meios de controlar qual instância de réplica é promovida a principal: o nível de failover, conforme descrito anteriormente, e o parâmetro `--target-db-instance-identifier`.

-- identificador de instância de banco de dados de **`destino`**

Para testar, você pode forçar um evento de failover usando a operação `failover-db-cluster`. Você pode usar o parâmetro `--target-db-instance-identifier` para especificar qual réplica

deve ser promovida à principal. O uso do parâmetro `--target-db-instance-identifier` substitui o nível de prioridade de failover. Se você não especificar o parâmetro `--target-db-instance-identifier`, o failover primário estará de acordo com o nível de prioridade de failover.

O que acontece durante um failover

O failover é automaticamente controlado pelo Amazon DocumentDB para que suas aplicações possam retomar operações de banco de dados o mais rápido possível e sem intervenção administrativa.

- Se você tiver uma instância da réplica do Amazon DocumentDB na mesma zona de disponibilidade ou em outra, ao fazer o failover: o Amazon DocumentDB alterará o registro de nome canônico (CNAME) da instância para apontar para a réplica íntegra que, por sua vez, é promovida e se torna a nova principal. Normalmente, o failover é concluído em 30 segundos do início ao fim.
- Se você não tiver uma instância de réplica do Amazon DocumentDB (por exemplo, um cluster de instância única): o Amazon DocumentDB tentará criar uma instância na mesma zona de disponibilidade que a instância original. O melhor possível é feito para realizar essa substituição da instância original, mas pode ser que isso não tenha êxito se, por exemplo, ocorrer um problema que afete amplamente a zona de disponibilidade.

Sua aplicação deve tentar novamente fazer as conexões do banco de dados em caso de uma perda de conexão.

Testar o failover

Um failover para um cluster promove uma das réplicas do Amazon DocumentDB (instâncias somente leitura) no cluster para ser a instância primária (o gravador do cluster).

Quando a instância principal falhar, o Amazon DocumentDB executará failover automaticamente em uma réplica do Amazon DocumentDB, se houver. É possível forçar um failover quando quiser simular uma falha de uma instância principal para testes. Cada instância em um cluster tem seu próprio endereço de endpoint. Portanto, você precisa limpar e restabelecer as conexões existentes que usam os endereços de endpoint quando o failover é concluído.

Para forçar um failover, use a operação `failover-db-cluster` com esses parâmetros.

- `--db-cluster-identifier`—Obrigatório. O nome do cluster no qual executar failover.

- `--target-db-instance-identifier`: opcional. O nome da instância a ser promovida à instância principal.

Example

A operação a seguir força um failover do cluster `sample-cluster`. Ela não especifica qual instância se tornará a nova instância principal, portanto o Amazon DocumentDB escolhe a instância de acordo com a prioridade do nível de failover.

Para Linux, macOS ou Unix:

```
aws docdb failover-db-cluster \
   --db-cluster-identifier sample-cluster
```

Para Windows:

```
aws docdb failover-db-cluster ^
   --db-cluster-identifier sample-cluster
```

A operação a seguir força um failover do cluster `sample-cluster`, especificando a `sample-cluster-instance` que será promovida à função principal. (Observe `"IsClusterWriter": true` na saída.)

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante ao seguinte (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"
    }
}
```

Lag de replicação

O atraso de replicação geralmente é de 50 ms ou menos. Os motivos mais comuns para aumentar o atraso da réplica são:

- Uma alta taxa de gravação no primário que faz com que as réplicas de leitura fiquem atrás do primário.
- Contenção nas réplicas de leitura entre consultas de longa execução (por exemplo, grandes varreduras sequenciais, consultas de agregação) e replicação de gravação recebida.
- Número muito grande de consultas simultâneas nas réplicas de leitura.

Para minimizar o atraso na replicação, experimente estas técnicas de solução de problemas:

- Se você tiver uma alta taxa de gravação ou alta utilização da CPU, amplie as instâncias em seu cluster.
- Se houver consultas de longa duração em suas réplicas de leitura e atualizações muito frequentes nos documentos que estão sendo consultados, considere alterar suas consultas de longa duração ou executá-las na primary/write réplica para evitar contenção nas réplicas de leitura.
- Se houver um número muito grande de consultas simultâneas ou alta utilização da CPU somente nas réplicas de leitura, outra opção é aumentar a escala horizontalmente do número de réplicas de leitura para espalhar a workload.
- Como o atraso na replicação resulta de uma alta taxa de transferência de gravação e consultas de longa execução, solucione o atraso de replicação usando a `DBClusterReplicaLagMaximum`

CloudWatch métrica em combinação com o registrador de consultas lento e as métricas//. `WriteThroughput` `WriteIOPS`

Garanta que todas as suas réplicas sejam do mesmo tipo de instância para que um failover de cluster não cause uma degradação no desempenho.

Se você estiver escolhendo entre aumentar a escala verticalmente e aumentar a escala horizontalmente (por exemplo, seis instâncias menores versus três instâncias maiores), geralmente recomendamos tentar aumentar a escala verticalmente primeiro (instâncias maiores) antes de aumentar a escala horizontalmente, pois você obterá um cache de buffer maior por instância de banco de dados.

Proativamente, você deve definir um alarme de atraso de replicação e definir seu limite para um valor que você acha que é o limite superior para o quão longe (ou “obsoleto”) seus dados em instâncias de réplica podem estar antes de começar a afetar a funcionalidade da aplicação. Em geral, aconselhamos que o limite de atraso de replicação seja excedido para vários pontos de dados antes do alarme, devido a workloads transitórias.

> **Note**
>
> Defina outro alarme para atrasos de replicação que excedam 10 segundos. Se você ultrapassar esse limite para vários pontos de dados, amplie suas instâncias ou reduza sua taxa de transferência de gravação na instância primária.

Gerenciando eventos do Amazon DocumentDB

O Amazon DocumentDB (compatível com MongoDB) mantém um registro de eventos que se relacionam a seus clusters, instâncias, snapshots, grupos de segurança e grupos de parâmetros do cluster. Essas informações incluem a data e a hora do evento, o nome e o tipo de origem do evento, além de uma mensagem que está associada ao evento.

> **Important**
>
> Para determinados recursos de gerenciamento, o Amazon DocumentDB usa a tecnologia operacional que é compartilhada com o Amazon RDS e o Amazon Neptune. Os limites de região, limites que são governados no nível da Região, são compartilhados entre o Amazon

DocumentDB, o Amazon RDS e o Amazon Neptune. Para obter mais informações, consulte Cotas regionais.

Tópicos

- Visualização das categorias de eventos do Amazon DocumentDB
- Visualizando eventos do Amazon DocumentDB

Visualização das categorias de eventos do Amazon DocumentDB

Cada tipo de recurso do Amazon DocumentDB tem tipos específicos de eventos que podem ser associados a ele. Você pode usar a AWS CLI `describe-event-categories` operação para visualizar o mapeamento entre os tipos de eventos e os tipos de recursos do Amazon DocumentDB.

Parâmetros

- **`--source-type`**—Opcional. Use o parâmetro `--source-type` para ver as categorias de evento para um determinado tipo de origem. Os seguintes valores são permitidos:
  - `db-cluster`
  - `db-instance`
  - `db-parameter-group`
  - `db-security-group`
  - `db-cluster-snapshot`
- **`--filters`**—Opcional. Para visualizar as categorias de eventos apenas do Amazon DocumentDB, use o filtro `--filter Name=engine,Values=docdb`.

Example

O código a seguir lista as categorias de eventos associadas aos clusters.

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante ao seguinte (formato JSON).

```
{
    "EventCategoriesMapList": [
        {
            "EventCategories": [
                "notification",
                "failure",
                "maintenance",
                "failover"
            ],
            "SourceType": "db-cluster"
        }
    ]
}
```

O código a seguir lista as categorias de eventos associadas a cada tipo de origem do Amazon DocumentDB.

```
aws docdb describe-event-categories
```

A saída dessa operação é semelhante ao seguinte (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"
            ]
        }
    ]
}
```

Visualizando eventos do Amazon DocumentDB

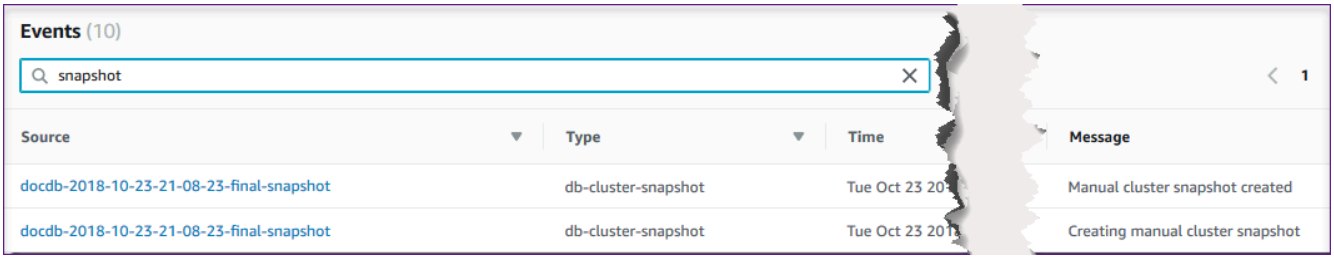
Você pode recuperar eventos para seus recursos do Amazon DocumentDB pelo console do Amazon DocumentDB, que mostra eventos das últimas 24 horas. Você também pode recuperar eventos para

seus recursos do Amazon DocumentDB usando o comando AWS CLI describe-events ou a operação da API Amazon DescribeEventsDocumentDB. Se você usar a API do Amazon DocumentDB AWS CLI ou a API do Amazon DocumentDB para visualizar eventos, poderá recuperar eventos dos últimos 14 dias.

Using the Console de gerenciamento da AWS

Para visualizar todos os eventos de instâncias do Amazon DocumentDB nas últimas 24 horas

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em. https://console.aws.amazon.com/docdb
2. No painel de navegação, escolha Eventos. Os eventos disponíveis aparecem em uma lista.
3. Use a lista Filtro para filtrar os eventos por tipo. Insira um termo na caixa de texto para filtrar os resultados. Por exemplo, a captura de tela a seguir mostra a filtragem de todos os eventos do Amazon DocumentDB para eventos snapshot.



Using the AWS CLI

Para visualizar todos os eventos de instância do Amazon DocumentDB dos últimos 7 dias

Você pode visualizar todos os eventos de instância do Amazon DocumentDB nos últimos 7 dias executando a AWS CLI operação describe-events com o `--duration` parâmetro definido como (10.080 minutos). `10080`

```
aws docdb describe-events --duration 10080
```

Filtragem para eventos do Amazon DocumentDB

Para ver os eventos específicos do Amazon DocumentDB, use a operação `describe-events` com os seguintes parâmetros.

Parâmetros

- **`--filter`**—Necessário para limitar os valores retornados ao evento Amazon DocumentDB. Use **`Name=engine,Values=docdb`** para filtrar todos os eventos somente para o Amazon DocumentDB.
- **`--source-identifier`**—Opcional. O identificador da origem do evento para o qual os eventos são retornados. Se omitido, os eventos de todas as origens são incluídos nos resultados.
- **`--source-type`**—Opcional, a menos que `--source-identifier` seja fornecido, então é necessário. Se `--source-identifier` for fornecido, `--source-type` deve estar de acordo com o tipo do `--source-identifier`. Os seguintes valores são permitidos:
  - `db-cluster`
  - `db-instance`
  - `db-parameter-group`
  - `db-security-group`
  - `db-cluster-snapshot`

O exemplo a seguir lista todos os eventos do Amazon DocumentDB.

```
aws docdb describe-events --filters Name=engine,Values=docdb
```

A saída dessa operação é semelhante ao seguinte (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 obter mais informações, consulte Auditar eventos do Amazon DocumentDB.

Gerenciando grupos de parâmetros de cluster do Amazon DocumentDB

É possível gerir a configuração do mecanismo do Amazon DocumentDB usando parâmetros em um grupo de parâmetros de cluster. Um grupo de parâmetros de cluster é um conjunto de valores de configuração do Amazon DocumentDB que facilita o gerenciamento dos parâmetros de seus clusters do Amazon DocumentDB. Grupos de parâmetro de cluster atuam como contêiner de valores de configuração de mecanismo que são aplicados a todas as instâncias no cluster.

Esta seção descreve como criar, exibir e modificar grupos de parâmetros de cluster. Ele também mostra como é possível determinar qual grupo de parâmetros de cluster está associado a um determinado cluster.

Tópicos

- Descrever os grupos de parâmetros de cluster do Amazon DocumentDB
- Criando grupos de parâmetros de cluster do Amazon DocumentDB
- Modificando grupos de parâmetros de cluster do Amazon DocumentDB
- Modificando clusters do Amazon DocumentDB para usar grupos de parâmetros de cluster personalizados
- Copiando grupos de parâmetros de cluster do Amazon DocumentDB
- Redefinindo grupos de parâmetros de cluster do Amazon DocumentDB
- Excluindo grupos de parâmetros de cluster do Amazon DocumentDB
- Referência de parâmetros de cluster do Amazon DocumentDB

Descrever os grupos de parâmetros de cluster do Amazon DocumentDB

Um grupo de parâmetros de `default` cluster é criado automaticamente quando você cria o primeiro cluster Amazon DocumentDB em um novo mecanismo Região da AWS ou está usando um novo mecanismo. Os clusters subsequentes, criados na mesma região e com a mesma versão de mecanismo, são criados com o grupo de parâmetros do `default` cluster.

Tópicos

- Descrevendo os detalhes de um grupo de parâmetros de cluster do Amazon DocumentDB
- Determinando o grupo de parâmetros de um cluster do Amazon DocumentDB
- Determinação de clusters e instâncias associados a um grupo de parâmetros de cluster do Amazon DocumentDB

Descrevendo os detalhes de um grupo de parâmetros de cluster do Amazon DocumentDB

Para descrever os detalhes de um determinado grupo de parâmetros do cluster, conclua as etapas a seguir usando o Console de gerenciamento da AWS ou o AWS Command Line Interface (AWS CLI).

Using the Console de gerenciamento da AWS

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Grupos de parâmetros.

> **Tip**
>
> Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

3. No painel Grupos de parâmetros de cluster, escolha o nome do grupo de parâmetros de cluster cujos detalhes deseja visualizar.
4. A página resultante exibe os parâmetros do grupo de parâmetros, a atividade recente e as tags.

   - Em Parâmetros de cluster, é possível ver o nome do parâmetro, o valor atual, os valores permitidos, se o parâmetro é modificável, o tipo de aplicação, o tipo de dado e a descrição. É possível modificar parâmetros individuais selecionando o parâmetro e escolhendo Editar na seção Parâmetros de cluster. Para obter mais informações, consulte Modificando os parâmetros de cluster do Amazon DocumentDB.
   - Em Eventos recentes, é possível ver os eventos mais recentes para esse grupo de parâmetros. É possível filtrar esses eventos usando a barra de pesquisa nesta seção. Para obter mais informações, consulte Gerenciando eventos do Amazon DocumentDB.
   - Em Tags, você pode ver as tags nesse grupo de parâmetro do cluster. É possível adicionar ou remover tags escolhendo Editar na seção Tags. Para obter mais informações, consulte Marcação de recursos do Amazon DocumentDB.

Using the AWS CLI

Você pode usar o `describe-db-cluster-parameter-groups` AWS CLI comando para visualizar o nome de recurso da Amazon (ARN), a família, a descrição e o nome de um único grupo de parâmetros de cluster ou de todos os grupos de parâmetros de cluster que você tem para o Amazon DocumentDB. Você também pode usar o `describe-db-cluster-parameters` AWS CLI comando para visualizar os parâmetros e seus detalhes dentro de um único grupo de parâmetros do cluster.

- **`--describe-db-cluster-parameter-groups`** — para ver uma lista de todos os grupos de parâmetros de cluster e seus detalhes.

- **`--db-cluster-parameter-group-name`** — Opcional. O nome do grupo de parâmetro de cluster que você deseja descrever. Se esse parâmetro for omitido, todos os grupos de parâmetro de cluster serão descritos.

- **`--describe-db-cluster-parameters`** — para listar todos os parâmetros em um grupo de parâmetros e seus valores.
  - **`--db-cluster-parameter-group name`** — obrigatório. O nome do grupo de parâmetro de cluster que você deseja descrever.

Example

O código a seguir lista até 100 grupos de parâmetros de cluster e seus ARNs, famílias, descrições e nomes.

```
aws docdb describe-db-cluster-parameter-groups
```

A saída dessa operação é semelhante ao seguinte (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

O código a seguir lista o ARN, a família, a descrição e o nome do `sample-parameter-group`.

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante ao seguinte (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

O código a seguir lista os valores dos parâmetros no `sample-parameter-group`.

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante ao seguinte (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"
         }
   ]
}
```

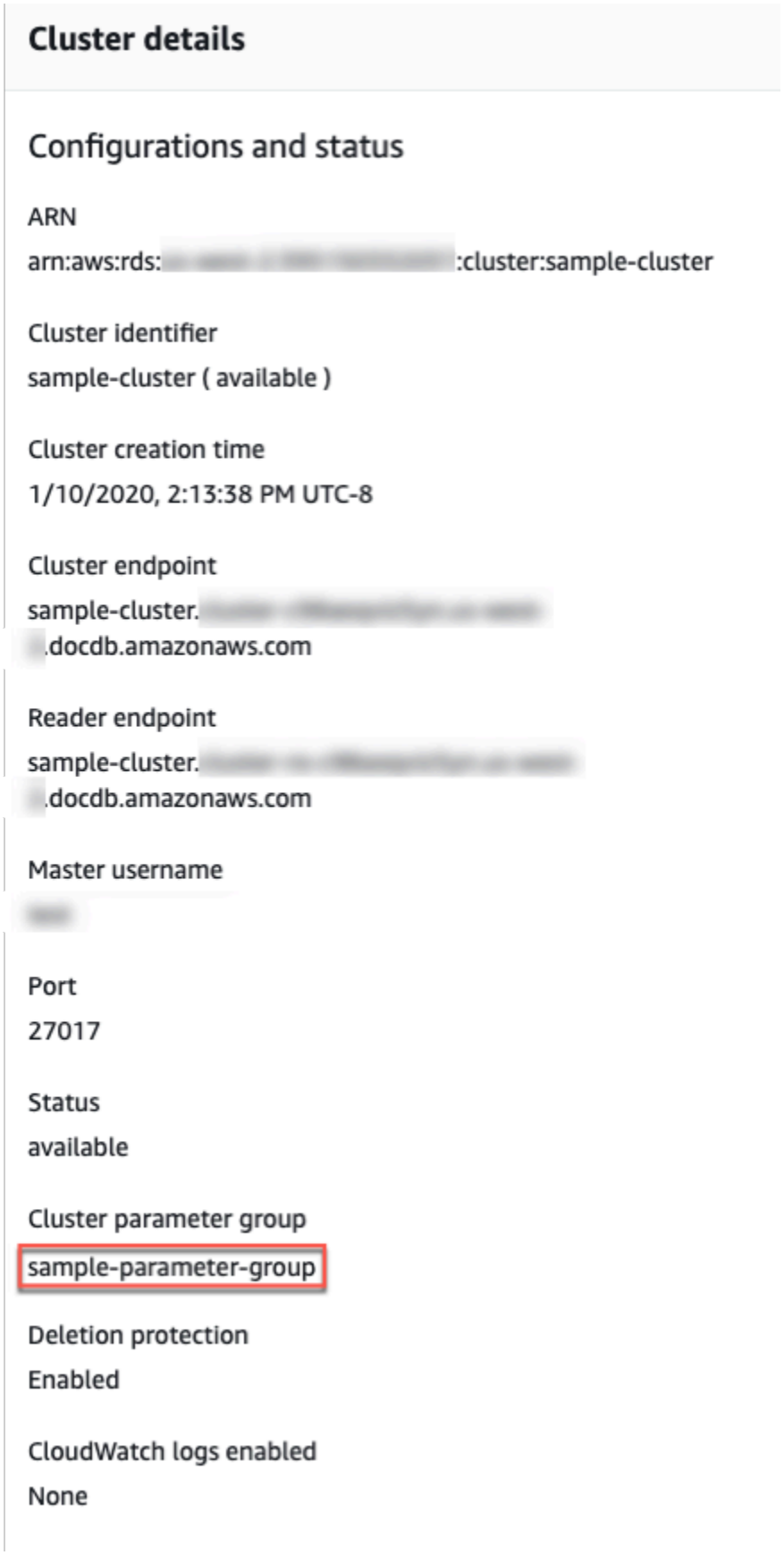
Determinando o grupo de parâmetros de um cluster do Amazon DocumentDB

Para determinar qual grupo de parâmetros está associado a um cluster específico, conclua as etapas a seguir usando o Console de gerenciamento da AWS ou AWS CLI o.

Using the Console de gerenciamento da AWS

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação à esquerda, escolha Clusters.
3. Na lista de clusters, selecione o nome do cluster desejado.

4. A página resultante exibe os detalhes do cluster selecionado. Role para baixo até Detalhes do cluster. Na parte inferior dessa seção, localize o nome do grupo de parâmetros sob Grupo de parâmetros de cluster.


Cluster details
Configurations and status
ARN
arn:aws:rds:[redacted]:cluster:sample-cluster
Cluster identifier
sample-cluster (available)
Cluster creation time
1/10/2020, 2:13:38 PM UTC-8
Cluster endpoint
sample-cluster.[redacted]
.docdb.amazonaws.com
Reader endpoint
sample-cluster.[redacted]
.docdb.amazonaws.com
Master username
[redacted]
Port
27017
Status
available
Cluster parameter group
sample-parameter-group
Deletion protection
Enabled
CloudWatch logs enabled
None

Using the AWS CLI

O AWS CLI código a seguir determina qual grupo de parâmetros está governando o cluster`sample-cluster`.

```
aws docdb describe-db-clusters \
      --db-cluster-identifier sample-cluster \
      --query 'DBClusters[*].[DBClusterIdentifier,DBClusterParameterGroup]'
```

A saída dessa operação é semelhante ao seguinte (formato JSON).

```
[
       [
           "sample-cluster",
           "sample-parameter-group"
       ]
]
```

Determinação de clusters e instâncias associados a um grupo de parâmetros de cluster do Amazon DocumentDB

Para determinar quantos clusters e instâncias estão associados a cada grupo de parâmetros, conclua as etapas a seguir usando AWS CLI o.

Using the AWS CLI

O AWS CLI código a seguir determina a quantos clusters e instâncias estão associados`sample-parameter-group`.

Para Linux, macOS ou 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
      }) | .[]'
```

A saída dessa operação é semelhante ao seguinte (formato JSON).

```
{
   "parameter_group_name": "sample-parameter-group",
   "total_clusters": 50,
   "total_instances": 150
}
```

Criando grupos de parâmetros de cluster do Amazon DocumentDB

Grupos de parâmetros de cluster padrão `default.docdb5.0default.docdb4.0`, como, ou`default.docdb3.6`, são criados quando você cria um cluster com uma nova versão do mecanismo e em uma nova Região da AWS. Clusters subsequentes criados na mesma região e com a mesma versão de mecanismo herdam o grupo de parâmetros do `default` cluster. Depois de criados, os grupos de parâmetros `default` não podem ser excluídos nem renomeados. É possível modificar o comportamento do mecanismo das instâncias de cluster criando um grupo de parâmetros personalizado com valores de parâmetros preferenciais e anexando-o ao seu cluster do Amazon DocumentDB.

O procedimento a seguir orienta na criação de um grupo de parâmetros de cluster personalizado com base na família. Depois disso, é possível modificar os parâmetros desse grupo de parâmetros.

> **Note**
>
> Depois de modificar um grupo de parâmetros de cluster, é necessário esperar pelo menos 5 minutos antes de usar esse grupo de parâmetros em particular. Isso permite que o Amazon

DocumentDB conclua a ação `create` integralmente antes que o grupo de parâmetros de cluster seja usado para um novo cluster. Você pode usar a `describe-db-cluster-parameter-groups` AWS CLI operação Console de gerenciamento da AWS ou para verificar se o grupo de parâmetros do cluster foi criado. Para obter mais informações, consulte Descrever os grupos de parâmetros de cluster do Amazon DocumentDB.

Using the Console de gerenciamento da AWS

Para criar um grupos de parâmetros de cluster

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Grupos de parâmetros.

   > **Tip**
   >
   > Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

3. No painel Grupos de parâmetros de cluster, escolha Criar.
4. No painel Criar grupo de parâmetros do cluster insira o seguinte:

   a. Nome do grupo de parâmetros do novo cluster: insira um nome para o grupo de parâmetros do cluster. Por exemplo, `.sample-parameter-group` Os grupos de parâmetros de cluster têm as seguintes restrições de nomenclatura:

      - O tamanho é de [1–255] caracteres alfanuméricos.
      - O primeiro caractere deve ser uma letra.
      - Não pode terminar com um hífen ou conter dois hifens consecutivos.

   b. Família: escolha uma versão do DocumentDB que você pretende usar para seu cluster.

   c. Descrição — fornece uma descrição para este grupo de parâmetros de cluster.

5. Para criar o grupo de parâmetro de cluster, escolha Criar. Para cancelar a operação, escolha Cancelar.
6. Depois de escolher Criar, o texto a seguir será exibido na parte superior da página, para verificar se o grupo de parâmetros de cluster foi criado com êxito:

```
Successfully created cluster parameter group 'sample-parameter-group'.
```

Using the AWS CLI

Para criar um novo grupo de parâmetros de cluster para clusters do Amazon DocumentDB 4.0, use a AWS CLI `create-db-cluster-parameter-group` operação com os seguintes parâmetros:

- **`--db-cluster-parameter-group-name`** — o nome do grupo de parâmetros de cluster personalizado. Por exemplo, `.sample-parameter-group`
- **`--db-cluster-parameter-group-family`** — a família do grupo de parâmetros de cluster usada como modelo para o grupo de parâmetros de cluster personalizado.
- **`--description`** — a descrição fornecida pelo usuário para esse grupo de parâmetros de cluster. O exemplo a seguir usa "`Custom docdb4.0 parameter group`".

Para Linux, macOS ou 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"
```

A saída dessa operação é semelhante ao seguinte (formato JSON).

```
{
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupName": "sample-parameter-group",
```

```
        "DBParameterGroupFamily": "docdb5.0",
        "Description": "Custom docdb4.0 parameter group",
        "DBClusterParameterGroupArn": "sample-parameter-group-arn"
    }
}
```

Modificando grupos de parâmetros de cluster do Amazon DocumentDB

Esta seção explica como modificar um grupo de parâmetros personalizado do Amazon DocumentDB. No Amazon DocumentDB, você não pode modificar um grupo de parâmetros de `default` cluster que é criado quando você cria pela primeira vez um cluster com a nova versão do mecanismo em uma nova região. Se o cluster do Amazon DocumentDB estiver usando o grupo de parâmetros de cluster padrão e você desejar modificar um valor, primeiro será necessário criar um novo grupo de parâmetros, ou copiar um grupo de parâmetros existente, para então modificá-lo e, em seguida, aplicar o grupo de parâmetros modificado ao seu cluster.

Conclua as etapas a seguir para modificar um grupo personalizado de parâmetros de cluster. As ações de modificação podem demorar um pouco para se propagar. Aguarde até que o grupo de parâmetros do cluster modificado esteja disponível antes de anexá-lo ao seu cluster. Você pode usar a AWS CLI `describe-db-cluster-parameters` operação Console de gerenciamento da AWS ou para verificar se o grupo de parâmetros do cluster foi modificado. Para obter mais informações, consulte Descrevendo grupos de parâmetros de cluster.

Using the Console de gerenciamento da AWS

Siga estas etapas para modificar um grupo de parâmetros do Amazon DocumentDB personalizado. Não é possível modificar um grupo de parâmetros `default`. Se você deseja modificar um valor no grupo de parâmetros `default`, você pode copiar o grupo de parâmetros de cluster padrão, modificá-lo e aplicá-lo ao cluster. Para obter mais informações sobre como aplicar grupos de parâmetros ao cluster, consulte Modificar um cluster do Amazon DocumentDB.

Para modificar um grupos de parâmetros de cluster personalizado

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, no lado esquerdo do console, selecione Grupos de parâmetros. Na lista de grupos de parâmetro, escolha o nome do grupo de parâmetro que deseja modificar.

> **Tip**
>
> Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

3. Para cada parâmetro no grupo de parâmetro que desejar modificar, faça o seguinte:

   a. Localize o parâmetro que deseja modificar e verifique se ele é modificável e se está listado como `true` na coluna Modificável.

   b. Se for modificável, selecione o parâmetro e escolha Editar no canto superior direito da página do console.

   c. Na caixa de diálogo Modificar **<parameter-name>**, faça as alterações que deseja. Depois disso, escolha Modificar parâmetro de cluster ou Cancelar para descartar as alterações.

Using the AWS CLI

É possível modificar o `ParameterValue`, a `Description` ou o `ApplyMethod` de qualquer parâmetro modificável em um grupo de parâmetros de cluster personalizado do Amazon DocumentDB usando a AWS CLI. Não é possível fazer modificações diretamente em um grupo de parâmetros de cluster.

Para modificar os parâmetros de um grupo de parâmetros de cluster personalizado, use a operação `modify-db-cluster-parameter-group` com os parâmetros a seguir.

- **`--db-cluster-parameter-group-name`** — Obrigatório. O nome do grupos de parâmetros de cluster que você está modificando.
- **`--parameters`** — Obrigatório. Os parâmetros que você está modificando. Para obter uma lista dos parâmetros que se aplicam a todas as instâncias em um cluster do Amazon DocumentDB, consulte Referência de parâmetros de cluster do Amazon DocumentDB. Cada entrada de parâmetro deve incluir o seguinte:
  - **`ParameterName`** — O nome do parâmetro que você está modificando.
  - **`ParameterValue`** — o novo valor para esse parâmetro.
  - **`ApplyMethod`** — como você deseja aplicar as alterações nesse parâmetro. Os valores permitidos são `immediate` e `pending-reboot`.

> **Note**
>
> Os parâmetros com `ApplyType` de `static` devem ter um `ApplyMethod` de `pending-reboot`.

Example- Modificando o valor de um parâmetro

Neste exemplo, você lista os valores de parâmetro de `sample-parameter-group` e modifica o parâmetro `tls`. Em seguida, após esperar 5 minutos, você listará novamente os valores dos parâmetros do `sample-parameter-group` para vê-los alterados.

1. Liste os parâmetros e os valores do `sample-parameter-group`.

   Para Linux, macOS ou 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
   ```

   A saída dessa operação é semelhante ao seguinte (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 o parâmetro `tls` para que seu valor seja `disabled`.

   Não é possível modificar o `ApplyMethod` porque o `ApplyType` é `static`.

   Para Linux, macOS ou 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
```

   A saída dessa operação é semelhante ao seguinte (formato JSON).

```
{
       "DBClusterParameterGroupName": "sample-parameter-group"
   }
```

3. Aguarde pelo menos 5 minutos.
4. Liste os valores de parâmetros de `sample-parameter-group` para verificar se o parâmetro `tls` foi modificado.

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante ao seguinte (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"
           }
       ]
}
```

Modificando clusters do Amazon DocumentDB para usar grupos de parâmetros de cluster personalizados

Ao criar um cluster do Amazon DocumentDB, um grupo de parâmetros `default.docdb4.0` é criado automaticamente para esse cluster. Não é possível modificar o grupo de parâmetros de cluster `default`. Em vez disso, é possível modificar seu cluster do Amazon DocumentDB para associar um novo grupo de parâmetros personalizado a ele.

Esta seção explica como modificar um cluster existente do Amazon DocumentDB para usar um grupo de parâmetros de cluster personalizado usando o Console de gerenciamento da AWS e o AWS Command Line Interface (AWS CLI).

Using the Console de gerenciamento da AWS

Como modificar um cluster do Amazon DocumentDB para usar um novo grupo de parâmetros de cluster não padrão

1. Antes de começar, lembre-se de criar um cluster do Amazon DocumentDB e um grupo de parâmetros de cluster. Consulte Criar um cluster do Amazon DocumentDB e Criando grupos de parâmetros de cluster do Amazon DocumentDB para obter mais instruções.
2. Depois de criar o grupo de parâmetros de cluster, abra o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb. No painel de navegação, escolha Clusters para adicionar o novo grupo de parâmetros a um cluster.
3. Escolha o cluster que deseja associar ao seu grupo de parâmetros. Escolha Ações e Modificar para modificar seu cluster.
4. Em Opções de cluster, escolha o novo grupo de parâmetros ao qual deseja associar o cluster.
5. Escolha Continuar para exibir um resumo das modificações.
6. Depois de verificar suas alterações, é possível aplicá-las imediatamente, ou durante a próxima janela de manutenção em Programação de modificações.
7. Escolha Modificar cluster para atualizar seu cluster com o novo grupo de parâmetros.

Using the AWS CLI

Antes de começar, lembre-se de criar um cluster do Amazon DocumentDB e um grupo de parâmetros de cluster. Você pode criar um cluster Amazon DocumentDB usando a AWS CLI

`create-db-cluster` operação. Você pode criar um grupo de parâmetros de cluster usando a AWS CLI `create-db-cluster-parameter-group` operação.

Para adicionar seu novo grupo de parâmetros de cluster ao seu cluster, use a AWS CLI `modify-db-cluster` operação com os parâmetros a seguir.

- --db-cluster-identifier — o nome do seu cluster (por exemplo, `sample-cluster`).
- --db-cluster-parameter-group-name — o nome do grupo de parâmetros ao qual deseja associar seu cluster (por exemplo, `sample-parameter-group`).

Example

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --db-cluster-parameter-group-name sample-parameter-group
```

A saída dessa operação é semelhante ao seguinte (formato JSON).

```
"DBCluster": {
    "AvailabilityZones": [
            "us-west-2c",
            "us-west-2b",
            "us-west-2a"
    ],
    "BackupRetentionPeriod": 1,
    "DBClusterIdentifier": "sample-cluster",
    "DBClusterParameterGroup": "sample-parameter-group",
    "DBSubnetGroup": "default",
   ...
}
```

Copiando grupos de parâmetros de cluster do Amazon DocumentDB

Você pode fazer uma cópia de um grupo de parâmetros de cluster no Amazon DocumentDB usando o Console de gerenciamento da AWS ou o AWS Command Line Interface (AWS CLI).

Using the Console de gerenciamento da AWS

O procedimento a seguir orienta durante a criação de um novo grupo de parâmetros de cluster fazendo uma cópia de um grupo de parâmetros de cluster existente.

Como copiar um grupo de parâmetros de cluster

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, selecione Grupos de parâmetros.
3. No painel Grupos de parâmetros de cluster, escolha o nome do grupo de parâmetros de cluster que deseja copiar.
4. Escolha Ações e Copiar para copiar esse grupo de parâmetros.
5. Em Opções de cópia, insira um nome e uma descrição para o novo grupo de parâmetros de cluster. Depois disso, escolha Copiar para salvar as alterações.

Using the AWS CLI

Para fazer uma cópia de um grupo de parâmetro de cluster, use a operação `copy-db-cluster-parameter-group` com os parâmetros a seguir.

- **`--source-db-cluster-parameter-group-identifier`** — Obrigatório. O nome ou nome do recurso da Amazon (ARN) do grupo de parâmetro de cluster cuja cópia você deseja fazer.

  Se os grupos de parâmetros do cluster de origem e de destino estiverem iguais Região da AWS, o identificador poderá ser um nome ou um ARN.

  Se os grupos de parâmetros do cluster de origem e de destino estiverem diferentes Regiões da AWS, o identificador deverá ser um ARN.
- **`--target-db-cluster-parameter-group-identifier`** — Obrigatório. O nome ou ARN da cópia do grupo de parâmetro do cluster.

  Restrições:
  - Não pode ser nulo, vazio ou estar em branco.
  - Deve conter de 1 a 255 letras, números ou hifens.
  - O primeiro caractere deve ser uma letra.
  - Não pode terminar com um hífen ou conter dois hifens consecutivos.
- **`--target-db-cluster-parameter-group-description`** — Obrigatório. Uma descrição fornecida pelo usuário para a cópia do grupo de parâmetro de cluster.

Example

O código a seguir faz uma cópia de `sample-parameter-group`, nomeando a cópia `sample-parameter-group-copy`.

Para Linux, macOS ou 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"
```

A saída dessa operação é semelhante ao seguinte (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"
    }
}
```

Redefinindo grupos de parâmetros de cluster do Amazon DocumentDB

Você pode redefinir alguns ou todos os valores de parâmetros de um grupo de parâmetros de cluster do Amazon DocumentDB para seus valores padrão usando o Console de gerenciamento da AWS ou o AWS Command Line Interface (AWS CLI) para redefinir o grupo de parâmetros do cluster.

Using the Console de gerenciamento da AWS

Siga estas etapas para redefinir alguns ou todos os valores de parâmetros de um grupo de parâmetros de cluster para seus valores padrão.

Como redefinir os valores de parâmetros de um grupo de parâmetros de cluster

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, no lado esquerdo do console, selecione Grupos de parâmetros.
3. No painel Grupos de parâmetros de cluster, escolha o nome do grupo de parâmetros de cluster que deseja redefinir.
4. Escolha Ações e Redefinir para redefinir esse grupo de parâmetros.
5. Na página Confirmação de redefinição do grupo de parâmetros de cluster resultante, confirme se você deseja redefinir todos os parâmetros de cluster para esse grupo de parâmetros para seus padrões. Depois disso, escolha Redefinir para redefinir o grupo de parâmetros. Também é possível escolher Cancelar para cancelar as alterações.

Using the AWS CLI

Para redefinir alguns ou todos os valores de parâmetro do grupos de parâmetros de cluster para seus valores padrão, use a operação `reset-db-cluster-parameter-group` com os parâmetros a seguir.

- **`--db-cluster-parameter-group-name`** — Obrigatório. O nome do grupo de parâmetro de cluster a ser redefinido.
- **`--parameters`** — Opcional. Uma lista de `ParameterName` e `ApplyMethod` no grupo de parâmetro de cluster para redefinir para seus valores padrão. Os parâmetros estáticos devem ser definidos como `pending-reboot` para entrar em vigor na próxima reinicialização da instância ou na solicitação `reboot-db-instance`. É necessário chamar `reboot-db-instance` para cada instância no cluster cujo parâmetro estático atualizado desejar aplicar.

  Esse parâmetro e `--reset-all-parameters` são mutuamente exclusivos: você pode usar um, mas não ambos.
- **`--reset-all-parameters`** ou **`--no-reset-all-parameters`** — opcional. Especifica se é necessário redefinir todos os parâmetros (`--reset-all-parameters`) ou apenas alguns dos parâmetros (`--no-reset-all-parameters`) para seus valores padrão. O parâmetro `--reset-all-parameters` e `--parameters` são mutuamente exclusivos: você pode usar um, mas não ambos.

  Quando você redefinir o grupo inteiro, os parâmetros dinâmicos serão atualizados imediatamente. Os parâmetros estáticos são definidos como `pending-reboot` para entrarem

em vigor na próxima reinicialização da instância, ou na solicitação `reboot-db-instance`. É necessário chamar `reboot-db-instance` para cada instância no cluster na qual deseja que o parâmetro estático atualizado seja aplicado.

Example

Exemplo 1: redefinindo todos os parâmetros para seus valores padrão

O código a seguir redefine todos os parâmetros no grupos de parâmetros de cluster `sample-parameter-group` para seus valores padrão.

Para Linux, macOS ou 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
```

Exemplo 2: redefinindo os parâmetros especificados para seus valores padrão

O código a seguir redefine o parâmetro `tls` no grupo de parâmetro de cluster `sample-parameter-group` para seu valor padrão.

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante ao seguinte (formato JSON).

```
{
        "DBClusterParameterGroupName": "sample-parameter-group"
   }
```

Reiniciando uma instância de cluster

Antes que o valor de um parâmetro estático seja alterado, a instância do cluster deve ser reiniciada. Reinicie cada instância no cluster a qual deseje que o parâmetro estático atualizado seja aplicado.

Para Linux, macOS ou 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
```

Excluindo grupos de parâmetros de cluster do Amazon DocumentDB

Você pode excluir um grupo de parâmetros de cluster personalizado do Amazon DocumentDB usando o Console de gerenciamento da AWS ou o AWS Command Line Interface (AWS CLI). Não é possível excluir o grupo de parâmetros de cluster `default.docdb4.0`.

Using the Console de gerenciamento da AWS

Para excluir um grupo de parâmetros de cluster

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Grupos de parâmetros.

> **Tip**
>
> Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

3. No painel Grupos de parâmetro, escolha o botão de opção à esquerda do grupos de parâmetros de cluster que deseje excluir.
4. Escolha Ações e, em seguida, Excluir.
5. No painel de confirmação Excluir, escolha Excluir para excluir o grupo de parâmetros de cluster. Para manter o grupo de parâmetros de cluster, escolha Cancelar.

Using the AWS CLI

Para excluir um grupo de parâmetros de cluster, use a operação `delete-db-cluster-parameter-group` com o parâmetro a seguir.

- **`--db-cluster-parameter-group-name`** — Obrigatório. O nome do grupo de parâmetro de cluster a ser excluído. Ele deve ser um grupo de parâmetro de cluster existente. Não é possível excluir o grupo de parâmetros de cluster *`default.docdb4.0`*.

Example- Excluindo um grupo de parâmetros de cluster

O exemplo a seguir mostra as três etapas para excluir um grupo de parâmetros de cluster:

1. Descobrindo o nome do grupos de parâmetros de cluster que deseja excluir.
2. Excluindo o grupo de parâmetros de cluster especificado.
3. Verificando se o grupos de parâmetros de cluster foi excluído.

1. Localize o nome do grupos de parâmetros de cluster que deseja excluir.

A lista de códigos a seguir elenca os nomes de todos os grupos de parâmetro de cluster.

Para Linux, macOS ou Unix:

```
aws docdb describe-db-cluster-parameter-groups \
```

```
        --query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

Para Windows:

```
aws docdb describe-db-cluster-parameter-groups ^
        --query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

A saída da operação anterior é uma lista dos nomes dos grupos de parâmetros de cluster semelhantes à seguinte (formato JSON).

```
[
        [
            "default.docdb4.0"
        ],
        [
            "sample-parameter-group"
        ],
        [
            "sample-parameter-group-copy"
        ]
   ]
```

2. Exclua um grupos de parâmetros de cluster específico.

O código a seguir exclui o grupos de parâmetros de cluster `sample-parameter-group-copy`.

Para Linux, macOS ou 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
```

Não há saída dessa operação.

3. Verifique se o grupo de parâmetros de cluster especificado foi excluído.

O código a seguir lista os nomes de todos os grupos de parâmetro de cluster restantes.

Para Linux, macOS ou Unix:

```
aws docdb describe-db-cluster-parameter-groups \
       --query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

Para Windows:

```
aws docdb describe-db-cluster-parameter-groups ^
       --query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

A saída da operação anterior é uma lista dos grupos de parâmetro de cluster semelhantes ao seguinte (formato JSON). O grupo de parâmetros de cluster que você acabou de excluir não deve estar na lista.

A saída dessa operação é semelhante ao seguinte (formato JSON).

```
[
       [
           "default.docdb4.0"
       ],
       [
           "sample-parameter-group"
       ]
   ]
```

Referência de parâmetros de cluster do Amazon DocumentDB

Quando você altera um parâmetro dinâmico e salva o grupo de parâmetro do cluster, a alteração é aplicada imediatamente, independente da configuração Aplicar imediatamente. Quando você altera um parâmetro estático e salva o grupo de parâmetro do cluster, a alteração do parâmetro entra em vigor depois que você reinicia manualmente a instância do cluster. É possível reiniciar uma instância usando o console do Amazon DocumentDB, ou chamando explicitamente `reboot-db-instance`.

A tabela a seguir mostra os parâmetros que se aplicam a todas as instâncias em um cluster do Amazon DocumentDB.

Parâmetros no nível do cluster do Amazon DocumentDB

Parâmetro	Valor padrão	Valores válidos	Modificável	Aplicar tipo	Tipo de dado	Description
`audit_logs`	desabilitado	ativado, desativado, ddl, dml_read, dml_write, todos, nenhum	Sim	Dinâmico	String	Define se os registros CloudWatch de auditoria da Amazon estão habilitados. • **`enabled`**— os registros de CloudWatch auditoria estão habilitados. • **`disabled`**— os registros CloudWatch de auditoria estão desativados. • **`ddl`** — a auditoria de

Parâmetro	Valor padrão	Valores válidos	Modificável	Aplicar tipo	Tipo de dado	Description
						eventos de DDL está habilitada. • **`dml_read`**: a auditoria de eventos de gravação em DML está habilitada. • **`dml_write`** — a auditoria de eventos de gravação em DML está habilitada. • **`all`** — a auditoria de todos os eventos do banco

Parâmetro	Valor padrão	Valores válidos	Modificável	Aplicar tipo	Tipo de dado	Description
						de dados está habilitada. • **none** — a auditoria está desabilitada.
`change_stream_log_retention_duration`	10800	3600-604800	Sim	Dinâmico	Inteiro	Define a duração do tempo (em segundos) na qual o log do fluxo de alteração é retido e pode ser consumido.

Parâmetro	Valor padrão	Valores válidos	Modificável	Aplicar tipo	Tipo de dado	Description
`default_collection_compression`	desabilitado	ativado, desativado (Amazon DocumentDB 5.0)/ zstd, lz4, nenhum (Amazon) DocumentDB8.0	Sim	Dinâmico	String	Define a configuração de compactação padrão para novas coleções em um cluster • **`enabled`**— A compactação Lz4 é habilitada por padrão no Amazon DocumentDB 5.0. A compactação zstd é habilitada por padrão no Amazon DocumentDB 8.0. • **`disabled, none`**— a compactaç

Parâmetro	Valor padrão	Valores válidos	Modificável	Aplicar tipo	Tipo de dado	Description
						ão é desativada por padrão (desativada no Amazon DocumentDB 5.0, nenhuma no Amazon DocumentDB 8.0).

Parâmetro	Valor padrão	Valores válidos	Modificável	Aplicar tipo	Tipo de dado	Description
`profiler`	desabilitado	habilitado, desabilitado	Sim	Dinâmico	String	Habilita a criação do perfil para operações lentas. • **`enabled`**— as operações que demoram mais do que um valor limite definido pelo cliente (por exemplo, 100 ms) são registradas no Amazon Logs. CloudWatch • **`disabled`**— operações lentas não são registradas nos

Parâmetro	Valor padrão	Valores válidos	Modificável	Aplicar tipo	Tipo de dado	Description
						CloudWatch Logs.
`profiler_sampling_rate`	1,0	0.0-1.0	Sim	Dinâmico	Float	Define a taxa de amostragem de operações registradas.
`profiler_threshold_ms`	100	50-2147483646	Sim	Dinâmico	Inteiro	Define o limite para `profiler`. • Todas as operações maiores que `profiler_threshold_ms` são registradas no CloudWatch Logs.

Parâmetro	Valor padrão	Valores válidos	Modificável	Aplicar tipo	Tipo de dado	Description
`planner_version`	3.0	1,0, 2,0, 3,0	Sim	Dinâmico	Float	Define a versão do planejador de consultas a ser usada para consultas. • **`1.0`**— Compatível com o Amazon DocumentDB 5.0. • **`2.0`**— Melhorias para encontrar e atualizar operadores. Compatível com o Amazon DocumentDB 5.0. • **`3.0`**— Melhorias nos operadores de

Parâmetro	Valor padrão	Valores válidos	Modificável	Aplicar tipo	Tipo de dado	Description
						localização, atualização e agregação. Planejador de consultas padrão no Amazon DocumentDB 8.0.

Parâmetro	Valor padrão	Valores válidos	Modificável	Aplicar tipo	Tipo de dado	Description
`tls`	habilitado	habilitado, desabilitado, fips-140-3, tls1.2+, tls1.3+	Sim	Estático	String	Define se as conexões Transport Layer Security (TLS) são necessárias. • **`enabled`** — conexões TLS são necessárias para conectar. Para o Amazon DocumentDB 5.0 e versões anteriores, isso habilita o TLS 1.0 a 1.3, mas usa o TLS 1.2 ou superior. Somente o TLS 1.2 e

Parâmetro	Valor padrão	Valores válidos	Modificável	Aplicar tipo	Tipo de dado	Description
						superior é compatível com o Amazon DocumentDB 5.0, começando com a versão secundária 5.0.1, Amazon DocumentDB 8.0 e versões mais recentes. • **`disabled`** — conexões TLS não podem ser usadas para conectar. • **`fips-140-3`** : para conectar, são necessárias

Parâmetro	Valor padrão	Valores válidos	Modificável	Aplicar tipo	Tipo de dado	Description
						conexões TLS com atributos Federal Information Process Standards (FIPS). O cluster só aceita conexões seguras em acordo com a publicação 140-3 do FIPS. Isso só é suportado a partir dos clusters Amazon DocumentDB 5.0 (engine versão 3.0.3727) nas seguintes regiões: ca-

Parâmetro	Valor padrão	Valores válidos	Modificável	Aplicar tipo	Tipo de dado	Description
						central-1, us-west-2, us-east-1, us-east-2, us-gov-east-1, us-gov-west-1. • **`tls1.2+`**: conexões TLS usando TLS versão 1.2 e posteriores são necessárias para se conectar. Só há suporte para isso a partir do Amazon DocumentDB 4.0 (versão de

Parâmetro	Valor padrão	Valores válidos	Modificável	Aplicar tipo	Tipo de dado	Description
						mecanismo 2.0.10980) e do Amazon DocumentDB 5.0 (versão de mecanismo 3.0.11051). • **`tls1.3+`**: conexões TLS usando TLS versão 1.3 e posteriores são necessárias para se conectar. Só há suporte para isso a partir do Amazon DocumentDB 4.0 (versão de

Parâmetro	Valor padrão	Valores válidos	Modificável	Aplicar tipo	Tipo de dado	Description
						mecanismo 2.0.10980) e do Amazon DocumentDB 5.0 (versão de mecanismo 3.0.11051).

Parâmetro	Valor padrão	Valores válidos	Modificável	Aplicar tipo	Tipo de dado	Description
`ttl_monitor`	habilitado	habilitado, desabilitado	Sim	Dinâmico	String	Define se o monitoramento Time to Live (TTL) está habilitado para o cluster. • **`enabled`** — o monitoramento de TTL está habilitado. • **`disabled`** — o monitoramento de TTL está desabilitado.

Modificando os parâmetros de cluster do Amazon DocumentDB

No Amazon DocumentDB, os grupos de parâmetros de cluster consistem em parâmetros que se aplicam a todas as instâncias criadas no cluster. Para grupos de parâmetros de cluster personalizados, é possível modificar um valor de parâmetro a qualquer momento, ou redefinir todos os valores de parâmetro como padrão para os grupos de parâmetros criados. Esta seção descreve como visualizar os parâmetros e valores de um grupo de parâmetros de cluster do Amazon DocumentDB, além de mostrar como alterar ou atualizar esses valores.

Os parâmetros podem ser dinâmicos ou estáticos. Ao alterar um parâmetro dinâmico e salvar o grupo de parâmetros de cluster, a alteração é aplicada imediatamente, independente da configuração `Apply Immediately`. Quando você altera um parâmetro estático e salva o grupo de parâmetros do cluster, a alteração do parâmetro entra em vigor depois que você reinicia manualmente a instância do cluster.

Visualizando os parâmetros de um grupo de parâmetros de cluster do Amazon DocumentDB

Você pode ver os parâmetros de um cluster Amazon DocumentDB e seus valores usando o Console de gerenciamento da AWS ou AWS CLI.

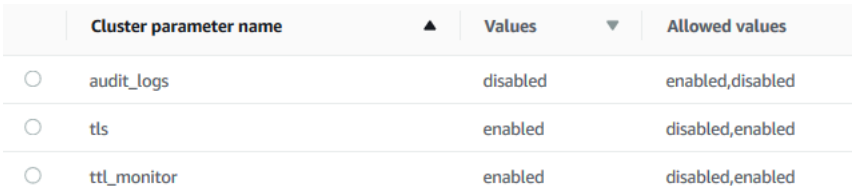
Using the Console de gerenciamento da AWS

Para visualizar os detalhes de um grupos de parâmetros de cluster

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Grupos de parâmetros.

> **Tip**
>
> Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

3. No painel grupos de parâmetro, escolha o nome do grupo de parâmetro do cluster cujos detalhes deseja visualizar.
4. A página resultante exibe os seguintes valores para cada parâmetro: o nome, o valor atual, os valores permitidos, se o parâmetro é modificável, o tipo de aplicação, o tipo de dado e a descrição.



Cluster parameter name	Values	Allowed values
audit_logs	disabled	enabled,disabled
tls	enabled	disabled,enabled
ttl_monitor	enabled	disabled,enabled

Using the AWS CLI

Para ver os parâmetros de um grupos de parâmetros de cluster e seus valores, use a operação `describe-db-cluster-parameters` com os seguintes parâmetros.

- **`--db-cluster-parameter-group-name`** — Obrigatório. O nome do grupos de parâmetros de cluster cuja lista detalhada de parâmetros você deseja.
- **`--source`** — Opcional. Se fornecido, retorna apenas parâmetros para uma origem específica. As origens de parâmetros podem ser `engine-default`, `system` ou `user`.

Example

O código a seguir lista os parâmetros e seus valores para o grupo de parâmetros `custom3-6-param-grp`. Para obter mais informações sobre o grupo de parâmetros, omita a linha `--query`. Para obter informações sobre todos os grupos de parâmetros, omita a linha `--db-cluster-parameter-group-name`.

Para Linux, macOS ou 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]'
```

A saída dessa operação é semelhante ao seguinte (formato JSON).

```
[
    [
        "audit_logs",
        "disabled"
    ],
    [
        "tls",
        "enabled"
    ],
    [
```

```
        "ttl_monitor",
        "enabled"
    ]
]
```

Modificando parâmetros de um grupo de parâmetros de cluster do Amazon DocumentDB

Você pode modificar os parâmetros de um grupo de parâmetros usando o Console de gerenciamento da AWS ou AWS CLI.

Using the Console de gerenciamento da AWS

Como atualizar os parâmetros de um grupo de parâmetros de cluster

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Grupos de parâmetros.

   > **Tip**
   >
   > Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

3. No painel Grupos de parâmetros, selecione um grupo de parâmetros do cluster cujos parâmetros deseja atualizar.
4. A página resultante mostra os parâmetros e os detalhes correspondentes para este grupo de parâmetros do cluster. Selecione um parâmetro para atualizar.
5. No canto superior direito da página, selecione Editar para alterar o valor do parâmetro. Para obter mais informações sobre os tipos de parâmetros de cluster, consulte Referência de parâmetros de cluster do Amazon DocumentDB.
6. Faça a alteração e escolha Modificar parâmetro de cluster para salvar as alterações. Para descartar as alterações, escolha Cancelar.

Using the AWS CLI

Para modificar os parâmetros de um grupo de parâmetros de cluster, use a operação `modify-db-cluster-parameter-group` com os parâmetros a seguir:

- **`--db-cluster-parameter-group-name`** — Obrigatório. O nome do grupos de parâmetros de cluster que você está modificando.
- **`--parameters`** — Obrigatório. O parâmetro ou os parâmetros que você está modificando. Cada entrada de parâmetro deve incluir o seguinte:
  - **`ParameterName`** — O nome do parâmetro que você está modificando.
  - **`ParameterValue`** — o novo valor para esse parâmetro.
  - **`ApplyMethod`** — como você deseja aplicar as alterações nesse parâmetro. Os valores permitidos são `immediate` e `pending-reboot`.

    > **Note**
    >
    > Os parâmetros com `ApplyType` de `static` devem ter um `ApplyMethod` de `pending-reboot`.

Para alterar os valores dos parâmetros de um grupo de parâmetros de cluster (AWS CLI)

O exemplo a seguir altera o parâmetro `tls`.

1. Liste os parâmetros e os valores do **`sample-parameter-group`**

   Para Linux, macOS ou 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
   ```

   A saída dessa operação é semelhante ao seguinte (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 o parâmetro **`tls`** para que seu valor seja **`disabled`**. Não é possível modificar o `ApplyMethod` porque o `ApplyType` é `static`.

   Para Linux, macOS ou 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"
```

   A saída dessa operação é semelhante ao seguinte (formato JSON).

```
{
```

```
    "DBClusterParameterGroupName": "sample-parameter-group"
}
```

3. Aguarde pelo menos 5 minutos.
4. Liste os valores dos parâmetros do **`sample-parameter-group`**.

   Para Linux, macOS ou 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
```

   A saída dessa operação é semelhante ao seguinte (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"
        }
```

```
    ]
}
```

Noções básicas sobre endpoints do Amazon DocumentDB

É possível usar endpoints do Amazon DocumentDB (compatível com MongoDB) para se conectar a um cluster ou instância. O Amazon DocumentDB tem três tipos diferentes de endpoints, cada um com sua própria finalidade.

Tópicos

- Localizar os endpoints de um cluster
- Localizar o endpoint de uma instância
- Conectar-se a endpoints

Endpoint do cluster

Um endpoint de cluster é um endpoint para um cluster do Amazon DocumentDB que se conecta à instância principal atual do cluster. Cada cluster do Amazon DocumentDB tem um único endpoint de cluster e uma instância principal. No caso de um failover, o endpoint do cluster é remapeado para a nova instância principal.

Endpoint de leitor

Um endpoint de leitor é o endpoint de um cluster do Amazon DocumentDB que se conecta a uma das réplicas disponíveis para esse cluster. Cada cluster do Amazon DocumentDB tem um endpoint de leitor. Se houver mais de uma réplica, o endpoint de leitor direcionará cada solicitação de conexão para uma das réplicas do Amazon DocumentDB.

Endpoint da instância

Um endpoint de instância é um endpoint que se conecta a uma instância específica. Cada instância em um cluster, independentemente de ser uma instância principal ou de réplica, tem seu próprio endpoint de instância exclusivo. É melhor não usar endpoints de instância na sua aplicação. Isso ocorre porque eles podem alterar funções no caso de um failover, exigindo alterações de código em sua aplicação.

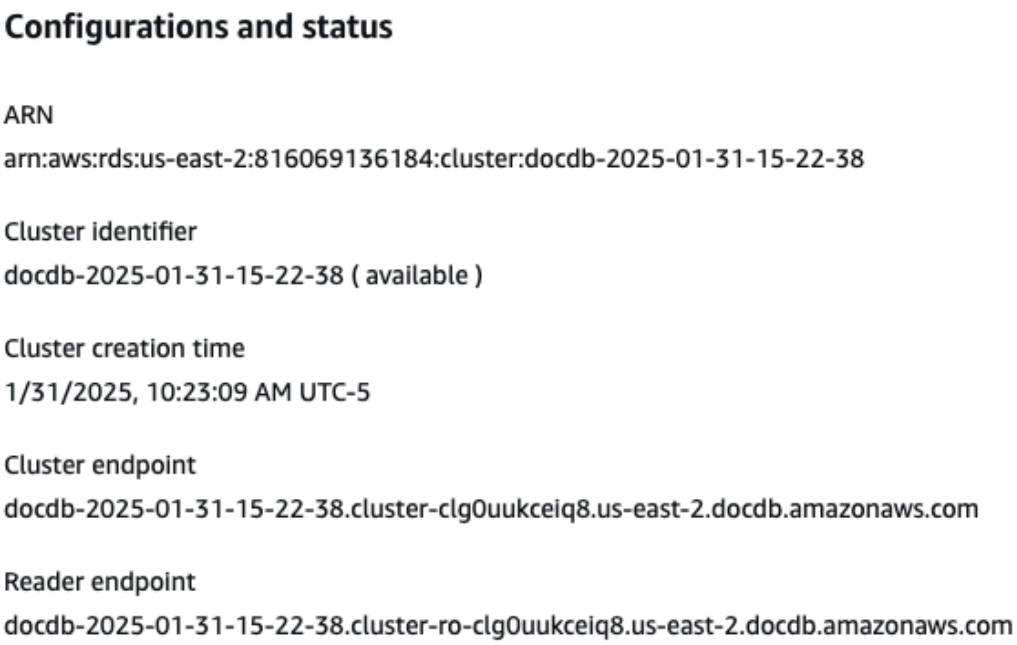
Localizar os endpoints de um cluster

É possível localizar o endpoint do cluster e o endpoint do leitor usando o console ou a Amazon DocumentDB do AWS CLI.

Using the Console de gerenciamento da AWS

Para localizar os endpoints de um cluster usando o console:

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em. https://console.aws.amazon.com/docdb
2. No painel de navegação, escolha Clusters.
3. Na lista de clusters, escolha o nome do cluster desejado.
4. Na página de detalhes do cluster, selecione a guia Configuração. Na seção Configurações e status, você encontrará o Endpoint do cluster e o Endpoint do leitor.



5. Para se conectar a esse cluster, selecione a guia Conectividade e segurança. Localize a string de conexão para o shell do `mongo` e a string de conexão que pode ser usada no código da aplicação para se conectar ao cluster.



Using the AWS CLI

Para encontrar o cluster e os endpoints do leitor para seu cluster usando o AWS CLI, execute o `describe-db-clusters` comando com esses parâmetros.

Parâmetros

- **`--db-cluster-identifier`**—Opcional. Especifica o cluster para o qual retornar os endpoints. Se omitido, retorna endpoints para até 100 dos seus clusters.
- **`--query`**—Opcional. Especifica os campos a serem exibidos. Isso é útil ao reduzir a quantidade de dados que você precisa visualizar para localizar os endpoints. Se omitido, todas as informações sobre um cluster serão retornadas.
- **`--region`**—Opcional. Use o parâmetro `--region` para especificar a região à qual deseja aplicar o comando. Se omitido, sua região padrão será usada.

Example

O exemplo a seguir retorna o `DBClusterIdentifier`, o endpoint (endpoint de cluster) e o `ReaderEndpoint` para o `sample-cluster`.

Para Linux, macOS ou 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]'
```

A saída dessa operação é semelhante ao seguinte (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"
  ]
]
```

Agora que você tem o endpoint de cluster, é possível se conectar ao cluster usando o `mongo` ou o `mongodb`. Para obter mais informações, consulte Conectar-se a endpoints.

Localizar o endpoint de uma instância

É possível encontrar o endpoint de uma instância de banco de dados usando o console do Amazon DocumentDB ou a AWS CLI.

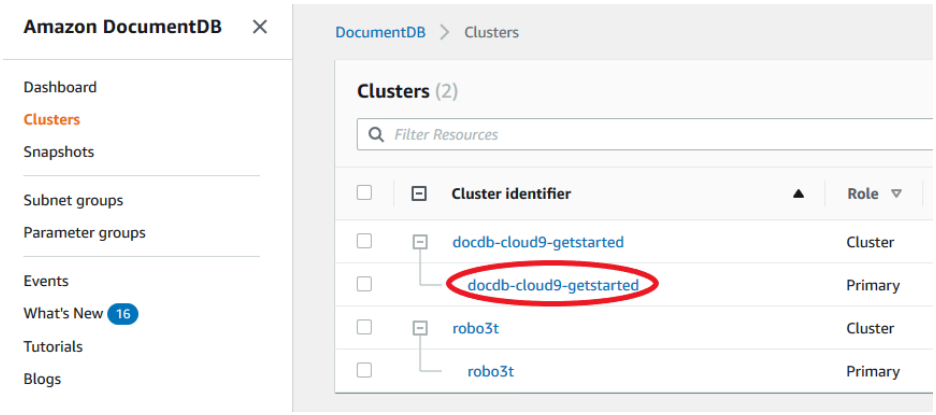
Using the Console de gerenciamento da AWS

Para localizar o endpoint de uma instância usando o console

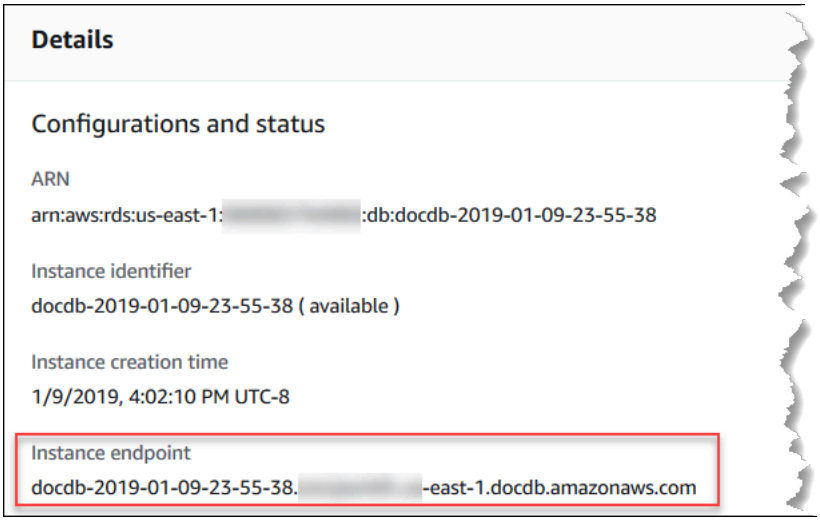
1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em. https://console.aws.amazon.com/docdb
2. No painel de navegação, escolha Clusters.

   > **Tip**
   >
   > Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

3. Na caixa de navegação Clusters, você verá a coluna Identificador do Cluster. Suas instâncias estão listadas em clusters, semelhante à captura de tela a seguir.



4. Marque a caixa à esquerda da instância na qual você está interessado.
5. Role para baixo até a seção Detalhes, em seguida, localize o endpoint da instância.



6. Para se conectar a essa instância, role até a seção Conectar. Localize a string de conexão para o shell mongo e uma string de conexão que pode ser usada no código da aplicação para se conectar à instância.



Using the AWS CLI

Para encontrar o endpoint da instância usando o AWS CLI, execute o comando a seguir com esses argumentos.

Argumentos

- **`--db-instance-identifier`**—Opcional. Especifica a instância para a qual retornar o endpoint. Se omitido, retorna o endpoint para até 100 das suas instâncias.
- **`--query`**—Opcional. Especifica os campos a serem exibidos. Isso é útil ao reduzir a quantidade de dados que você precisa visualizar para localizar os endpoints. Se omitido, todas as informações em uma instância serão retornadas. O campo `Endpoint` tem três membros, portanto, listá-lo na consulta, como no exemplo a seguir, retorna todos os três membros. Se você quiser apenas alguns dos membros `Endpoint`, substitua `Endpoint` na consulta pelos membros desejados, como no segundo exemplo.
- **`--region`**—Opcional. Use o parâmetro `--region` para especificar a região à qual deseja aplicar o comando. Se omitido, sua região padrão será usada.

Example

Para Linux, macOS ou 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]'
```

A saída dessa operação é semelhante ao seguinte (formato JSON).

```
[
    [
        "sample-cluster-instance",
        {
            "Port": 27017,
            "Address": "sample-cluster-instance.corcjozrlsfc.us-
east-1.docdb.amazonaws.com",
            "HostedZoneId": "Z2R2ITUGPM61AM"
```

```
        }
    ]
]
```

Ao reduzir a saída para eliminar o `HostedZoneId`do endpoint, é possível modificar sua consulta especificando `Endpoint.Port` e `Endpoint.Address`.

Para Linux, macOS ou 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]'
```

A saída dessa operação é semelhante ao seguinte (formato JSON).

```
[
    [
        "sample-cluster-instance",
        27017,
        "sample-cluster-instance.corcjozrlsfc.us-east-1.docdb.amazonaws.com"
    ]
]
```

Agora que você tem o endpoint de instância, é possível se conectar à instância usando o `mongo` ou o `mongodb`. Para obter mais informações, consulte Conectar-se a endpoints.

Conectar-se a endpoints

Quando você tiver o endpoint, seja de cluster ou instância, você pode se conectar a ele usando o shell `mongo` ou uma string de conexão.

Conectar-se usando o shell mongo

Use a seguinte estrutura para construir a string que você precisa para se conectar ao cluster ou à instância usando o shell `mongo`:

```
mongo \
    --ssl \
    --host Endpoint:Port \
    --sslCAFile global-bundle.pem \
    --username UserName \
    --password Password
```

exemplos de conchas **`mongo`**

Conecte-se a um cluster:

```
mongo \
    --ssl \
    --host sample-cluster.corcjozrlsfc.us-east-1.docdb.amazonaws.com:27017 \
    --sslCAFile global-bundle.pem \
    --username UserName \
    --password Password
```

Conecte-se a uma instância:

```
mongo \
    --ssl \
    --host sample-cluster-instance.corcjozrlsfc.us-east-1.docdb.amazonaws.com:27017 \
    --sslCAFile global-bundle.pem \
    --username UserName \
    --password Password
```

Conectar-se usando uma string de conexão

Use a seguinte estrutura para construir a string de conexão que você precisa para se conectar ao cluster ou à instância.

```
mongodb://UserName:Password@endpoint:port?replicaSet=rs0&ssl_ca_certs=global-
bundle.pem
```

Exemplos de string de conexão

Conecte-se a um cluster:

```
mongodb://UserName:Password@sample-cluster.cluster-corlsfccjozr.us-
east-1.docdb.amazonaws.com:27017?replicaSet=rs0&ssl_ca_certs=global-bundle.pem
```

Conecte-se a uma instância:

```
mongodb://UserName:Password@sample-cluster-instance.cluster-corlsfccjozr.us-
east-1.docdb.amazonaws.com:27017?replicaSet=rs0&ssl_ca_certs=global-bundle.pem
```

Compreender os nomes do recursos da Amazon (ARN) do Amazon DocumentDB

Cada um dos recursos que você cria AWS é identificado exclusivamente com um nome de recurso da Amazon (ARN). Para determinadas operações do Amazon DocumentDB (compatível com MongoDB), você precisará identificar exclusivamente um recurso do Amazon DocumentDB especificando seu ARN. Por exemplo, quando você adiciona uma tag a um recurso, deve fornecer o ARN do recurso.

Tópicos

- Criar um ARN para um recurso do Amazon DocumentDB
- Localizar um ARN de recurso do Amazon DocumentDB

Criar um ARN para um recurso do Amazon DocumentDB

É possível criar um ARN para um recurso do Amazon DocumentDB usando a seguinte sintaxe. O Amazon DocumentDB compartilha o formato de ARNs do Amazon Relational Database Service (Amazon RDS). Os ARNs do Amazon DocumentDB contêm `rds` e não `docdb`.

`arn:aws:rds:region:account_number:resource_type:resource_id`

Nome da região	Região	Zonas de disponibilidade (computação)
Leste dos EUA (Ohio)	`us-east-2`	3

Nome da região	Região	Zonas de disponibilidade (computação)
Leste dos EUA (Norte da Virgínia)	us-east-1	6
Oeste dos EUA (Oregon)	us-west-2	4
África (Cidade do Cabo)	af-south-1	3
América do Sul (São Paulo)	sa-east-1	3
Ásia-Pacífico (Hong Kong)	ap-east-1	3
Ásia-Pacífico (Hyderabad)	ap-south-2	3
Ásia-Pacífico (Malásia)	ap-southeast-5	3
Ásia-Pacífico (Mumbai)	ap-south-1	3
Ásia-Pacífico (Osaka)	ap-northeast-3	3
Ásia-Pacífico (Seul)	ap-northeast-2	4
Ásia-Pacífico (Singapura)	ap-southeast-1	3
Ásia-Pacífico (Sydney)	ap-southeast-2	3
Ásia-Pacífico (Jacarta)	ap-southeast-3	3

Nome da região	Região	Zonas de disponibilidade (computação)
Ásia-Pacífico (Melbourne)	ap-southeast-4	3
Ásia-Pacífico (Tailândia)	ap-southeast-7	3
Ásia-Pacífico (Tóquio)	ap-northeast-1	3
Canadá (Central)	ca-central-1	3
Oeste do Canadá (Calgary)	ca-west-1	3
Região China (Pequim)	cn-north-1	3
China (Ningxia)	cn-northwest-1	3
Europa (Frankfurt)	eu-central-1	3
Europa (Zurique)	eu-central-2	3
Europa (Irlanda)	eu-west-1	3
Europa (Londres)	eu-west-2	3
Europa (Milão)	eu-south-1	3
Europa (Paris)	eu-west-3	3
Europa (Espanha)	eu-south-2	3
Europa (Estocolmo)	eu-north-1	3
México (Centro)	mx-central-1	3
Oriente Médio (Emirados Árabes Unidos)	me-central-1	3

Nome da região	Região	Zonas de disponibilidade (computação)
Israel (Tel Aviv)	`il-central-1`	3
AWS GovCloud (US-West)	`us-gov-west-1`	3
AWS GovCloud (US-East)	`us-gov-east-1`	3

> **Note**
>
> A arquitetura do Amazon DocumentDB separa armazenamento e computação. Para a camada de armazenamento, o Amazon DocumentDB replica seis cópias dos seus dados em três zonas de AWS disponibilidade (AZs). As AZs listadas na tabela acima são o número de AZs que você pode usar em uma determinada região para provisionar instâncias de computação. Por exemplo, se você estiver executando um cluster do Amazon DocumentDB em ap-northeast-1, seu armazenamento será replicado de seis maneiras em três AZs, mas suas instâncias de computação só estarão disponíveis em duas AZs.

A tabela a seguir mostra o formato que deve ser usado para criar um ARN para um recurso específico do Amazon DocumentDB. O Amazon DocumentDB compartilha o formato de ARNs do Amazon RDS. Os ARNs do Amazon DocumentDB contêm `rds` e não `docdb`.

Tipo de recurso	Formato/exemplo de ARN
Instância (db)	`arn:aws:rds:` *`region`*`:`*`account_number`* `:db:`*`resource_id`* `arn:aws:rds:us-east-1:` *`1234567890`* `:db:`*`sample-db-instance`*
Cluster (`cluster`)	`arn:aws:rds:` *`region`*`:`*`account_number`* `:cluster:` *`resource_id`*

Tipo de recurso	Formato/exemplo de ARN
	`arn:aws:rds:us-east-1:` *1234567890* `:cluster:` *sample-db-cluster*
Grupo de parâmetros do cluster (`cluster-pg`)	`arn:aws:rds:` *region*`:`*account_number* `:cluster-pg:` *resource_id* `arn:aws:rds:us-east-1:` *1234567890* `:cluster-pg:` *sample-db-cluster-parameter-group*
Grupo de segurança (`secgrp`)	`arn:aws:rds:` *region*`:`*account_number* `:secgrp:`*resource_id* `arn:aws:rds:us-east-1:` *1234567890* `:secgrp:`*sample-public-secgrp*
Snapshot de cluster (`cluster-snapshot`)	`arn:aws:rds:` *region*`:`*account_number* `:cluster-snapshot:` *resource_id* `arn:aws:rds:us-east-1:` *1234567890* `:cluster-snapshot:` *sample-db-cluster-snapshot*
Grupo de sub-rede (`subgrp`)	`arn:aws:rds:` *region*`:`*account_number* `:subgrp:`*resource_id* `arn:aws:rds:us-east-1:` *1234567890* `:subgrp:`*sample-subnet-10*

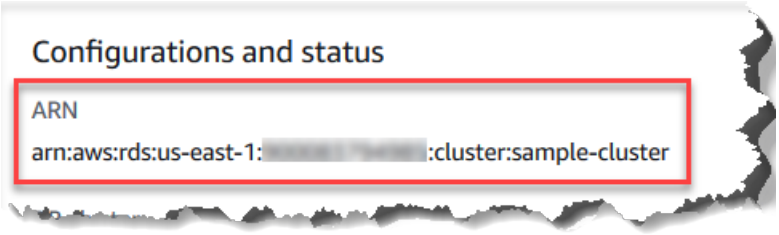
Localizar um ARN de recurso do Amazon DocumentDB

Você pode encontrar o ARN de um recurso do Amazon DocumentDB usando o Console de gerenciamento da AWS ou o. AWS CLI

Using the Console de gerenciamento da AWS

Para localizar um ARN usando o console, navegue até o recurso para o qual deseja um ARN e veja os detalhes desse recurso.

Por exemplo, é possível obter o ARN de um cluster selecionando a guia Configuração na página de detalhes do cluster. O ARN pode ser encontrado na seção Configurações e status, conforme mostrado na captura de tela a seguir.



Using the AWS CLI

Para obter um ARN usando o AWS CLI para um recurso específico do Amazon DocumentDB, use `describe` a operação para esse recurso. A tabela a seguir mostra cada AWS CLI operação e a propriedade ARN que é usada com a operação para obter um ARN.

AWS CLI Comando	Propriedade do 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- Localização do ARN de um cluster

A AWS CLI operação a seguir encontra o ARN do cluster. `sample-cluster`

Para Linux, macOS ou 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'
```

A saída dessa operação é semelhante ao seguinte (formato JSON).

```
[
    "arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster"
]
```

Example- Localização de ARNs para vários grupos de parâmetros

Para Linux, macOS ou Unix:

```
aws docdb describe-db-cluster-parameter-groups \
   --query 'DBClusterParameterGroups[*].DBClusterParameterGroupArn'
```

Para Windows:

```
aws docdb describe-db-cluster-parameter-groups ^
   --query 'DBClusterParameterGroups[*].DBClusterParameterGroupArn'
```

A saída dessa operação é semelhante ao seguinte (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"
]
```

Marcação de recursos do Amazon DocumentDB

Você pode usar tags do Amazon DocumentDB (compatível com MongoDB) para adicionar metadados aos recursos do Amazon DocumentDB. Essas tags podem ser usadas com políticas AWS Identity and Access Management (IAM) para gerenciar o acesso aos recursos do Amazon DocumentDB e controlar quais ações podem ser aplicadas aos recursos. Você também pode usar as tags para monitorar custos agrupando despesas de recursos marcados com tags semelhantes.

Você pode marcar os seguintes recursos do Amazon DocumentDB:

- Clusters
- Instâncias
- Snapshots
- Snapshots do cluster
- Grupos de parâmetros
- Grupos de parâmetros de clusters
- Grupos de segurança
- Grupos de sub-redes

Visão geral de tags de recurso do Amazon DocumentDB

Uma tag do Amazon DocumentDB é um par de nome/valor que você define e associa a um recurso do Amazon DocumentDB. O nome é referido como chave. Fornecer um valor para a chave é opcional. É possível usar tags para atribuir informações arbitrárias a um domínio do Amazon DocumentDB. É possível usar uma chave de tag, por exemplo, para definir uma categoria, e o valor da tag pode ser um item nessa categoria. Por exemplo, você pode definir uma chave de tag como `project` e um valor de tag de `Salix`, indicando que o recurso Amazon DocumentDB está atribuído ao projeto Salix. Você também pode usar tags para designar recursos do Amazon DocumentDB como usados para teste ou produção usando uma chave como `environment=test` ou `environment=production`. Recomendamos que você use um conjunto consistente de chaves de tag para facilitar o rastreamento de metadados associados aos recursos do Amazon DocumentDB.

Você pode usar etiquetas para organizar sua AWS fatura de forma a refletir sua própria estrutura de custos. Para fazer isso, inscreva-se para receber sua Conta da AWS fatura com os valores-chave da etiqueta incluídos. Então, para ver o custo de recursos combinados, organize suas informações

de faturamento de acordo com recursos com os mesmos valores de chave de tags. Por exemplo, é possível marcar vários recursos com um nome de aplicação específico, e depois organizar suas informações de faturamento para ver o custo total daquela aplicação em vários serviços. Para mais informações, consulte Using Cost Allocation Tags no AWS Guia do usuário do Billing and Cost Management.

Cada recurso do Amazon DocumentDB tem um conjunto de tags que contém todas as tags atribuídas a esse recurso. Um conjunto de tags pode conter até 10 tags ou estar vazio. Se você adicionar uma tag a um recurso do Amazon DocumentDB que tenha a mesma chave de uma tag existente no recurso, o novo valor substituirá o valor antigo.

AWS não aplica nenhum significado semântico às suas tags; as tags são interpretadas estritamente como cadeias de caracteres. O Amazon DocumentDB pode definir tags em uma instância de banco de dados ou outros recursos do Amazon DocumentDB, dependendo das configurações usadas ao criar o recurso. Por exemplo, o Amazon DocumentDB pode adicionar uma tag indicando que uma instância é para produção ou teste.

Você pode adicionar uma tag a um snapshot, mas sua conta não refletirá esse agrupamento.

Você pode usar o Console de gerenciamento da AWS ou o AWS CLI para adicionar, listar e excluir tags nos recursos do Amazon DocumentDB. Ao usar o AWS CLI, você deve fornecer o Amazon Resource Name (ARN) para o recurso com o qual deseja trabalhar. Para obter mais informações sobre ARNs do Amazon DocumentDB, consulte Compreender os nomes do recursos da Amazon (ARN) do Amazon DocumentDB.

Restrições de tag

As restrições a seguir se aplicam às tags do Amazon DocumentDB:

- Número máximo de tags por recurso - 10
- Comprimento máximo da chave - 128 caracteres Unicode
- Comprimento máximo de valor - 256 caracteres Unicode
- Caracteres válidos para chave e valor - letras maiúsculas e minúsculas no conjunto de UTF-8 caracteres, dígitos, espaço e os seguintes caracteres: `_ . : / = + -` e `@` (regex Java:) `"^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$"`
- As chaves e valores das tags diferenciam maiúsculas de minúsculas.
- O prefixo `aws:` não pode ser usado para chaves ou valores de tag; ele é reservado para a AWS.

Adicionar e atualizar tags em um recurso do Amazon DocumentDB

Você pode adicionar até 10 tags a um recurso usando o Console de gerenciamento da AWS ou AWS CLI o.

Using the Console de gerenciamento da AWS

O processo de adicionar uma tag a um recurso é semelhante, independentemente do recurso ao qual você está adicionando a tag. Neste exemplo, você adiciona uma tag a um cluster.

Para adicionar ou atualizar tags a um cluster usando o console

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em. https://console.aws.amazon.com/docdb
2. No painel de navegação, escolha clusters.
3. Escolha o nome do cluster ao qual você deseja adicionar tags.
4. Role para baixo até a seção Tags e depois escolha Editar.
5. Para cada tag que você deseja adicionar a este recurso, faça o seguinte:
   a. Para adicionar uma nova tag, insira o nome da tag na caixa Chave. Para alterar o valor de uma tag, localize o nome da tag na coluna Chave.
   b. Para atribuir um valor novo ou atualizado à tag, insira um valor para a tag na caixa Valor.
   c. Se você tiver mais tags para adicionar, escolha Adicionar. Caso contrário, quando terminar, escolha Salvar.

Using the AWS CLI

O processo de adicionar uma tag a um recurso é semelhante, independentemente do recurso ao qual você está adicionando as tags. Neste exemplo, você adiciona três tags a um cluster. A segunda tag, `key2`, não tem valor.

Use a AWS CLI operação `add-tags-to-resource` com esses parâmetros.

Parâmetros

- **`--resource-name`**—O ARN do recurso do Amazon DocumentDB ao qual você deseja adicionar as tags.

- **`--tags`**: uma lista das tags (par de chave/valor) que você deseja adicionar a esse recurso no formato `Key=key-name,Value=tag-value`.

Example

Para Linux, macOS ou 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
```

A operação `add-tags-to-resource` não produzirá uma saída. Para ver os resultados da operação, use a operação `list-tags-for-resource`.

Listar tags em um recurso do Amazon DocumentDB

Você pode usar o Console de gerenciamento da AWS ou o AWS CLI para obter uma lista das tags de um recurso do Amazon DocumentDB.

Using the Console de gerenciamento da AWS

O processo de listar tags em um recurso é semelhante, independentemente do recurso ao qual você está adicionando a tag. Neste exemplo, você lista as tags para um cluster.

Para listar as tags em um cluster usando o console

1. Abra o console do Amazon DocumentDB em. https://console.aws.amazon.com/docdb
2. No painel de navegação, escolha clusters.
3. Escolha o nome do cluster para o qual você deseja listar tags.
4. Para ver uma lista das tags nesse recurso, role para baixo até a seção Tags.

Using the AWS CLI

O processo de listar as tags em um recurso é semelhante, independentemente do recurso para o qual você está listando a tag. Neste exemplo, você lista as tags em um cluster.

Use a AWS CLI operação `list-tags-for-resource` com esses parâmetros.

Parâmetros

- **`--resource-name`**—Obrigatório. O ARN do recurso do Amazon DocumentDB para o qual você deseja listar as tags para.

Example

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante ao seguinte (formato JSON).

```
{
    "TagList": [
        {
            "Key": "key1",
            "Value": "value1"
        },
        {
            "Key": "key2",
            "Value": ""
        },
        {
            "Key": "key3",
            "Value": "value3"
        }
```

```
    ]
}
```

Remover tags de um recurso do Amazon DocumentDB

Você pode usar o Console de gerenciamento da AWS ou o AWS CLI para remover tags dos recursos do Amazon DocumentDB.

Using the Console de gerenciamento da AWS

O processo de remover tags de um recurso é semelhante, independentemente do recurso do qual você está removendo a tag. Neste exemplo, você remove as tags de um cluster.

Para remover tags de um cluster usando o console

1. Abra o console do Amazon DocumentDB em. https://console.aws.amazon.com/docdb
2. No painel de navegação, escolha clusters.
3. Escolha o nome do cluster do qual você deseja remover as tags.
4. Role para baixo até a seção Tags e depois escolha Edit (Editar).
5. Se você deseja remover todas as tags desse recurso, escolha Remover todas. Caso contrário, para cada tag que você deseja remover deste recurso, faça o seguinte:
    a. Localize o nome da tag na coluna Chave.
    b. Escolha Remover na mesma linha que a chave de tag.
    c. Quando terminar, escolha Salvar.

Using the AWS CLI

O processo de remover uma tag de um recurso é semelhante, independentemente do recurso do qual você está removendo a tag. Neste exemplo, você remove a tag de um cluster.

Use a AWS CLI operação `remove-tags-from-resource` com esses parâmetros.

- **`--resource-name`**—Obrigatório. O ARN do recurso do Amazon DocumentDB do qual você deseja remover as tags.
- **`--tag-keys`**—Obrigatório. Uma lista das chaves de tag que você deseja remover deste recurso.

Example

Para Linux, macOS ou 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
```

A operação `removed-tags-from-resource` não produzirá uma saída. Para ver os resultados da operação, use a operação `list-tags-for-resource`.

Manutenção do Amazon DocumentDB

O Amazon DocumentDB executa periodicamente dois tipos de manutenção:

- A manutenção do cluster atualiza o mecanismo do banco de dados. As atualizações do mecanismo incluem correções de segurança, correções de erros, novos recursos e outros aprimoramentos do mecanismo.
- A manutenção da instância atualiza o sistema operacional (SO) na instância.

Os patches de mecanismo e as atualizações do sistema operacional usam as mesmas três categorias de ciclo de vida — opcional , obrigatório e forçado — com a mesma notificação e comportamento de aplicação para cada categoria. As versões do motor também têm uma quarta categoria: versões secundárias, para as quais você atualiza manualmente. As categorias são:

- Opcional — contém melhorias não críticas. Sem data de aplicação automática e sem notificação de AHD; inscreva-se quando quiser. (Para atualizações do sistema operacional, você pode se inscrever `RDS-EVENT-0230` para ser notificado quando uma estiver disponível.)
- Obrigatório — contém segurança e outras correções críticas. Você recebe uma notificação por meio do Health Dashboard (AHD) e e-mail. Uma ação necessária se aplica automaticamente durante a janela de manutenção do cluster ou da instância após a`AutoAppliedAfterDate`. Você pode adiar alterando a janela de manutenção antes dessa data.

- Forçado — uma correção rara e altamente crítica. Auto-applies fora de sua janela de manutenção após sua`ForcedApplyDate`. O Amazon DocumentDB só designa uma ação forçada quando nenhuma outra opção está disponível.
- Versão secundária (somente versões do mecanismo) — uma versão numerada do mecanismo sobre uma versão principal (por exemplo,`5.0.1`). User-driven: você atualiza modificando a versão do mecanismo do cluster. Nunca se aplica automaticamente; sem notificação de AHD. As versões secundárias não são publicadas para as versões principais anteriores à 5.0.

Os patches do motor são lançados em uma única categoria (opcional, obrigatória ou forçada) e permanecem lá. Progresso das atualizações do sistema operacional: a maioria começa como opcional e, se não for aplicada, passa para necessária e, eventualmente, forçada. O tempo exato depende do patch e é publicado nos campos de notificação e data do AHD retornados por `describe-pending-maintenance-actions` (consulteAplicar datas). As notas de lançamento do Amazon DocumentDB usam esses nomes de categorias ao anunciar mudanças no mecanismo.

A aplicação de qualquer patch de mecanismo deixa o cluster off-line brevemente. O restante deste tópico mostra como as janelas de manutenção funcionam, como encontrar trabalhos pendentes, como aplicar patches de mecanismo e versões secundárias, como funcionam as atualizações do sistema operacional e como lidar com clusters globais.

Tópicos

- Ações de manutenção para o Amazon DocumentDB
- Numeração da versão do motor
- Gerenciar suas janelas de manutenção do Amazon DocumentDB
- Notificações para patches do mecanismo do Amazon DocumentDB
- Consultar ações de manutenção pendentes do Amazon DocumentDB
- Atualizações do mecanismo do Amazon DocumentDB
- Atualizações de versões secundárias
- Atualizações do sistema operacional do Amazon DocumentDB
- User-initiated atualizações
- Aplicação de patches em clusters globais

Ações de manutenção para o Amazon DocumentDB

As seguintes ações de manutenção se aplicam aos clusters do Amazon DocumentDB:

- `system-update`— Atualize o patch do mecanismo para o cluster Amazon DocumentDB. Para obter mais informações, consulte Atualizações do mecanismo do Amazon DocumentDB.
- `os-upgrade`— Atualize os sistemas operacionais de todas as instâncias de banco de dados no cluster Amazon DocumentDB usando atualizações contínuas. Para obter mais informações, consulte Atualizações do sistema operacional do Amazon DocumentDB.

As seguintes ações de manutenção se aplicam às instâncias do Amazon DocumentDB:

- `system-update`— Atualize o sistema operacional da instância Amazon DocumentDB. Em vez disso, recomendamos que você use a ação de `os-upgrade` manutenção em nível de cluster. Para obter mais informações, consulte Atualizações do sistema operacional do Amazon DocumentDB.

Numeração da versão do motor

O Amazon DocumentDB usa dois identificadores de versão separados:

- Versão do motor — um número de três partes no formato *`major`*`.`*`major`*`.`*`minor`* (por exemplo, `5.0.0` ou`5.0.1`). As duas primeiras partes (`5.0`) são a versão de compatibilidade com o MongoDB; a terceira parte é a versão secundária, incrementada quando o Amazon DocumentDB publica uma versão secundária contendo correções de erros e melhorias ininterruptas. Essa é a versão que você especifica ao criar ou atualizar um cluster.
- Versão do patch do mecanismo — um número separado de três partes no formato *`major`*`.0.`*`patch`* (por exemplo,`3.0.17983`) que identifica o nível de patch aplicado ao seu cluster. O dígito do meio é sempre`0`. As versões de patch contêm correções críticas de segurança e estabilidade.

Quando a versão do patch do mecanismo aparece sozinha, você pode determinar a versão do mecanismo a partir de seu prefixo, conforme mostrado na tabela a seguir.

Prefixo da versão do patch do motor	Versão do mecanismo Amazon DocumentDB
`1.0.`*`x`*	3.6
`2.0.`*`x`*	4,0

Prefixo da versão do patch do motor	Versão do mecanismo Amazon DocumentDB
`3.0.x`	5,0
`4.0.x`	8.0

Para verificar a versão do patch que seu cluster está executando, conecte-se e executedb.runCommand({getEngineVersion: 1}).

> **Note**
>
> Começando com o Amazon DocumentDB 5.0.2 e 8.0.2, db.runCommand({getEngineVersion: 1}) retorna a versão do mecanismo e a versão do patch do mecanismo como uma única string no formulário. *engine-version*+*engine-patch-version* Por exemplo, um cluster no Amazon DocumentDB 5.0.2 retorna 5.0.2+3.0.12345 e um cluster no 8.0.2 retorna. 8.0.2+4.0.12345 Os clusters em versões anteriores retornam a versão do patch do mecanismo sozinha, como3.0.12345. Se você tiver scripts ou monitoramento que analisam a saída dogetEngineVersion, atualize-os para lidar com o formato combinado.

Para ver a lista de versões lançadas de patches de motor e o que cada uma contém, consulteNotas da versão.

Gerenciar suas janelas de manutenção do Amazon DocumentDB

Cada cluster e cada instância têm sua própria janela de manutenção semanal de 30 minutos — o período em que as modificações programadas e os patches de software são executados. A maioria dos eventos é concluída em 30 minutos; os maiores podem durar mais tempo.

Se você não escolher uma janela ao criar o recurso, o Amazon DocumentDB atribui uma aleatoriamente em um bloco diário de 8 horas definido para a região, em um dia selecionado aleatoriamente. Escolha janelas que minimizem o impacto em seu aplicativo — à noite ou nos fins de semana, por exemplo.

Para atualizações do mecanismo de banco de dados, o Amazon DocumentDB usa a janela do cluster, não as janelas de instâncias individuais.

A tabela a seguir mostra os blocos de tempo padrão por região.

Nome da região	Região	Bloco de tempo UTC
Leste dos EUA (Ohio)	us-east-2	03:00-11:00
Leste dos EUA (Norte da Virgínia)	us-east-1	03:00-11:00
Oeste dos EUA (Oregon)	us-west-2	06:00-14:00
África (Cidade do Cabo)	af-south-1	03:00-11:00
Ásia-Pacífico (Hong Kong)	ap-east-1	06:00-14:00
Ásia-Pacífico (Hyderabad)	ap-south-2	06:30–14:30
Ásia-Pacífico (Malásia)	ap-southeast-5	13:00-21:00
Ásia-Pacífico (Mumbai)	ap-south-1	06:00-14:00
Ásia-Pacífico (Osaka)	ap-northeast-3	12:00-20:00
Ásia-Pacífico (Seul)	ap-northeast-2	13:00-21:00
Ásia-Pacífico (Singapura)	ap-southeast-1	14:00-22:00
Ásia-Pacífico (Sydney)	ap-southeast-2	12:00-20:00
Ásia-Pacífico (Jacarta)	ap-southeast-3	08:00-16:00
Ásia-Pacífico (Melbourne)	ap-southeast-4	11:00-19:00
Ásia-Pacífico (Tailândia)	ap-southeast-7	15:00-23:00
Ásia-Pacífico (Tóquio)	ap-northeast-1	13:00-21:00
Canadá (Central)	ca-central-1	03:00-11:00
Oeste do Canadá (Calgary)	ca-west-1	18:00-02:00
China (Pequim)	cn-north-1	06:00-14:00
China (Ningxia)	cn-northwest-1	06:00-14:00

Nome da região	Região	Bloco de tempo UTC
Europa (Frankfurt)	eu-central-1	21:00-05:00
Europa (Zurique)	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 (Milão)	eu-south-1	02:00-10:00
Europa (Paris)	eu-west-3	23:59-07:29
Europa (Espanha)	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
Oriente Médio (Emirados Árabes Unidos)	me-central-1	05:00-13:00
América do Sul (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

Gerenciar suas janelas de manutenção do Amazon DocumentDB

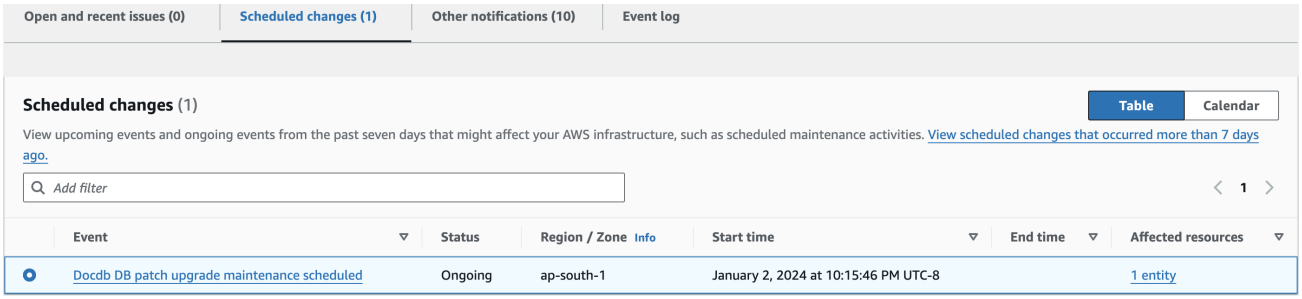
Escolha a janela de menor tráfego possível e ajuste-a com o tempo à medida que seus padrões de tráfego mudam. O cluster ou a instância não estará disponível durante a janela somente se uma alteração no sistema — uma operação de armazenamento em escala ou uma alteração na classe da instância, por exemplo — exigir uma interrupção, e somente pelo tempo que essa alteração realmente precisar.

Para alterar a janela de manutenção

- Para um cluster: consulte Modificar um cluster do Amazon DocumentDB.
- Para uma instância: consulte Modificar uma instância do Amazon DocumentDB.

Notificações para patches do mecanismo do Amazon DocumentDB

Quando um patch de mecanismo necessário é disponibilizado em uma AWS região, cada AWS conta com um cluster Amazon DocumentDB afetado nessa região recebe uma notificação por meio do Health Dashboard (AHD) e por e-mail (enviada para o endereço do usuário raiz da AWS conta). Uma notificação é entregue por versão afetada do mecanismo Amazon DocumentDB. Você pode encontrá-los em Mudanças programadas no AHD. Cada notificação lista o tempo de disponibilidade do patch, o cronograma de aplicação automática, os clusters afetados e as notas de lançamento.



Os patches de motor necessários seguem um único prazo de entrega de aproximadamente 30 dias. Quando um patch é disponibilizado em sua região, o Amazon DocumentDB envia a notificação descrita acima. Nesse ponto, o patch `AutoAppliedAfterDate` está configurado para aproximadamente 30 dias depois. Até essa data, o patch permanece pendente: você pode aplicá-lo a qualquer momento ou adiá-lo movendo a janela de manutenção do cluster para um dia posterior. Em ou após o`AutoAppliedAfterDate`, o patch se aplica automaticamente durante a próxima janela de manutenção do cluster.

Por exemplo, um patch obrigatório que se torna disponível em 1º de junho de 2026 tem um `AutoAppliedAfterDate` de aproximadamente 1º de julho de 2026. Você recebe a notificação em 1º de junho de 2026 e, se não fizer nada, o patch será aplicado automaticamente durante a primeira janela de manutenção do cluster, em ou após 1º de julho de 2026.

Você tem duas opções depois de receber a notificação: aplicar o patch automaticamente antes da data de aplicação automática ou esperar que ele seja aplicado automaticamente durante uma próxima janela de manutenção (o padrão). Para se inscrever automaticamente, abra a guia Manutenção e backups do cluster e procure a entrada do tipo`system-update`.

> **Note**
>
> O status da notificação no AHD permanece em andamento até que o Amazon DocumentDB lance outro patch de mecanismo com uma nova versão de patch. Depois que o patch é aplicado, a versão do patch do mecanismo do cluster é atualizada para corresponder à versão na notificação. Verifique a nova versão executando`db.runCommand({getEngineVersion: 1})`.

Patches opcionais e novas versões secundárias não geram notificações por e-mail ou AHD. Para rastreá-los, assista às notas de lançamento do Amazon DocumentDB.

Patches forçados (a categoria mais rara, reservada para as correções de segurança mais críticas) também são anunciados por meio de AHD e e-mail. Diferentemente dos patches necessários, eles se aplicam fora da janela de manutenção, portanto, o exemplo de tempo de aplicação automática acima não se aplica.

Reagindo às notificações de patch de forma programática

AWS Health integra-se à Amazon EventBridge, que permite criar aplicativos orientados por eventos em mais de 20 destinos, incluindo AWS Lambda o Amazon Simple Queue Service (SQS). Para reagir programaticamente à disponibilidade do patch do motor, configure de acordo com o evento. EventBridge `AWS_DOCDB_DB_PATCH_UPGRADE_MAINTENANCE_SCHEDULED` A partir daí, você pode capturar dados de eventos, gerar eventos adicionais, enviar notificações push por meio do AWS Console Mobile Application ou realizar qualquer outra ação necessária.

Se o Amazon DocumentDB cancelar um patch (raro), você receberá uma notificação do AHD e um e-mail sobre o cancelamento. Use o código do `AWS_DOCDB_DB_PATCH_UPGRADE_MAINTENANCE_CANCELLED` evento com EventBridge a Amazon para lidar com esse caso. Para saber mais sobre como escrever regras, consulte o Guia EventBridge do usuário da Amazon.

Consultar ações de manutenção pendentes do Amazon DocumentDB

Use o Console de gerenciamento da AWS ou o AWS CLI para verificar qual manutenção está pendente para um cluster ou instância.

As atualizações pendentes aparecem com o tipo de ação`system-update`, que abrange tanto os patches do mecanismo quanto as atualizações do sistema operacional.

Quando uma atualização está pendente, você pode:

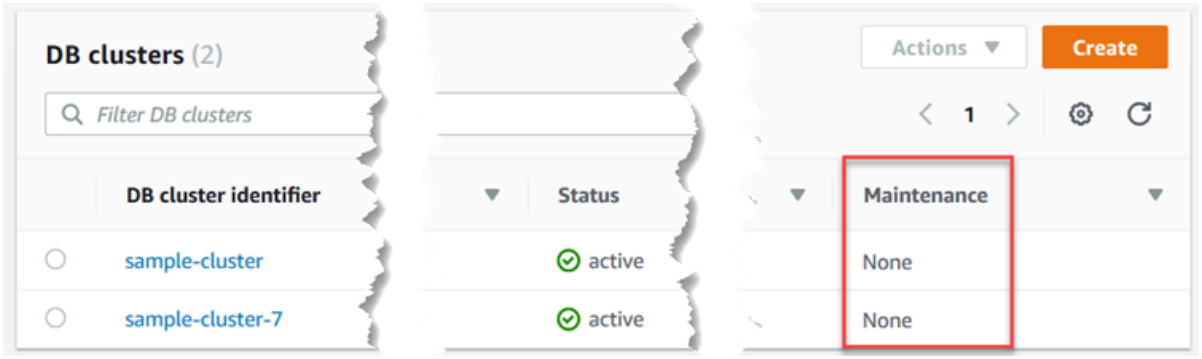
- Aplique imediatamente.
- Agende-o para a próxima janela de manutenção.
- Adie-o (somente patches de motor e atualizações do sistema operacional) alterando sua janela de manutenção antes`AutoAppliedAfterDate`. Depois que essa data passar, a ação será aplicada automaticamente durante a próxima janela de manutenção. Uma vez `ForcedApplyDate` aprovado, nenhum adiamento adicional é possível.

> **Note**
>
> Se você não fizer nada, as ações de manutenção necessárias, como os patches de motor necessários, serão aplicadas automaticamente durante uma próxima janela de manutenção. Patches opcionais e versões secundárias nunca se aplicam automaticamente.

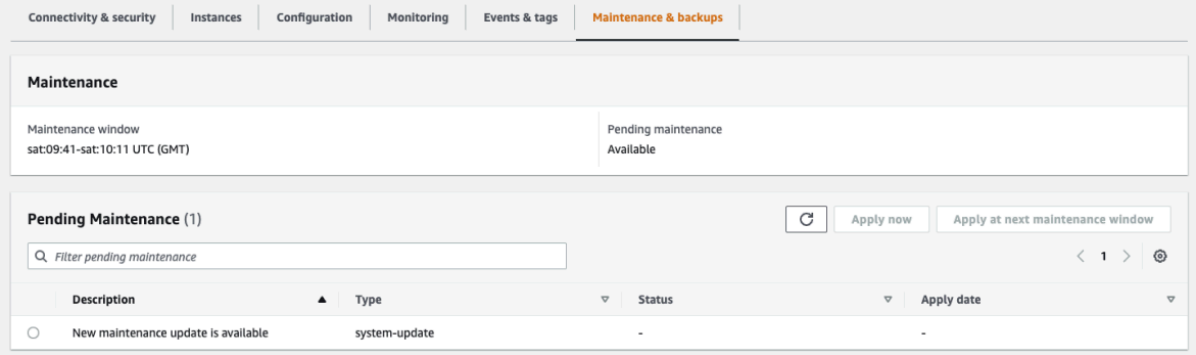
A janela de manutenção controla quando as operações pendentes começam, não quanto tempo elas levam para serem concluídas.

Using the Console de gerenciamento da AWS

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Clusters.
3. A coluna Manutenção do cluster mostra a janela Disponível , Obrigatória ou Próxima quando uma atualização está pendente.



4. Abra o cluster e escolha Manutenção e backups para ver os itens de manutenção pendente e agir sobre eles.



Using the AWS CLI

Corra `describe-pending-maintenance-actions` para ver o que está pendente. O exemplo a seguir mostra uma conta sem ações pendentes.

```
aws docdb describe-pending-maintenance-actions
```

A saída dessa operação é semelhante ao seguinte (formato JSON).

```
{
    "PendingMaintenanceActions": []
}
```

Uma conta com uma ação pendente retorna uma saída parecida com esta:

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

Você pode definir o escopo da listagem para clusters específicos com`--filters`, no formulárioName=*filter-name*,Values=*resource-id*,.... O filtro aceito Name édb-cluster-id, que usa uma lista de identificadores de cluster ou ARNs.

Example

Para Linux, macOS ou 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
```

Aplicar datas

Cada ação de manutenção pendente tem até três datas de aplicação. Eles aparecem na AWS CLI saída de `describe-pending-maintenance-actions` e indicam quando a ação será executada. Os campos são `null` para manutenção opcional.

- **`CurrentApplyDate`**—quando a ação está programada para ser executada, agora ou na próxima janela de manutenção. Preenchido para ações obrigatórias e forçadas.
- **`AutoAppliedAfterDate`**—a data após a qual a aplicação automática começa durante a janela de manutenção do cluster ou da instância. Preenchido para as ações necessárias.
- **`ForcedApplyDate`**—o prazo rígido. Após essa data, a ação é executada automaticamente, independentemente da janela de manutenção. Preenchido para ações forçadas.

Para adiar uma ação pendente, mova sua janela de manutenção para um dia posterior. `AutoAppliedAfterDate` Depois de `AutoAppliedAfterDate` aprovada, a ação será aplicada automaticamente durante a próxima janela de manutenção. Uma vez `ForcedApplyDate` aprovado, nenhum adiamento adicional é possível. A janela exata de adiamento varia de acordo com o patch; as datas são publicadas na notificação do AHD e na AWS CLI saída.

Atualizações do mecanismo do Amazon DocumentDB

Depois de identificar um patch de motor pendente, use um dos procedimentos a seguir para aplicá-lo ou programá-lo. Você pode executar esses procedimentos a partir do Console de gerenciamento da AWS ou do AWS CLI.

Using the Console de gerenciamento da AWS

Para gerenciar a atualização de um cluster

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Clusters.
3. Selecione o cluster que você deseja atualizar.
4. No menu Ações, escolha uma das seguintes opções:
   - Atualize agora — execute a manutenção pendente imediatamente.
   - Atualize na próxima janela — execute-o durante a próxima janela de manutenção do cluster.

   Você também pode usar Aplicar agora ou Aplicar na próxima janela de manutenção na seção Manutenção pendente da guia Manutenção e backups do cluster (consulteConsultar ações de manutenção pendentes do Amazon DocumentDB).

   > **Note**
   >
   > Se nada estiver pendente, todas essas opções estarão inativas.

Using the AWS CLI

Aplique uma atualização pendente com`apply-pending-maintenance-action`.

Parâmetros

- **`--resource-identifier`**—Amazon DocumentDB Amazon Resource Name (ARN) do recurso que a ação pendente visa.

- **`--apply-action`**—a ação de manutenção pendente a ser aplicada. Use `system-update` para aplicar um patch de motor.
- **`--opt-in-type`**—o tipo de solicitação de aceitação ou se deseja desfazer uma. Valores válidos:
  - `immediate`—inscreva-se agora. Não pode ser desfeito depois de enviado.
  - `next-maintenance`—aplique durante a próxima janela de manutenção do recurso.
  - `undo-opt-in`—cancelar um `next-maintenance` opt-in existente.

Example

Para Linux, macOS ou 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
```

Leia a disponibilidade durante a aplicação de patches

Os mecanismos 5.0 e 8.0 do Amazon DocumentDB preservam a disponibilidade de leitura durante a aplicação de patches quando o cluster tem várias instâncias. O Amazon DocumentDB corrige instâncias de leitores de forma contínua, em três grupos, para que os leitores restantes continuem fornecendo tráfego. O escritor está brevemente indisponível enquanto faz o patch. Para atingir zero tempo de inatividade de leitura, defina sua preferência de leitura para que as leituras possam voltar para o redator: `secondaryPreferred` ou `primaryPreferred` trabalhar; `primary` ou `secondary` sozinhas, podem causar inatividade de leitura.

Modo de preferência de leitura	Durante a atualização do escritor	Durante a atualização do leitor	Número mínimo de leitores necessários para zero tempo de inatividade de leitura
`primary`	Read/write tempo de inatividade	Sem impacto	N/A
`primaryPreferred`	Anote o tempo de inatividade	Sem impacto	1
`secondary`	Anote o tempo de inatividade	Tempo de inatividade de leitura (se for apenas um leitor)	2
`secondaryPreferred`	Anote o tempo de inatividade	Sem impacto	1
`nearest`	Anote o tempo de inatividade	Sem impacto	1

Enquanto os leitores estão corrigindo, a taxa de transferência geral de leitura do cluster diminui temporariamente. Para manter a taxa de transferência estável, provisione leitores adicionais antes da atualização e remova-os após sua conclusão.

Nos mecanismos 3.6 e 4.0, esses recursos de disponibilidade de leitura não se aplicam: um patch do mecanismo causa maior tempo de inatividade que afeta tanto as leituras quanto as gravações. Para atualizar para uma versão principal que funcione, consulteAtualização da versão principal implementada do Amazon DocumentDB no local.

Duração do tempo de inatividade do patch

Engine-patch o tempo de inatividade varia. Os maiores fatores são a utilização da CPU e a pressão de memória na instância no momento do patch, portanto, dimensionar corretamente suas instâncias é importante. Para minimizar o tempo de inatividade, execute a versão mais recente do mecanismo principal do Amazon DocumentDB e distribua as instâncias em várias zonas de disponibilidade.

Atualizações e substituições de patches

O Amazon DocumentDB monitora os patches após o lançamento. No caso raro de um problema ser identificado, o Amazon DocumentDB pausa o lançamento enquanto prepara uma versão atualizada. Quando isso acontece, os clusters que ainda não receberam o patch não o veem mais como uma ação de manutenção disponível e a notificação de alteração programada correspondente no Health Dashboard é retirada. Os clusters que já executam a versão afetada continuam operando normalmente e não exigem nenhuma ação de sua parte.

Um patch atualizado será lançado em breve. Quando ele estiver disponível em sua região, você receberá uma nova notificação por e-mail, conforme descrito emNotificações para patches do mecanismo do Amazon DocumentDB. Health Dashboard

Atualizações de versões secundárias

O Amazon DocumentDB publica versões secundárias além da versão principal 5.0 e posterior (por exemplo,`5.0.1`). As versões secundárias não são publicadas para as versões principais anteriores à 5.0. As versões secundárias se comportam de maneira diferente dos patches de motor obrigatórios e opcionais:

- Eles não aparecem como uma ação de manutenção pendente e nunca são aplicados automaticamente.
- Eles não geram notificações por AHD ou por e-mail. Novas versões secundárias são anunciadas nas notas de lançamento do Amazon DocumentDB.
- Para atualizar, você modifica a versão do mecanismo do cluster (imediatamente ou durante a próxima janela de manutenção). Os upgrades de versões secundárias exigem um breve tempo de inatividade e são unidirecionais — você não pode fazer o downgrade para uma versão secundária anterior. Para clusters globais, atualize os clusters secundários antes dos primários.

Leia mais:Atualização da versão secundária do Amazon DocumentDB.

Atualizações do sistema operacional do Amazon DocumentDB

Às vezes, as instâncias precisam de atualizações do sistema operacional. O Amazon DocumentDB atualiza o sistema operacional para melhorar o desempenho e reforçar a segurança. As atualizações do sistema operacional deixam a versão do mecanismo de cluster e a classe da instância inalteradas. Assim como os patches do mecanismo, as atualizações do sistema operacional usam

o ciclo de vida opcional/exigido/forçado descrito no início deste tópico; diferentemente dos patches do mecanismo, uma atualização do sistema operacional pode passar por essas categorias ao longo do tempo se você adiá-la. Aplique as atualizações do sistema operacional assim que elas estiverem disponíveis e defina as janelas de manutenção de clusters e instâncias para horários que atendam às suas necessidades comerciais.

Use a ação de `os-upgrade` manutenção em nível de cluster para aplicar atualizações do sistema operacional em todas as instâncias em um cluster. O Amazon DocumentDB atualiza as instâncias de forma contínua, algumas de cada vez, e atualiza a instância primária por último para minimizar os failovers. A atualização é executada durante a janela de manutenção do cluster, não a janela de manutenção de instâncias individuais, que você configura.

Depois que uma instância recebe uma atualização do sistema operacional, seu cache de buffer começa vazio. Até que o conjunto de trabalho seja repreenchido a partir do volume de armazenamento, as consultas nessa instância podem ter uma latência maior e menor. `BufferCacheHitRatio`

Quando o Amazon DocumentDB atualiza a instância primária, um failover promove uma réplica como a nova primária. Use o endpoint do cluster para que seu aplicativo processe isso de forma transparente. Para manter a disponibilidade de leitura enquanto as instâncias estão sendo atualizadas, defina sua preferência `secondaryPreferred` de leitura para que as leituras `primaryPreferred` possam voltar para uma instância disponível. Mantenha os possíveis alvos de failover (réplicas com o nível de maior prioridade) na mesma classe de instância da primária. Isso evita a degradação do desempenho de gravação após a promoção. Para obter detalhes, consulte Failover do Amazon DocumentDB.

As ações em nível de cluster `os-upgrade` e de instância podem aparecer simultaneamente como `system-update` ações disponíveis. `describe-pending-maintenance-actions` No entanto, você não pode agendar os dois ao mesmo tempo. Se `system-update` as ações no nível da instância estiverem ativamente agendadas em qualquer instância, você deverá cancelá-las ou concluí-las antes de programar a ação no nível do cluster `os-upgrade` e vice-versa.

> **Important**
>
> Sua instância do Amazon DocumentDB fica off-line para a atualização do sistema operacional. Multi-instance os clusters minimizam o impacto. Se você executar um cluster de instância única, poderá adicionar temporariamente um secundário para a atualização e removê-lo posteriormente. O secundário incorre nas cobranças usuais enquanto existe.

> **Note**
>
> A `system-update` ação no nível da instância ainda está disponível para compatibilidade com versões anteriores. Se você precisar usá-lo, atualize as réplicas primeiro e a principal por último — evite corrigi-las simultaneamente, pois um failover durante o patch pode prolongar o tempo de inatividade.

Para receber um evento quando uma nova atualização opcional do sistema operacional chegar, inscreva-se `RDS-EVENT-0230` na categoria de eventos de patches de segurança. Para obter mais informações, consulte Tornar-se assinante de eventos do Amazon DocumentDB.

> **Note**
>
> Manter-se atualizado sobre as atualizações opcionais e obrigatórias pode ser necessário para fins de conformidade. Aplique `os-upgrade` ações rotineiramente durante suas janelas de manutenção.

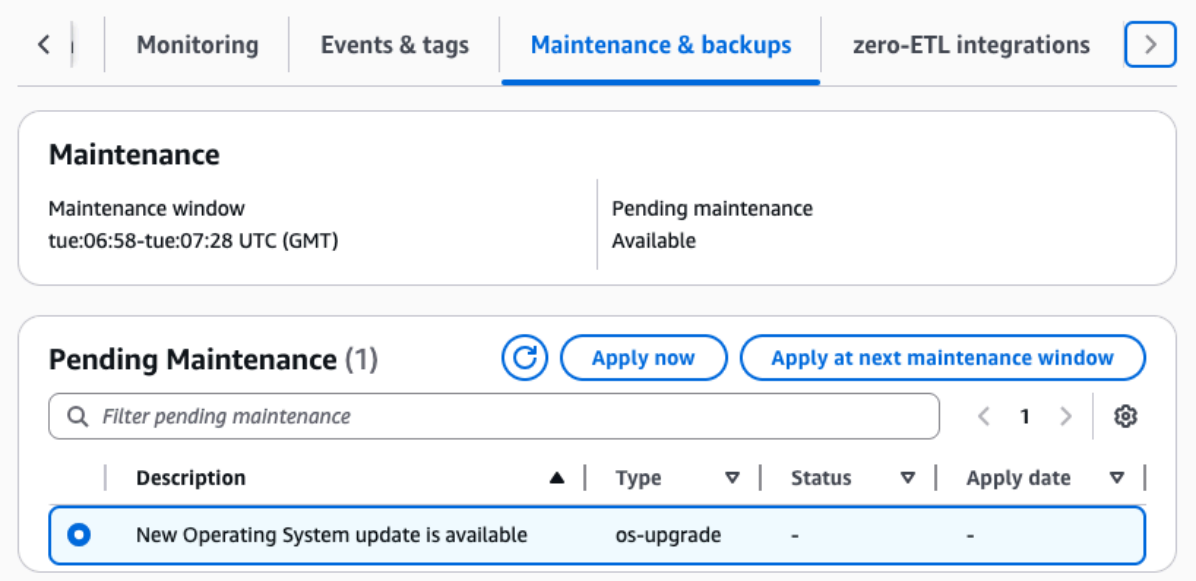
As atualizações do sistema operacional estão vinculadas a classes de instâncias específicas, portanto, instâncias diferentes se tornam elegíveis em momentos diferentes. Se seu cluster não estiver no patch de mecanismo mais recente, a atualização do sistema operacional pode não aparecer — aplique primeiro o patch de mecanismo mais recente (consulteAtualizações do mecanismo do Amazon DocumentDB).

Use o Console de gerenciamento da AWS ou AWS CLI para verificar se uma atualização está disponível.

Using the Console de gerenciamento da AWS

Para verificar se há uma atualização do sistema operacional no console:

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Clusters e selecione o nome do cluster.
3. Escolha a guia Manutenção e backups.
4. Em Manutenção pendente, a `os-upgrade` ação aparece se uma atualização do sistema operacional estiver disponível.



5. Selecione a `os-upgrade` ação e escolha Aplicar agora ou Aplicar na próxima janela de manutenção. Se o valor for a próxima janela, você poderá adiar com Adiar a atualização, desde que a ação não tenha sido iniciada.

Using the AWS CLI

Verifique se há uma atualização pendente do sistema operacional:

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

As atualizações do sistema operacional aparecem no nível do cluster como `os-upgrade` e no nível da instância como`system-update`. Use a ação em nível de cluster`os-upgrade`.

Example

O exemplo a seguir aplica a atualização do sistema operacional imediatamente.

Para Linux, macOS ou 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 atualizações

Algumas mudanças você mesmo inicia, por exemplo, trocando uma classe de instância por uma com mais ou menos memória ou alterando o grupo de parâmetros do cluster. O Amazon DocumentDB trata essas atualizações de forma diferente das atualizações que ele inicia. Para obter detalhes, consulte:

- Modificar um cluster do Amazon DocumentDB
- Modificar uma instância do Amazon DocumentDB

Para listar as alterações iniciadas pelo usuário que ainda estão pendentes:

Example

Para listar as alterações pendentes iniciadas pelo usuário em suas instâncias

Para Linux, macOS ou Unix:

```
aws docdb describe-db-instances \
    --query 'DBInstances[*].
[DBClusterIdentifier,DBInstanceIdentifier,PendingModifiedValues]'
```

Para Windows:

```
aws docdb describe-db-instances ^
    --query 'DBInstances[*].
[DBClusterIdentifier,DBInstanceIdentifier,PendingModifiedValues]'
```

A saída dessa operação é semelhante ao seguinte (formato JSON).

Neste exemplo, `sample-cluster-instance` tem uma alteração pendente em`db.r5.xlarge`; não `sample-cluster-instance-2` tem nenhuma.

```
[
    [
        "sample-cluster",
        "sample-cluster-instance",
        {
            "DBInstanceClass": "db.r5.xlarge"
        }
    ],
```

```
    [
        "sample-cluster",
        "sample-cluster-instance-2",
        {}
    ]
]
```

Aplicação de patches em clusters globais

Em um cluster global, cada cluster membro — primário e secundário — é atualizado durante sua própria janela de manutenção. Quando um patch de motor necessário está disponível em todas as regiões, você recebe uma notificação por AHD e por e-mail. Patches opcionais e novas versões secundárias não geram notificações; verifique as notas de lançamento do Amazon DocumentDB para ver essas notificações.

Se você se inscrever automaticamente, sempre corrija os secundários primeiro e os primários por último. Esse pedido mantém o failover e o switchover disponíveis durante todo o lançamento.

> **Important**
>
> Se você corrigir o primário primeiro por engano, coloque todos os secundários na mesma versão o mais rápido possível. O failover e o switchover permanecem desativados até que cada cluster esteja na mesma versão.

Se você não fizer nada, o patch se aplicará automaticamente durante a próxima janela de manutenção de cada cluster: primeiro as secundárias, depois as primárias em sua janela quando as secundárias estiverem concluídas.

Mantenha os clusters de banco de dados primário e secundário na mesma versão. O failover gerenciado entre regiões só funciona em um banco de dados global quando cada cluster compartilha a mesma versão de mecanismo e nível de patch. O mesmo se aplica se você adicionar um novo secundário que usa uma versão de mecanismo mais nova do que a primária — crie novos secundários na versão primária antes de juntá-los ao banco de dados global.

Depois de uma notificação de patch, atualize a primária e a secundária para a versão mais recente na primeira oportunidade de manter o failover e o switchover funcionando. Se uma solicitação de failover ou switchover for rejeitada, compare as versões do patch do mecanismo entre os clusters; se elas não corresponderem, aplique o patch disponível nos clusters atrasados.

Noções básicas dos perfis vinculados ao serviço

O Amazon DocumentDB (com compatibilidade com o MongoDB) usa funções vinculadas a AWS Identity and Access Management serviços (IAM). Uma função vinculada a serviços é um tipo exclusivo de função do IAM vinculada diretamente ao Amazon DocumentDB. Service-linked as funções são predefinidas pelo Amazon DocumentDB e incluem todas as permissões que o serviço exige para chamar outros AWS serviços em seu nome.

Um perfil vinculado ao serviço facilita a configuração do Amazon DocumentDB porque você não precisa adicionar as permissões necessárias manualmente. O Amazon DocumentDB define as permissões dos perfis vinculados ao serviço e, a não ser que esteja definido de outra forma, somente o Amazon DocumentDB poderá assumir os perfis. As permissões definidas incluem a política de confiança e a política de permissões, que não pode ser anexada a nenhuma outra entidade do IAM.

Você pode excluir os perfis somente depois de primeiro excluir seus recursos relacionados. Isso protege seus recursos do Amazon DocumentDB, pois você não pode remover por engano as permissões para acessar os recursos.

Para ter informações sobre outros serviços compatíveis com perfis vinculados ao serviço, consulte Serviços da AWS que funcionam com o IAM e procure os serviços que tiverem Sim na coluna Service-Linked Funções. Escolha um Sim com um link para visualizar a documentação do perfil vinculado para esse serviço.

Permissões de perfil vinculado ao serviço para o Amazon DocumentDB

O Amazon DocumentDB (com compatibilidade com o MongoDB) usa a função vinculada ao serviço chamada AWSServiceRoleForRDS para permitir que o Amazon DocumentDB chame AWS serviços em nome de seus clusters.

A função AWSServiceRoleForRDS vinculada ao serviço confia nos seguintes serviços para assumir a função:

- `docdb.amazonaws.com`

A política de permissões do perfil permite que o Amazon DocumentDB conclua as seguintes ações nos recursos especificados:

- Ações em `ec2`:

  - `AssignPrivateIpAddresses`
  - `AuthorizeSecurityGroupIngress`
  - `CreateNetworkInterface`
  - `CreateSecurityGroup`
  - `DeleteNetworkInterface`
  - `DeleteSecurityGroup`
  - `DescribeAvailabilityZones`
  - `DescribeInternetGateways`
  - `DescribeSecurityGroups`
  - `DescribeSubnets`
  - `DescribeVpcAttribute`
  - `DescribeVpcs`
  - `ModifyNetworkInterfaceAttribute`
  - `RevokeSecurityGroupIngress`
  - `UnassignPrivateIpAddresses`
- Ações em `sns`:
  - `ListTopic`
  - `Publish`
- Ações em `cloudwatch`:
  - `PutMetricData`
  - `GetMetricData`
  - `CreateLogStream`
  - `PullLogEvents`
  - `DescribeLogStreams`
  - `CreateLogGroup`

> **Note**
>
> É necessário configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua um perfil vinculado ao serviço. Você pode encontrar a seguinte mensagem de erro:

Impossível criar o recurso. Você se você tem permissão para criar o perfil vinculado ao serviço. Caso contrário, aguarde e tente novamente mais tarde.
Se você vir esse erro, verifique se você tem as seguintes permissões:

```
{
    "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 obter mais informações, consulte Permissões de Service-Linked função no Guia do usuário do IAM.

Criar um perfil vinculado ao serviço para o Amazon DocumentDB

Não é necessário criar manualmente um perfil vinculado ao serviço. Quando você cria um cluster, o Amazon DocumentDB cria um perfil vinculado ao serviço para você.

Se você excluir essa função vinculada ao serviço e precisar criá-la novamente, poderá usar esse mesmo processo para recriar a função em sua conta. Quando você cria um cluster, o Amazon DocumentDB cria um perfil vinculado ao serviço para você novamente.

Modificar um perfil vinculado ao serviço do Amazon DocumentDB

O Amazon DocumentDB não permite que você modifique a função vinculada ao AWSServiceRoleForRDS serviço. Depois que criar um perfil vinculado ao serviço, você não poderá alterar o nome do perfil, pois várias entidades podem fazer referência a ele. No entanto, você poderá modificar a descrição do perfil usando o IAM. Para obter mais informações, consulte Editando uma Service-Linked função no Guia do usuário do IAM.

Excluir um perfil vinculado ao serviço para o Amazon DocumentDB

Se você não precisar mais usar um recurso ou serviço que exija uma função vinculada ao serviço, exclua essa função. Dessa forma, você não tem uma entidade não utilizada que não seja monitorada ou mantida ativamente. No entanto, você deve excluir todos os clusters antes de excluir a função vinculada ao serviço.

Limpeza de um perfil vinculado a serviço do Amazon DocumentDB

Antes de usar o IAM para excluir uma função vinculada a um serviço, você deve primeiro confirmar que a função não tem sessões ativas e remover todos os recursos usados por ela.

Para verificar se a função vinculada ao serviço tem uma sessão ativa usando o console

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM emhttps://console.aws.amazon.com/iam/.
2. No painel de navegação do console do IAM, escolha Funções e, em seguida, escolha o nome (não a caixa de seleção) da AWSServiceRoleForRDS função.
3. Na página Resumo para a função selecionada, escolha a guia Consultor de Acesso.
4. Na guia Consultor de acesso, revise a atividade recente para a função vinculada ao serviço.

> **Note**
>
> Se você não tiver certeza se o Amazon DocumentDB está usando a AWSServiceRoleForRDS função, tente excluir a função. Se o serviço estiver usando a função, a exclusão falhará e você poderá visualizar as regiões da em que a função está sendo usada. Se a função está sendo usada, você deve aguardar a sessão final antes de excluir a função. Não é possível revogar a sessão de uma função vinculada a um serviço.

Se você quiser remover a AWSServiceRoleForRDS função, primeiro exclua todas as suas instâncias e clusters. Para obter informações sobre como excluir instâncias e clusters, consulte os seguintes tópicos:

- Excluir uma instância do Amazon DocumentDB
- Excluir um cluster do Amazon DocumentDB

Regiões suportadas para funções vinculadas ao serviço Amazon DocumentDB

O Amazon DocumentDB oferece suporte a funções vinculadas a serviços em todas as regiões em que o serviço está disponível. Para obter mais informações, consulte https://docs.aws.amazon.com/documentdb/latest/devguide/regions-and-azs.html#regions-and-azs-availability.

Otimização de desempenho

Este capítulo fornece orientação sobre como otimizar o desempenho do Amazon DocumentDB e solucionar problemas comuns de desempenho.

Tópicos

- Problemas comuns de desempenho
- Anti-patterns
- Dicas de melhoria de desempenho
- Análise do plano de consulta
- Index-only escaneamentos (consultas cobertas)
- Planejar filtros de cache
- Práticas recomendadas do Amazon DocumentDB
- Gerenciar índices do Amazon DocumentDB
- Gerenciamento da compactação de documentos a nível de coleção
- Gerenciando a compactação baseada em dicionário no Amazon DocumentDB 8.0

Problemas comuns de desempenho

Esta seção aborda problemas comuns de desempenho que você pode encontrar com o Amazon DocumentDB e como resolvê-los.

Tópicos

- Alta utilização da CPU
- Problemas de conexão com o Amazon DocumentDB
- Consulta em execução lenta

Alta utilização da CPU

A alta utilização da CPU é um dos problemas que você pode encontrar em algum momento ao trabalhar com o Amazon DocumentDB. Esta seção fornece informações para resolver o problema.

Identificação - Identifique o problema

A alta utilização da CPU nas instâncias do Amazon DocumentDB causa lentidão nos aplicativos ao aumentar as latências em todas as operações do banco de dados. Quando o uso da CPU do Amazon DocumentDB sobe acima dos níveis ideais, os tempos de execução das consultas aumentam, o estabelecimento da conexão leva mais tempo e a capacidade de resposta do aplicativo diminui. Os picos de CPU na instância primária diminuem as operações de gravação, enquanto os picos nas réplicas resultam em maior latência nas operações de leitura.

Algumas das causas comuns da alta utilização da CPU no Amazon DocumentDB podem ser:

- Escaneamentos de coleção: consultas sem índices adequados forçam escaneamentos completos da coleção, consumindo CPU excessiva para examinar cada documento.
- Picos de conexão: picos repentinos nas conexões de aplicativos sobrecarregam os recursos de CPU necessários para autenticação e gerenciamento de sessões.
- Agregações complexas: as Multi-stage operações (classificação, agrupamento, união) consomem uma quantidade significativa de CPU, especialmente em grandes conjuntos de dados sem índices otimizados.
- Consultas ineficientes: o alto uso da CPU resulta da verificação de grandes conjuntos de resultados e operações de regex.
- Write-Heavy Cargas de trabalho: atualizações frequentes e inserções em massa sobrecarregam a CPU com processamento de documentos, atualizações de índices e registro de transações.

Investigue - Reúna métricas

O Amazon DocumentDB fornece monitoramento por meio da Amazon. CloudWatch As métricas do Amazon DocumentDB podem ser amplamente categorizadas em métricas de nível de cluster e nível de instância. Para recursos relacionados à CPU, memória e conexões, veja as métricas no nível da instância, pois o consumo de recursos é específico da instância. Identifique as instâncias (primárias ou de réplica) que estão mostrando os picos da CPU e anote os horários dos picos.

A CloudWatch métrica da utilização da CPU para`CPUUtilization`.

Diagnosticar - Encontre a causa raiz

Depois de identificar a instância e o cronograma do pico da CPU, o próximo passo é diagnosticar a causa que está afetando o uso da CPU em geral. Como há vários motivos que podem causar picos de CPU, vamos identificar o que está contribuindo para isso, começando com:

Conexões: verifique CloudWatch as métricas, `DatabaseConnections` incluindo: se houver picos nas conexões e isso coincidir com o tempo de pico da CPU, o aumento nas conexões em um curto período de tempo provavelmente é a causa do pico da CPU. `DatabaseConnectionsMax`

Consultas: se não for a conexão, uma consulta pode ser a causa. Se você ativou o Performance Insights, acesse o console Performance Insights da instância e veja as consultas na instância. A outra coisa a se observar nas consultas é do profiler.

Resolver - Corrija o problema

Intermitências de conexão: se as rajadas de conexão forem a causa de picos de CPU, considere reutilizar conexões em seu aplicativo ou implementar o pool de conexões. Para obter mais informações sobre como otimizar conexões altas, consultethe section called “Problemas de conexão”.

Distribuição de consultas: se os picos de CPU forem causados por consultas de leitura na instância primária, considere redirecionar as operações de leitura para instâncias de réplica se os resultados da consulta puderem ser consistentes.

Otimizações de consulta: se uma consulta ou um conjunto de consultas estiver causando picos na CPU, considere otimizar a consulta, como adicionar um índice se a consulta não estiver usando um. Para obter mais informações sobre como otimizar uma consulta, consultethe section called “Consulta em execução lenta”.

Escalabilidade da instância: se a utilização da CPU permanecer consistentemente alta após os esforços de otimização, atualize para uma classe de instância maior com mais capacidade de CPU para lidar com seus requisitos de carga de trabalho de forma eficaz.

Sem servidor: para cargas de trabalho imprevisíveis que causam picos de CPU de curta duração, considere usar. Uso do Amazon DocumentDB Sem Servidor As instâncias sem servidor escalam automaticamente os recursos com base na demanda, eliminando a necessidade de provisionar tamanhos de instância fixos para essas cargas de trabalho.

Problemas de conexão com o Amazon DocumentDB

Identificação - Identifique o problema

Causas comuns

Os problemas de conexão geralmente decorrem de três áreas principais:

A exaustão do pool de conexões ocorre quando um aplicativo atinge suas conexões máximas permitidas com o Amazon DocumentDB, seja por meio dos limites do pool de conexões do lado do cliente ou dos limites da instância do lado do servidor. Essa condição leva à degradação do desempenho do aplicativo, ao tempo limite e a possíveis falhas à medida que novas solicitações de conexão são colocadas em fila ou rejeitadas.

A sobrecarga de autenticação ocorre quando o Amazon DocumentDB recebe solicitações excessivas de autenticação simultânea, especialmente ao processar mais de 1.000 novas conexões em um curto período de tempo. Durante a manutenção da autenticação, o Amazon DocumentDB mantém um bloqueio exclusivo no mapa de sessões, fazendo com que as tentativas subsequentes de autenticação fiquem na fila até que a manutenção seja concluída.

Os problemas de configuração no Amazon DocumentDB geralmente resultam de configurações incorretas nas configurações de rede, segurança e clientes. Isso inclui itens como configurações incorretas do grupo de segurança, configuração incorreta da VPC ou problemas de SSL/TLS certificado. Compreender a configuração adequada é essencial para manter o acesso seguro e confiável ao banco de dados.

Diagnóstico - Encontre a causa raiz

Pools de conexão

O pool de conexões é inicializado ao criar uma MongoClient instância. Cada pool mantém conexões com base em dois parâmetros principais:

min PoolSize - Número mínimo de conexões mantidas

max PoolSize - Máximo de conexões permitidas

Quando uma solicitação precisa de uma conexão:

1. O pool verifica as conexões inativas disponíveis
2. Se não existir nenhum e o tamanho do pool for < máximoPoolSize, ele criará uma nova conexão
3. Se no máximoPoolSize, a solicitação entra em uma fila de espera
4. Se a fila estiver cheia ou o tempo limite for atingido, ela será lançada MongoWaitQueueFullException

O comportamento da fila de espera é tratado por meio desses parâmetros:

wait QueueTimeout MS - Tempo máximo de espera pela conexão

wait QueueSize - Máximo de solicitações em fila

Aqui está um exemplo de uma abordagem problemática para se conectar ao Amazon DocumentDB, em que um novo pool é criado a cada vez:

```
for(Request request : requests) {
    MongoClient client = MongoClients.create(settings);
    // Process request
    client.close();
}
```

CloudWatch As métricas críticas a serem monitoradas são:

- `DatabaseConnections`- O número de conexões (ativas e inativas) abertas em uma instância tomada com uma frequência de 1 minuto.
- `DatabaseConnectionsMax`- O número máximo de conexões abertas de banco de dados (ativas e inativas) em uma instância em um período de 1 minuto.
- `DatabaseConnectionsLimit`- O número máximo de conexões simultâneas de banco de dados (ativas e inativas) permitidas em uma instância a qualquer momento.
- `LowMemNumOperationsThrottled`- O número de solicitações que são limitadas devido à baixa memória disponível em um período de 1 minuto.

Para cotas por classe de instância, consulteCotas de instâncias.

Os sinais de alerta comuns de problemas no pool de conexões no nível do aplicativo incluem:

- Aumento dos tempos de aquisição de conexão
- Aumento do tamanho da fila de espera
- Aumento do número de exceções de tempo limite

Sobrecarga de autenticação

A conexão com o Amazon DocumentDB segue um fluxo semelhante ao abaixo:

Solicitação de conexão → SSL Handshake → Autenticação → Criação de sessão → Conexão pronta

Ao processar mais de 1.000 novas conexões, solicitações de conexão adicionais entrarão em uma fila para autenticação após concluírem o handshake SSL. O tempo médio de conexão do seu aplicativo aumentará durante esses eventos de sobrecarga.

CloudWatch As métricas críticas a serem monitoradas são:

- `DatabaseConnections`- O número de conexões (ativas e inativas) abertas em uma instância tomada com uma frequência de 1 minuto.
- `DatabaseConnectionsMax`- O número máximo de conexões abertas de banco de dados (ativas e inativas) em uma instância em um período de 1 minuto.
- `DatabaseConnectionsLimit`- O número máximo de conexões simultâneas de banco de dados (ativas e inativas) permitidas em uma instância a qualquer momento.

Problemas de configuração

O problema de configuração mais comum é causado ao tentar se conectar a um endpoint de cluster privado do Amazon DocumentDB a partir de um ambiente sem acesso ao ambiente de rede privada. O Amazon DocumentDB é apenas uma nuvem privada virtual (VPC) e não oferece suporte a endpoints públicos no momento. Você não pode se conectar diretamente ao seu cluster Amazon DocumentDB a partir do seu laptop ou ambiente de desenvolvimento local fora da sua VPC.

Isso se manifestará em erros como os seguintes:

```
Error: couldn't connect to server...
Failed to connect to...
exception: connect failed
connection attempt failed
```

Configurações incorretas do grupo de segurança também podem causar falhas na conexão. Por padrão, um cluster Amazon DocumentDB escuta conexões na porta TCP 27017. Seu aplicativo falhará se tentar se conectar a uma porta diferente daquela com a qual o cluster foi implantado ou se o aplicativo não estiver coberto na configuração do grupo de segurança de entrada do cluster.

O gerenciamento incorreto de certificados também pode causar problemas de conexão. Por padrão, a criptografia em trânsito é ativada para clusters recém-criados do Amazon DocumentDB. Quando a criptografia em trânsito está habilitada, conexões seguras usando TLS são necessárias para

se conectar ao cluster usando o certificado global-bundle.pem. Se você tentar usar o certificado incorreto, receberá erros como:

```
unable to get local issuer certificate
```

Se tentar se conectar a um cluster com o TLS ativado sem especificar os parâmetros do TLS, você receberá erros como:

```
Server selection timed out after 30000 ms
```

Resolver - Corrija o problema

Pools de conexão: analise o pool de conexões implementando ou ajustando os tamanhos dos pools para atender aos requisitos de carga de trabalho. As configurações ideais do pool dependem da sua carga de trabalho e dos seus requisitos. Você deve manter um mínimo PoolSize para que as conexões principais estejam prontas e disponíveis, e um máximo WaitTime curto o suficiente para falhar rapidamente se o pool estiver esgotado.

Aqui está um exemplo de como reutilizar um único pool sem criar um novo a cada vez:

```
MongoClient client = MongoClients.create(settings);
    for(Request request : requests) {
    // Process request
}
```

Sobrecarga de autenticação: gerencie a autenticação implementando o aumento gradual da conexão e limitando as novas conexões a 1.000 por vez. Use o pool de conexões para reutilizar conexões autenticadas de forma eficaz. Para evitar sobrecarregar o cluster Amazon DocumentDB com conexões, implemente uma estratégia de aumento de conexão.

```
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;
        }
    }
}
```

Você também pode definir as configurações do pool de conexões para limitar o número total de conexões 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 configuração: garanta que seu aplicativo tenha acesso à VPC privada e à sub-rede onde seus recursos do Amazon DocumentDB estão localizados. Se estiver usando o emparelhamento de VPC, consulte o guia do desenvolvedor Solucionar problemas de uma conexão de emparelhamento de VPC para obter mais informações. Você também pode revisar o artigo do Centro de Conhecimento Como soluciono problemas de conectividade da Internet com instâncias do Amazon EC2 em minha VPC? .

Para a configuração do grupo de segurança, você deve incluir uma regra de entrada no seu grupo de segurança do Amazon DocumentDB para permitir conexões do seu aplicativo.

```
{
  "SecurityGroupIngress": [
    {
      "IpProtocol": "tcp",
```

```
      "FromPort": 27017,
      "ToPort": 27017,
      "SourceSecurityGroupId": "<application-security-group>",
      "Description": "DocumentDB access from application tier"
    }
  ],
  "SecurityGroupEgress": [
    {
      "IpProtocol": "-1",
      "FromPort": -1,
      "ToPort": -1,
      "CidrIp": "0.0.0.0/0"
    }
  ]
}
```

Se o cluster estiver configurado com criptografia TLS, baixe o certificado TLS para o Amazon Amazon DocumentDB chamado global-bundle.pem e use-o ao conectar-se ao cluster.

```
wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
```

Long-term Soluções

O escalonamento da instância pode ser necessário por meio da atualização para uma classe de instância maior ou da adição de réplicas de leitura para distribuir a carga da conexão. A implementação adequada do balanceamento de carga garante a utilização ideal dos recursos em todo o cluster.

As mudanças nos aplicativos devem se concentrar na implementação de um tratamento robusto de conexões, monitoramento abrangente e adesão às melhores práticas de pool de conexões. Isso inclui o tratamento adequado de erros e o gerenciamento do ciclo de vida da conexão.

As melhorias na arquitetura podem envolver a adoção do Amazon DocumentDB Serverless para cargas de trabalho variáveis, a implementação de uma lógica de repetição sofisticada e o design para tolerância a falhas. Considere reestruturar a arquitetura do aplicativo para lidar melhor com o gerenciamento de conexões.

Práticas recomendadas

Pools de conexão

Por meio do gerenciamento e monitoramento adequados do pool de conexões, os aplicativos podem manter a conectividade estável do banco de dados e, ao mesmo tempo, evitar cenários de exaustão que podem afetar a confiabilidade e o desempenho do sistema. Configure os tempos limite apropriados e dimensione seu pool com base nas características da sua carga de trabalho.

Exemplo de configuração de pool de conexões

```
MongoClientSettings settings = MongoClientSettings.builder()
    .applyToConnectionPoolSettings(builder ->
        builder.maxSize(10))
    .applyToConnectionPoolSettings(builder ->
        builder.maxWaitQueueSize(2))
    .applyToConnectionPoolSettings(builder ->
        builder.maxConnectionIdleTime(10, TimeUnit.MINUTES))
    .build();
```

Para obter mais informações, consulte Criação de aplicativos resilientes com o Amazon DocumentDB — Parte 1: Configuração do cliente.

Sobrecarga de autenticação

Sempre implemente o pool de conexões com valores apropriados para parâmetros com base em sua carga de trabalho. Use uma técnica de estabelecimento de conexão gradual e mantenha conexões persistentes, sempre que possível. Implemente a limpeza adequada da conexão para garantir que nenhum recurso ocioso seja desperdiçado.

Problemas de configuração

Certifique-se de ter configurado o roteamento apropriado do seu aplicativo para os recursos do Amazon DocumentDB. Utilize o TLS para criptografia em trânsito e implemente o acesso com menos privilégios. Verifique suas credenciais do Amazon DocumentDB e valide os valores da string de conexão.

Consulta em execução lenta

Identificação - Identifique o problema

Uma consulta de execução lenta é aquela que excede o tempo de execução esperado para seus requisitos de negócios. A definição do que constitui uma consulta lenta varia entre diferentes cargas

de trabalho. Você pode identificar consultas lentas por meio de registros do profiler ou Performance Insights.

1. Ative o profiler se ainda não estiver ativado. Os registros do Profiler são gravados na Amazon CloudWatch sob o Log Group:`/aws/docdb/<cluster-identifier>/profiler`. Use o CloudWatch Logs Insights para consultá-los.

   Exemplo de consulta de análise de CloudWatch log para obter as 10 consultas mais lentas da coleção ecommerce.products:

   ```
   filter ns="ecommerce.products" | sort millis desc | limit 10
   ```

2. Use o Performance Insights para identificar consultas caras quase em tempo real. Ative o Performance Insights se ele ainda não estiver ativado.

   a. Abra o AWS console, navegue até Amazon Amazon DocumentDB, depois Performance Insights e selecione sua instância de cluster.

   b. Analise o cronograma de carregamento de banco de dados (AAS) e as principais consultas (por carga de banco de dados). Expanda um resumo de consulta para ver declarações secundárias literais.

   c. Capture as consultas que exigem análise.

   > **Note**
   >
   > Nem todas as consultas no Performance Insights podem ser ineficientes ou lentas.

Investigue — Reúna informações

1. O criador de perfil fornece o plano de execução da consulta e as principais métricas associadas a ele, incluindo millis, nreturned e planSummary (uso do índice):

   ```
   {
       "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 encontrar as consultas do COLLSCAN:

```
filter planSummary="COLLSCAN" | sort millis desc | limit 20
```

2. Use o Performance Insights para analisar tendências de execução de consultas, como esperas, carga e impacto nos recursos (por exemplo, IOPS ou CPU por consulta) em tempo real.

   Como o Performance Insights não fornece o plano de execução da consulta, capture a consulta e execute explain (“ExecutionStats”) na consulta em seu cluster Amazon DocumentDB:

   ```
   db.ecommerce.products.find().explain("executionStats")
   ```

3. Opcionalmente, correlacione as métricas do profiler com os dados do Performance Insights (por exemplo, combine muitos milhões de consultas no Profiler com as principais consultas no Performance Insights).

Diagnosticar — Encontre a causa raiz

Nesta etapa, diagnostique o plano de consulta para identificar possíveis otimizações. Use o seguinte fluxo - sintoma → causa provável:

- Sintoma: Resumo do plano: “COLLSCAN”

  Causa: Índice ausente ou incorreto.

- Sintoma: agregação lenta com $group/$sort

  Causa: o pipeline de agregação pode estar processando muitos dados na memória.

- Sintoma: altas latências enquanto o Performance Insights mostra a carga de banco de dados agrupada por esperas de E/S, embora os índices sejam utilizados (PlanSummary: “IXSCAN”).

  Causa: consultas I/O vinculadas; índices ou conjuntos de trabalho excedem o cache de buffer disponível na instância.

- Sintoma: PI mostra estados de espera da CPU, alto AAS devido a poucas consultas

  Causa: CPU-bound consultas (agregações complexas, $regex).

- Sintoma: muitas consultas lentas, mas nenhuma mostra um plano caro. Resumo

  Causa: gargalo externo (rede, limitação, tarefas de manutenção, volume de consultas).

Resolver — Corrija o problema

Ao implementar correções, sempre valide com explain (“ExecutionStats”) e monitore a carga de banco de dados do Performance Insights.

1. Resumo do plano: “COLLSCAN”
   - Crie um índice direcionado.

     Exemplo: Para consultas frequentes que filtram por {categoria, preço} e classificam por preço decrescente:

     ```
     db.products.createIndex({ category: 1, price: -1 })
     ```
2. agregação lenta com $group/$sort
   - Pressione $match e $project com antecedência para reduzir o fluxo de documentos para $group/$sort.
   - Limite o número de campos no início do pipeline:

     ```
     db.products.explain("executionStats").aggregate([
       { $match: { category: "Electronics", price: { $gt: 0 } } },   // early filter
       { $project: { price: 1, category: 1 } },                     // reduce document
      size
       { $group: { _id: "$category", avgPrice: { $avg: "$price" } } }
     ])
     ```
3. altas latências, enquanto o Performance Insights mostra a carga de banco de dados agrupada por esperas de E/S
   - Valide se BufferCacheHitRatio está baixo ou se os ReadiOps na instância estão altos. Aumente a memória da instância (atualize a classe da instância, por exemplo, r6g.large → r6g.xlarge) ou use a classe de instância NVMe.
   - Reduza a pegada do índice.
   - Adicione réplicas de leitura para descarregar o tráfego de leitura (use a configuração ReadPreference para redirecionar o tráfego nas réplicas se a consulta for tolerante com uma eventual consistência).
4. PI mostra estados de espera da CPU, alto AAS devido a poucas consultas

- Substitua o caro $regex por pesquisas de prefixo indexado ou índice $text.
- Gravações em lote (inserção) para reduzir a amplificação de gravação.

> **Note**
>
> Sempre teste suas alterações em um ambiente inferior (sem produção) antes de promover essas alterações para a Produção.

Anti-patterns

Esta seção aborda antipadrões comuns que podem afetar negativamente o desempenho do Amazon DocumentDB.

Tópicos

- Índice composto com mais de 3 atributos
- Consultas de longa duração
- Índices pouco usados e redundantes
- Multi-Key Índices com matrizes grandes

Índice composto com mais de 3 atributos

Visão geral do

Um índice composto ou composto mantém referências a vários campos em uma única estrutura de índice. Esses índices otimizam o desempenho de consultas que filtram em vários campos simultaneamente ou combinam filtragem com operações de classificação. Eles também são eficazes para consultas de condição única nos campos indexados mais à esquerda. O banco de dados aproveita essas entradas de índice para localizar com eficiência os documentos correspondentes sem realizar digitalizações completas da coleção.

O Amazon DocumentDB pode usar o índice composto para oferecer suporte a consultas que incluam qualquer subconjunto principal dos campos indexados, um conceito conhecido como prefixos de índice. Por exemplo, se você tem um índice composto ativado`{state: 1, city: 1, zipcode: 1}`, o Amazon DocumentDB pode processar com eficiência consultas que usam somente o campo 'estado', os campos 'estado' e 'cidade' juntos, ou todos os três campos 'estado', 'cidade' e 'CEP”.

No entanto, as consultas devem usar os campos da esquerda para a direita sem ignorar nenhum campo intermediário. Isso significa que consultas usando somente o campo 'cidade' ou 'CEP', ou combinações como 'estado' e 'CEP' (ignorando 'cidade'), não podem utilizar totalmente o índice.

Na maioria dos cenários do mundo real, índices compostos com três ou menos atributos normalmente alcançam desempenho e eficiência de recursos ideais. Embora seja possível ter mais de 3 atributos no índice composto, isso geralmente leva ao aumento do consumo de recursos, superando os benefícios.

Impacto no cluster

Embora seja tentador criar um índice cobrindo todas as condições em uma consulta para obter o máximo desempenho teórico, a maior parte da filtragem de dados ocorre por meio dos primeiros 1-3 atributos em um índice composto. Campos adicionais além desse limite contribuem principalmente para o tamanho do índice, em vez de uma otimização significativa da consulta.

Armazenamento e I/O sobrecarga: índices compostos com muitos atributos consomem muito mais espaço de armazenamento do que alternativas mais simples. O tamanho é diretamente proporcional ao número de atributos indexados e ao tamanho dos próprios valores indexados.

Espaço de memória: o grande espaço de armazenamento de índices compostos com muitos atributos cria um espaço correspondente na memória, aumentando seu conjunto de trabalho e deslocando outros dados acessados com frequência do buffer pool.

Operações de gravação: cada modificação do documento que afeta vários campos indexados exige a atualização de toda a entrada do índice composto, multiplicando o trabalho necessário para concluir a operação de gravação.

Como identificar

Comece analisando todos os índices em sua coleção para identificar índices compostos com mais de três 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 }
```

Correção

Se um índice composto com mais de três atributos estiver sendo usado, identifique as consultas que o utilizam e procure oportunidades de otimização. Considere substituir esses índices por outros mais eficientes contendo três ou menos atributos. Depois de implementar os novos índices, remova os índices antigos que contêm mais de três atributos.

> **Note**
>
> Sempre coordene com as partes interessadas e valide o impacto no desempenho antes de descartar qualquer índice.

Siga a regra Equality, Sort, Range (ESR) ao criar índices compostos.

Regra de faixa de classificação de igualdade (ESR): essa ordenação maximiza a eficiência do índice filtrando o conjunto de dados primeiro com condições de igualdade, depois aplicando operações de classificação no conjunto reduzido e, finalmente, realizando varreduras de faixa no menor subconjunto possível

- campos de ordem como igualdade (correspondências exatas),
- Classificar (ordenar),
- Intervalo (>, <, $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 longa duração

Visão geral do

Long-running as consultas podem se transformar em problemas de desempenho em todo o cluster. Essas consultas podem interferir no processo de coleta de lixo do Amazon DocumentDB Multi-Version Concurrency Control (MVCC), levando ao acúmulo de documentos com versões antigas e à degradação do desempenho em todo o cluster. O Amazon DocumentDB implementa um tempo limite de 2 horas no servidor como um mecanismo de segurança para impedir que consultas descontroladas consumam recursos indefinidamente.

O que são consultas de longa duração:

- Consultas que são executadas por longos períodos (normalmente **>** 30 minutos)
- Cursores abertos que permanecem ativos por horas (o tempo limite de 2 horas não será aplicável se o cursor estiver ativo)

Impacto no cluster

Consultas de longa duração podem interferir no processo de coleta de lixo

- Versões antigas se acumulam: o coletor de lixo não pode recuperar versões antigas do documento
- Inchaço da coleção e do índice: as entradas da coleção e do índice se acumulam com o tempo, o inchaço aumenta e isso pode resultar em mais custos de armazenamento.
- Pressão da CPU e da memória: a pressão da CPU e da memória aumenta devido ao processamento ineficiente do maior número de versões de documentos antigos, entradas de índice e IDs de transações.

Consulta de longa duração → Bloqueia GC → Crescimento do armazenamento → Pressão da CPU e da memória → Mais consultas longas

Monitore e detecte

1. Para encontrar consultas de longa duração, use o 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 encontrar cursores ativos por mais 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. Monitorando o progresso do coletor de lixo por meio de CloudWatch

`LongestRunningGCProcess`— Duração em segundos do processo de coleta de lixo ativo mais longo. Atualiza a cada minuto e rastreia somente as operações ativas, excluindo os processos concluídos dentro da janela de um minuto.

`AvailableMVCCIds`-Um contador que mostra o número de operações de gravação restantes disponíveis antes de chegar a zero. Quando esse contador chega a zero, seu cluster entra no modo somente de leitura até que os IDs sejam recuperados e reciclados. O contador diminui a cada operação de gravação e aumenta à medida que a coleta de resíduos recicla IDs de MVCC antigos.

> **Note**
>
> IDs de MVCC mais baixos e maior duração da coleta de lixo não são atribuídos exclusivamente a consultas de longa duração. Write-intensive cargas de trabalho em

instâncias com recursos limitados também podem resultar na redução da disponibilidade do MVCC ID e em ciclos prolongados de coleta de lixo.

Estratégias de remediação

- Implemente tempos limite de consulta no aplicativo
- Não mantenha os cursores ativos por períodos mais longos
- Otimize as consultas para obter um melhor desempenho.
- Prefira o agrupamento em lotes de operações de gravação

Índices pouco usados e redundantes

Visão geral do

Os índices melhoram o desempenho geral das consultas das cargas de trabalho, mas têm custos. Cada índice consome armazenamento e I/O recursos, e as operações de gravação precisam atualizar todos os índices na coleção. Se seu banco de dados contiver índices não utilizados, subutilizados ou redundantes, isso resultará em sobrecarga para o cluster, levando à degradação do desempenho. Vários índices desnecessários criam um antipadrão que causa problemas de desempenho, maiores custos de armazenamento e maior sobrecarga operacional.

Cenários de indexação abaixo do ideal

- Índices não utilizados: índices criados para consultas únicas ou em iterações anteriores do produto que não são mais acessados pelos padrões de consulta atuais. Para identificar índices não utilizados em uma coleção, você pode usar IndexStats. Para obter mais informações, consulte the section called “Analise o uso do índice e identifique índices não utilizados”.
- Índices redundantes: vários índices que abrangem padrões de chave sobrepostos ou padrões de consulta idênticos. Exemplo: um índice de chave única no atributo A se torna redundante quando um índice composto abrange os atributos A e B, pois o índice composto pode lidar com consultas somente no atributo A.
- Over-Indexing: criar índices “por precaução” sem analisar os padrões reais de consulta ou os requisitos de desempenho.

- Índices de baixa cardinalidade: índices em campos com poucos valores distintos (por exemplo, campos booleanos, sinalizadores de status) que oferecem benefícios mínimos de otimização de consultas.

Impacto no desempenho do cluster

- Armazenamento e E/S: cada índice consome espaço de armazenamento e recursos de E/S, contribuindo para os custos gerais. Se um índice não for usado ou for redundante, isso representará um custo desnecessário em seu cluster.
- Desempenho de gravação reduzido: as operações de inserção, atualização e exclusão devem manter índices, criando uma sobrecarga que consome recursos. Índices não utilizados ou redundantes adicionam sobrecarga desnecessária a essas operações.
- Pressão de memória e CPU: os índices competem pela memória do buffer pool e podem despejar dados acessados com frequência se o conjunto de trabalho não couber na memória da instância, resultando em uma memória abaixo do ideal. BufferCacheHitRatio O sistema também consome recursos da CPU mantendo as estruturas de índice.

Ferramentas disponíveis para identificar oportunidades de otimização de índices

- Ferramenta de revisão de índice:
- Ferramenta de detecção de cardinalidade de índice

Reunindo o kit de ferramentas:

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

1. Ferramenta de revisão de índice

A ferramenta de revisão de índices analisa todas as coleções e índices para fornecer uma lista de índices, seus padrões de uso e identificar índices redundantes em seu cluster Amazon DocumentDB. Execute essa ferramenta em todas as instâncias do cluster para uma análise abrangente. Para obter mais informações, consulte Index Review Tool.

Uso:

```
cd performance/index-review/

python3 index-review.py --server-alias <server-alias> --uri <mongodb-uri>
```

Arquivos de saída

- `{server-alias}-collections.csv`: Collection-level estatísticas
- `{server-alias}-indexes.csv`: métricas detalhadas de uso do índice
- `{server-alias}-{timestamp}-index-review.json`: Dados brutos para análise posterior

> **Note**
>
> Execute para cada instância no cluster com um alias independente para gerar estatísticas de cada instância e, em seguida, execute a ferramenta para obter uma análise cumulativa

Por exemplo: `python3 index-review.py --files "{server-alias}-{timestamp}-index-review.json, {server-alias}-{timestamp}-index-review.json" --server-alias full-review`

2. Ferramenta de detecção de cardinalidade de índice

Essa ferramenta identifica índices com baixa cardinalidade que são ineficientes para o desempenho da consulta.

Uso

```
cd performance/index-cardinality-detection/

python3 detect-cardinality.py --uri <mongodb-uri>
```

Parâmetros

- `--threshold`: Porcentagem do limite de cardinalidade (padrão: 1%)
- `--sample-count`: Número de documentos a serem amostrados (padrão: 100.000)

- `--max-collections`: máximo de coleções a serem examinadas por banco de dados (padrão: 100)

Para obter mais informações, consulte https://aws.amazon.com/blogs/database/detect-and-fix-low-cardinality-indexes-in-amazon-documentdb/.

Estratégias de remediação

1. Remoção de índice não utilizado e redundante

```
// Example: Drop unused index
db.collection.dropIndex("unused_index_name")
```

> **Note**
>
> Os índices nunca devem ser descartados sem discutir com todas as partes interessadas e testar o desempenho.

2. Otimização do baixo índice de cardinalidade

- Use índices parciais com filtros
- Converta índices únicos de baixa cardinalidade em índices compostos

Multi-Key Índices com matrizes grandes

Visão geral do

Multi-key índices no Amazon DocumentDB permitem que você consulte campos de matriz de forma eficiente. Essa estratégia de indexação oferece latências de consulta significativamente menores ao pesquisar em dados de matriz, tornando-a uma opção atraente para aplicativos com estruturas de dados complexas.

No entanto, ao trabalhar com matrizes grandes ou vários campos de matriz em uma coleção, é importante entender as implicações dos recursos e as características de desempenho para tomar

decisões arquitetônicas informadas. Quando você cria um índice em um campo de matriz, o Amazon DocumentDB gera entradas de índice individuais para cada elemento na matriz.

Impacto no cluster

- Armazenamento e sobrecarga de E/S: às vezes, Multi-key os índices podem consumir espaço de armazenamento até várias vezes da tabela base; o tamanho é diretamente proporcional aos documentos da coleção com o atributo de matriz indexada, número de elementos na matriz e tamanho do elemento.
- Uso de memória: como o espaço de armazenamento é relativamente grande, esses índices ocupam uma área correspondente na memória, resultando em um conjunto de trabalho maior.
- Operações de gravação: Multi-key os índices criam sobrecarga durante as operações de gravação. Cada elemento da matriz gera entradas de índice separadas, multiplicando o trabalho necessário para concluir a operação de gravação.

Estratégias de remediação

- Crie um índice de várias chaves, se necessário, e limite o número de campos na matriz.
- Limite o número de índices de várias chaves na coleção
- Considere modificar seu modelo de dados

Dicas de melhoria de desempenho

Esta seção fornece cinco estratégias de otimização de desempenho do Amazon DocumentDB para melhorar a eficiência do aplicativo e a execução de consultas.

1. Use $match como primeiro estágio em pipelines de agregação

Sempre coloque $match como o primeiro estágio de filtragem em seu pipeline de agregação para maximizar o desempenho. O Amazon DocumentDB utilizará índices de forma eficaz quando $match liderar o pipeline, permitindo que o banco de dados filtre os dados com antecedência e reduza a sobrecarga de processamento.

```
// Optimized approach
db.orders.aggregate([
  { $match: { status: "active", category: "electronics" } }, // Index utilization
```

```
  { $group: { _id: "$category", total: { $sum: "$price" } } },
  { $sort: { total: -1 } }
])
```

Impacto: a filtragem antecipada reduz o número de documentos processados nas etapas subsequentes do pipeline, resultando em uma execução mais rápida de consultas e menor consumo de recursos.

2. Use $project no pipeline de agregação para minimizar o tamanho dos dados do pipeline

Leve somente os campos essenciais pelos estágios do pipeline de agregação para minimizar o tamanho dos dados e melhorar o desempenho. Use $project estrategicamente para incluir apenas os dados de que você precisa.

```
// 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: documentos menores reduzem o uso de memória e melhoram a eficiência do processamento do pipeline, resultando em um melhor desempenho geral da consulta.

3. Ative a compactação de documentos para reduzir os custos e I/O custos de armazenamento e melhorar o desempenho das consultas

Ative a compactação de documentos do grupo de parâmetros do cluster para reduzir os custos e I/O custos de armazenamento e aumentar o desempenho das consultas. O Amazon DocumentDB armazena documentos compactados em disco e também na RAM, reduzindo o consumo de memória e os custos. I/O

Impacto:

- Mais documentos cabem na memória disponível
- Acesso mais rápido aos dados com leituras de disco reduzidas

- Custos e I/O custos de armazenamento mais baixos e melhor desempenho de consultas

> **Note**
>
> O Amazon DocumentDB não habilita a compactação por padrão para a versão 5.0. Você pode ativar a compactação no nível da coleção ou do cluster para o cluster 5.0. Use o utilitário de revisão de compactação do Amazon DocumentDB para analisar as taxas de compactação de suas coleções.
> Para o Amazon DocumentDB 8.0, a compactação é ativada por padrão.

4. Utilize índices para um desempenho de consulta ideal

Garanta que suas consultas sempre utilizem índices para um desempenho ideal. O Amazon DocumentDB oferece vários tipos de índice para corresponder a diferentes casos de uso.

Princípios de indexação:

- Cada consulta deve utilizar um índice apropriado
- O Amazon DocumentDB fornece vários tipos de índice
- Os índices compostos oferecem a maior flexibilidade ao oferecer suporte a várias formas de consulta com um único índice.
- Projete índices para suportar operações de classificação e filtragem em conjunto

Compreendendo os prefixos de índice: os índices compostos funcionam por meio de prefixos de índice - o Amazon DocumentDB pode usar qualquer subconjunto dos campos de índice da esquerda para a direita. Por exemplo, o índice `{ category: 1, price: -1, inStock: 1 }` cria esses prefixos utilizáveis:

- `{ category: 1 }`- suporta filtragem de consultas somente por categoria
- `{ category: 1, price: -1 }`- suporta filtragem de consultas por categoria e sorting/filtering por preço
- `{ category: 1, price: -1, inStock: 1 }`- suporta a consulta composta completa

As consultas sobre preço, InStock ou InStock sozinhas não usarão esse índice, pois não começam com o primeiro campo (categoria).

Como identificar consultas que não usam índices: Use o método explain () para analisar a execução de consultas e identificar consultas que realizam varreduras de coleta em vez de usar índices.

Impacto: consultas sem utilização de índice resultam em escaneamentos de coleta, causando aumento da pressão da memória e da CPU na instância e maior latência da consulta.

5. Otimize modelos de dados com base em padrões de consulta

Alinhe seu modelo de dados com a forma como seu aplicativo consulta e atualiza os dados. A modelagem de dados é a base dos aplicativos Amazon DocumentDB de alto desempenho.

Estratégias de otimização:

Incorporação para desempenho

- Armazene dados relacionados juntos quando eles são frequentemente acessados como uma unidade
- Incorpore documentos que são sempre recuperados juntos
- Adequado para relacionamentos individuais

```
// 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" }
  ]
}
```

Referência para flexibilidade

- Use referências para dados grandes ou acessados com pouca frequência
- Recomendado para relacionamentos de um para muitos com grandes conjuntos de dados

- Evita o inchaço dos documentos e melhora o desempenho da atualização

Estratégia de divisão de coleções

Quando apenas alguns campos em documentos grandes são atualizados com frequência ou quando grandes dados acessados com pouca frequência sobrecarregam os documentos, considere dividir as coleções:

- Mantenha os campos atualizados com frequência em uma coleção menor e separada
- Armazene dados estáticos ou acessados com pouca frequência em outra coleção
- Vincule-os a referências quando necessário

```
// 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 */]
}
```

Ganho de desempenho: documentos menores significam atualizações mais rápidas, menor uso de memória e maior eficiência do cache.

Impacto: a modelagem de dados ineficiente resulta em consultas abaixo do ideal, maiores tamanhos de documentos e maior uso de memória, levando à degradação do desempenho do aplicativo e ao aumento dos custos operacionais.

Análise do plano de consulta

A análise do plano de consulta por meio do plano de explicação fornece informações essenciais sobre o desempenho das consultas do Amazon DocumentDB. O plano de consulta com ExecutionStats revela as principais métricas, incluindo:

- Documentos devolvidos por estágio (nReturned)
- Stage-specific tempos de execução (execuçãoTimeMillisEstimate)
- Duração da geração do plano (planejamentoTimeMillis)

Ao examinar a saída do plano de consulta, os desenvolvedores podem analisar os padrões de execução, avaliar a utilização do índice e identificar possíveis oportunidades de otimização nos estágios do pipeline de consulta.

Para analisar o plano de consulta, você pode usar o comando explain () nos formatos a seguir.

```
db.runCommand({explain: {query document}, verbosity: "executionStats"})
db.collection.find().explain("executionStats");
```

Veja a seguir um exemplo de operação:

```
db.collection.find({ companyname: { '$eq': 'ANYCOMPANY' }, isDeleted: { '$eq':
 false } }).sort({"createdAt":1}).limit(2).explain("executionStats");
```

A saída dessa operação é semelhante à seguinte:

```
{
  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 })
}
```

Abaixo está uma análise detalhada de um plano de execução de consultas do Amazon DocumentDB, detalhando cada componente e suas características de desempenho.

Cronometragem geral

a execução TimeMillis representa o tempo total gasto pela consulta, incluindo o tempo de planejamento.

o planejamento TimeMillis representa o tempo total de planejamento gasto pela consulta.

Etapas de execução

Ele descreve o processo passo a passo que o Amazon DocumentDB usa para executar uma consulta, mostrando como os dados fluem por meio de diferentes operações.

```
"executionStages": {
    "stage": "[STAGE_NAME]",
    "nReturned": "[NUMBER_OF_DOCS]",
    "executionTimeMillisEstimate": "[TIME]",
```

```
    "inputStage": {
        // Nested stages
    }
}
```

Estágios comuns em um plano de consulta

Abaixo estão os estágios comuns de execução em um plano de consulta. Cada estágio retorna as métricas de execução TimeMillisEstimate (tempo de execução) e nReturned (número de documentos) para ajudar a avaliar o desempenho da consulta em cada estágio.

COLLSCAN (Digitalização de coleção)

- Digitaliza todo o documento da coleção por documento
- nReturned: retorna todos os documentos correspondentes na coleção
- Pode ser uma operação cara, vista quando nenhum índice está disponível

IXSCAN (Digitalização de índice)

- Usa um índice para encontrar documentos correspondentes
- nReturned: Somente documentos correspondentes com base no índice
- Operação eficiente, preferida ao COLLSCAN

SORT

- Classifica documentos com base em campos especificados
- Se o atributo de classificação na consulta não fizer parte de um índice, esse estágio aparecerá explicitamente
- nRetornado: Mesmo número dos documentos de entrada
- Memory-intensive para grandes conjuntos de resultados. Para otimizar, adicione o nome do campo de classificação a um índice

LIMIT_SKIP

- Controla o número de documentos devolvidos e ignorados

- nRetornado: Limitado pelo valor LIMITE
- Usado para operações de paginação ou LIMIT

SUBVARREDURA

- Executa operações de consulta aninhadas
- nReturned: varia com base nos resultados da subconsulta
- Usado em consultas complexas com vários estágios

Estágios com alta execução TimeMillisEstimate são bons candidatos para otimização.

> **Note**
>
> No momento, o parâmetro ExecutionStats não oferece suporte aos comandos de atualização e exclusão.

Compreendendo os documentos examinados em Explique as estatísticas de execução do plano

Quando você executa uma consulta com`explain("executionStats")`, o Amazon DocumentDB fornece métricas de exame que ajudam você a entender quantos documentos foram digitalizados para produzir os resultados da consulta. Essas métricas são úteis para identificar planos de consulta ineficientes e otimizar o desempenho.

> **Note**
>
> Disponível somente no Amazon DocumentDB 5.0.1+ e 8.0.0+.

Campos

Campo	Description	Nível
total DocsExamined	Número total de documentos examinados em todos os estágios de execução.	Top-level Estatísticas de ExecutionStats

Campo	Description	Nível
Documentos examinados	Número de documentos examinados por um estágio específico de execução.	Stage-level

O `docsExamined` campo aparece nos seguintes estágios:

Estágio	Description
COLLSCAN	Digitalização da coleção. Todos os documentos da coleção são examinados.
IXSCAN	Escaneamento de índice. Somente documentos que correspondam à condição do índice são examinados.
FETCH	Quando o otimizador recupera documentos em um estágio separado do IXSCAN, o estágio FETCH relata DocsExamined. Nos planos de escaneamento de índice, os estágios secundários do IXSCAN não relatam DocsExamined. Nos planos $lookup, tanto o estágio FETCH quanto seus estágios secundários podem relatar DocsExamined.

> **Note**
>
> `docsExamined`não aparece nos estágios do IXONLYSCAN porque a consulta é satisfeita inteiramente a partir do índice sem acessar documentos.

Mostrar documentos examinados na saída do Explain Plan ExecutionStats

Os exemplos a seguir usam uma coleção com 500.000 documentos e demonstram como isso `docsExamined` muda em diferentes estágios de execução da consulta.

Crie dados de amostra:

```
for (let i = 0; i < 500000; i++) {
```

```
    db.coll.insertOne({ number: i, arr: [i, [i+1]], value: "test", bool: i % 2 ===
 0 });
}
```

Digitalização da coleção (COLLSCAN)

Sem um índice, o Amazon DocumentDB executa uma varredura completa da coleção.

```
db.coll.find({ "number": { "$lt": 500 } }).explain("executionStats").executionStats
```

Saída:

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

Todos os 500.000 documentos foram examinados para retornar 500 resultados. Criar um índice no campo numérico melhora isso.

Análise de índice (IXSCAN)

```
db.coll.createIndex({ number: 1 });
db.coll.find({ "number": { "$lt": 5000 } }).explain("executionStats").executionStats
```

Saída:

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

Com o índice, apenas 5.000 dos 500.000 documentos foram examinados e buscados, correspondendo ao número retornado. A condição do índice filtrou 495.000 documentos que não correspondiam à consulta.

Digitalização de índice com filtro residual

```
db.coll.find({ "number": { "$lt": 5000 }, "arr": { "$gt":
 4000 } }).explain("executionStats").executionStats
```

Saída:

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

A condição do índice correspondeu a 5.000 documentos de 500.000 e, em seguida, o filtro residual `arr` reduziu o resultado para 999. O `docsExamined` valor de 5.000 reflete todos os documentos examinados após a condição do índice, mas antes da aplicação do filtro residual.

Busque com o IXSCAN

```
db.coll.find({ "$or": [{ "number": { "$lt": 100000 } }, { "number": { "$gt":
 400000 } }] }).explain("executionStats").executionStats
```

Saída:


```
{
    "executionSuccess" : true,
    "nReturned" : "199999",
    "executionTimeMillis" : "899.801",
    "planningTimeMillis" : "0.183",
    "totalDocsExamined" : "199999",
    "executionStages" : {
        "stage" : "FETCH",
        "nReturned" : "199999",
        "executionTimeMillisEstimate" : "894.141",
        "docsExamined" : "199999",
        "inputStage" : {
            "stage" : "IXOR",
            "nReturned" : "0",
            "executionTimeMillisEstimate" : "874.897",
            "inputStages" : [
                {
                    "stage" : "IXSCAN",
                    "nReturned" : "100000",
                    "executionTimeMillisEstimate" : "462.208",
                    "indexName" : "number_1",
                    "indexCond" : {
                        "$and" : [
                            {
                                "number" : {
                                    "$lt" : 100000
                                }
                            }
                        ]
                    }
                },
                {
                    "stage" : "IXSCAN",
                    "nReturned" : "99999",
                    "executionTimeMillisEstimate" : "412.684",
                    "indexName" : "number_1",
```

```
                    "indexCond" : {
                        "$and" : [
                            {
                                "number" : {
                                    "$gt" : 400000
                                }
                            }
                        ]
                    }
                }
            ]
        }
    }
}
```

Quando o otimizador Amazon DocumentDB usa um estágio de busca para recuperar documentos, o estágio FETCH relata. `docsExamined` Os estágios secundários do IXSCAN não relatam `docsExamined` porque eles apenas digitalizam chaves de índice sem acessar documentos diretamente.

FETCH com pesquisa agregada

```
db.coll.explain("executionStats").aggregate([
    { $match: { "number": { "$lt": 5 } } },
    { $lookup: { from: "coll", pipeline: [{ $match: { "number": { "$lt": 3 } } }], as:
 "sub" } }
]).executionStats
```

Saída:

```
{
    "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
                                    }
                                }
                            ]
```

```
                        }
                    }
                }
            }
        ]
    }
}
```

O IXSCAN externo examinou 5 documentos com números menores que 5. O IXSCAN interno examinou 3 documentos com número < 3, e o estágio FETCH examinou 1 resultado agregado. O `totalDocsExamined` de 9 é a soma de todos os estágios (5 + 3 + 1).

Index-only escaneamentos (consultas cobertas)

Uma verificação somente de índice (também chamada de consulta coberta) é uma otimização na qual o Amazon DocumentDB responde a uma consulta inteiramente a partir dos campos armazenados em um índice, sem ler os documentos subjacentes. Como os valores de que uma consulta precisa já estão presentes no índice, o Amazon DocumentDB pode ignorar a busca de cada documento correspondente, evitando o custo aleatório I/O e de decodificação da leitura de documentos completos. No plano de consulta, uma varredura somente de índice aparece como o `IXONLYSCAN` estágio em vez de uma `IXSCAN` (varredura de índice) ou `COLLSCAN` (varredura de coleção). Isso reduz I/O e pode diminuir significativamente a latência das consultas suportadas.

Uma consulta é uma boa candidata para uma digitalização somente de índice quando atinge apenas um pequeno número de campos que existem em um único índice, especialmente quando os documentos em si são grandes. Exemplos comuns são painéis, conjuntos de análises e pesquisas que lêem alguns campos de um índice composto mais amplo.

Os resultados de uma consulta são os mesmos, independentemente de uma varredura somente de índice ser usada ou não. Somente o plano de consulta e a quantidade de I/O desempenho são alterados.

> **Note**
>
> Index-only o suporte à digitalização depende da versão do mecanismo Amazon DocumentDB:
>
> - `find`As projeções cobertas que incluem mais de uma versão fora do `_id` campo exigem a versão 8.0.2 ou posterior do motor. Nas versões anteriores, uma `find` projeção coberta

pode incluir no máximo um `_id` não-campo. Uma projeção com dois ou mais campos volta para uma varredura de índice normal ou varredura de coleção.

- Index-only as varreduras de pipelines de agregação, como um coberto`$group`, exigem a versão 8.0.2 ou posterior do mecanismo.

Quando o Amazon DocumentDB usa um escaneamento somente de índice

O Amazon DocumentDB considera uma verificação somente de índice para as seguintes operações quando um único índice contém todos os campos que a operação lê.

`Projeções de` achados cobertos

Uma `find` consulta se qualifica quando sua projeção é uma simples inclusão de um ou mais campos de nível superior e cada campo projetado está presente em um único índice. Por exemplo, com um índice composto ativado`{ customerId: 1, status: 1, total: 1 }`, a consulta a seguir é abordada. Todos os três campos projetados estão no índice e `_id` são explicitamente excluídos. (Cobrir uma projeção com mais de uma projeção que não seja de `_id` campo requer a versão 8.0.2 ou posterior do motor.)

```
db.orders.createIndex({ customerId: 1, status: 1, total: 1 })

db.orders.find(
  { customerId: 42 },
  { _id: 0, customerId: 1, status: 1, total: 1 }
)
```

O plano vencedor usa um `IXONLYSCAN` estágio e nunca retira os documentos correspondentes do armazenamento:

```
{
  "stage": "IXONLYSCAN",
  "indexName": "customerId_1_status_1_total_1",
  "direction": "forward",
  "indexCond": { "$and": [ { "customerId": { "$eq": 42 } } ] }
}
```

> **Note**
>
> Em uma `find` consulta coberta, os campos no resultado são retornados na ordem do índice (a ordem em que os campos aparecem na definição do índice), não na ordem em que foram listados na projeção ou na ordem em que são armazenados no documento. Isso corresponde ao comportamento do MongoDB para consultas cobertas. Em vez disso, uma consulta respondida por uma digitalização de coleção retorna os campos na ordem do documento armazenado.

Agregações **de `$group`** cobertas

Um `$group` estágio normalmente lê apenas alguns campos (a chave do grupo e as entradas para seus acumuladores), mas geralmente é executado sem uma entrelinha. `$match` Sem um índice de cobertura, o Amazon DocumentDB lê e decodifica cada documento completo na coleção apenas para inserir esses poucos campos no agrupamento.

Index-only escaneamentos de pipelines de agregação, incluindo `$group` estágios cobertos, estão disponíveis na versão 8.0.2 e posterior do mecanismo.

Quando um único índice contém todos os campos que o prefixo do pipeline lê, o Amazon DocumentDB pode colocar uma varredura somente de índice abaixo do grupo e ler esses campos diretamente do índice. Considere um funil que agrupa pedidos por status e soma seus totais, com um índice composto em: `{ status: 1, total: 1 }`

```
db.orders.createIndex({ status: 1, total: 1 })

db.orders.aggregate([
  { $group: { _id: "$status", revenue: { $sum: "$total" } } }
])
```

O pipeline lê somente `status` e`total`, ambos estão no índice, então o grupo é alimentado por uma varredura de coleção `IXONLYSCAN` em vez de uma varredura de coleção:

```
// 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)
```

Um líder`$match`,`$sort`,`$skip`, ou `$limit` antes do `$group` seja suportado. No entanto, todos os campos referenciados por esses estágios (por exemplo, um campo de `$match` filtro ou uma `$sort` chave) também devem estar presentes no mesmo índice para que o pipeline seja coberto. Se algum campo referenciado estiver ausente do índice, o Amazon DocumentDB retornará a uma coleta ou escaneamento de índice.

> **Note**
>
> Diferentemente de uma `find` projeção, `$group` a não inclui implicitamente o documento. `_id` O `_id` in a `$group` é a chave do grupo, não o identificador do documento, portanto, o índice não precisa conter o documento `_id` para cobrir o pipeline.

Requisitos para uma consulta coberta

Para que o Amazon DocumentDB use uma verificação somente de índice, todos os itens a seguir devem ser válidos:

- Um único índice contém todos os campos que a consulta lê. Isso inclui campos projetados e a chave do grupo, entradas do acumulador e quaisquer campos de filtro ou classificação de um pipeline coberto. `find` `$group` Se pelo menos um campo estiver ausente do índice, a consulta não poderá ser coberta.
- A projeção é uma inclusão de campos de nível superior (para`find`). Projeções de exclusão, campos computados e operadores de projeção, como `$slice` e `$elemMatch` não são elegíveis.
- O **`_id`** campo é contabilizado (para`find`). Como `_id` está incluído por padrão, um índice composto comum, como `{ a: 1, b: 1 }` não cobre`find({}, { a: 1, b: 1 })`: o implícito não `_id` está no índice. Exclua `_id` da projeção (`{ _id: 0, a: 1, b: 1 }`) ou adicione `_id` ao índice.
- O filtro está totalmente satisfeito com o índice. Se o Amazon DocumentDB não conseguir transformar parte do predicado da consulta em um limite de índice, ele manterá essa parte como um filtro residual, lerá o documento para avaliá-lo e não converterá o escaneamento em um

escaneamento somente de índice. Operadores como`$mod`, `$ne`, `$nin`, e `$exists` deixam um filtro residual.

- Nenhum campo projetado é uma coluna de várias teclas (matriz). Consulte Restrições.

Restrições

Os casos a seguir não podem usar uma varredura somente de índice e voltar para uma varredura normal de índice ou uma varredura de coleção. Esse é o comportamento esperado e nunca afeta a exatidão dos resultados.

- Campos com várias teclas (matriz). Se um campo já teve um valor de matriz em qualquer documento, sua coluna de índice é multichave. Um índice multichave armazena uma entrada por elemento de matriz, portanto, a matriz original não pode ser reconstruída a partir do índice. Qualquer consulta que leia esse campo não é abordada.
- Caminhos pontilhados ou aninhados. Uma projeção ou referência de campo em um caminho aninhado, como o, não `"address.city"` é elegível, mesmo que o índice contenha esse caminho.
- Projeções de exclusão, computadas ou de operadores. Projeções como`{ b: 0 }`,`{ total: { $add: ["$price", "$tax"] } }`, ou `{ a: { $slice: 5 } }` exigem o documento completo e não são abordadas.
- Whole-document referências em agregação. Uma `$group` referência ao documento inteiro (por exemplo,`{ $push: "$$ROOT" }`) não pode ser abordada, pois nenhum índice pode reconstruir todo o documento.
- Remodelando os estágios anteriores. **`$group`** Uma etapa que reformula documentos (como`$project`, `$addFields`, `$unwind`, ou`$lookup`) que aparecem antes do `$group` desqualifica o pipeline de ser coberto.
- Incompatibilidade de agrupamento em predicados de string. Se o agrupamento da consulta for diferente do agrupamento do índice, os predicados da string não poderão ser servidos pelo índice e deixarão um filtro residual, o que força a busca do documento. Os predicados numéricos e baseados em tipos não são afetados.

Confirmando uma verificação somente de índice

Use a `explain("executionStats")` saída para confirmar que uma consulta é atendida por uma digitalização somente de índice e que ela não está tocando documentos. Procure duas coisas:

- O plano vencedor usa o `IXONLYSCAN` palco.
- O `docsExamined` valor nesse estágio é `0` (e `totalDocsExamined` é`0`), o que significa que a consulta foi respondida inteiramente a partir do índice sem ler nenhum 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**
>
> Um `docsExamined` valor diferente de zero em um `IXONLYSCAN` palco significa que o índice continha os dados, mas o Amazon DocumentDB ainda verificou o armazenamento para confirmar se cada versão do documento estava visível. Isso pode acontecer quando a coleção tem gravações recentes cujas versões obsoletas do documento a coleta de lixo ainda não foi recuperada. Depois que a coleta de lixo processa esses documentos, `docsExamined` passa para`0`. Nesse ponto, a consulta obtém todos os benefícios do escaneamento coberto. Para obter mais informações sobre coleta de lixo, consulteColeta de resíduos no Amazon DocumentDB. Para obter mais informações sobre métricas do `docsExamined`, consulte Análise do plano de consulta.

Práticas recomendadas

- Crie índices compostos que contenham os campos nos quais suas consultas mais frequentes são projetadas ou agrupadas, para que essas consultas possam ser atendidas.
- `_id`Exclua das `find` projeções (`{ _id: 0, ... }`), a menos `_id` que faça parte do índice de cobertura.
- Evite campos de matriz de indexação (com várias chaves) quando você pretende que uma consulta seja abordada, pois as colunas de várias chaves desqualificam o caminho somente de índice.
- Permita que a coleta de lixo acompanhe sua carga de trabalho de gravação, para que as digitalizações cobertas ignorem o acesso aos documentos e permaneçam intactas. `docsExamined` `0` Para obter mais informações, consulte Coleta de resíduos no Amazon DocumentDB.

Planejar filtros de cache

Um filtro de cache de plano, também chamado de filtro de índice, restringe o conjunto de índices que o planejador de consultas pode considerar para uma forma de consulta específica. Quando uma consulta corresponde a uma forma que tem um filtro, o planejador escolhe um plano somente a partir dos índices nomeados nesse filtro, em vez de avaliar cada índice na coleção.

Uma forma de consulta é a combinação do predicado da consulta, da especificação de classificação e do namespace da coleção, com os valores do predicado normalizados. Duas consultas que diferem apenas em seus valores compartilham uma forma, portanto, um único filtro cobre todas elas. Na saída de`planCacheListFilters`, os valores normalizados são mostrados como`"@"`.

Um filtro de cache do plano oferece uma maneira de fixar o plano, para que você possa evitar escolher um índice mais lento devido à distribuição de dados ou às alterações no índice. Os filtros têm as seguintes propriedades:

- Nenhuma alteração no aplicativo é necessária. Os filtros são definidos com um comando de banco de dados e aplicados no servidor, para que você possa mitigar uma regressão sem uma implantação de código. Essa é a principal diferença de um`hint`, que deve ser adicionado em cada site de chamada.
- O escopo é uma forma de consulta. Uma forma é definida pela coleção, pelo predicado e pela classificação, não pela consulta completa, portanto, um filtro restringe somente as consultas

que correspondem a essa forma. Outras consultas sobre a mesma coleção continuam sendo planejadas normalmente.

- A mudança é durável. Os filtros persistem durante a reinicialização da instância e a correção do mecanismo. Descartar a coleção remove seus filtros.
- A mudança é reversível. Limpar o filtro retorna ao planejamento normal baseado em custos, portanto, um filtro é uma forma de baixo risco de estabilizar uma carga de trabalho enquanto você investiga a causa subjacente.

Tópicos

- Comandos compatíveis
- Formas de consulta suportadas
- Configurando, listando e limpando filtros
- Verificando se um filtro foi aplicado
- Exemplos
- Comportamento com dicas e alterações no índice

Comandos compatíveis

Os filtros de cache do plano exigem a versão 2.0 ou posterior do planejador. O comando ao qual um filtro pode ser aplicado depende da versão do planejador e da versão secundária do Amazon DocumentDB, conforme mostrado na tabela a seguir.

Suporte de comando para planejar filtros de cache

Command	Versão mínima do planejador	Versão secundária 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

Command	Versão mínima do planejador	Versão secundária mínima
`aggregate`	3.0	8.0.2

> **Note**
>
> A versão 3.0 do Planner está disponível somente no Amazon DocumentDB 8.0. Como os `aggregate` comandos `distinct` e exigem a versão 3.0 do planejador, os filtros de cache do plano para esses dois comandos estão disponíveis somente no Amazon DocumentDB 8.0. No Amazon DocumentDB 5.0, que oferece suporte à versão 2.0 do planejador, os filtros se aplicam aos `count` comandos`find`,`update`,`delete`, e. `findAndModify`

Para o `aggregate` comando, a forma é obtida da frente da tubulação: a tubulação `$match` fornece o filtro e a `$sort` imediatamente depois fornece a classificação. `$skipe` `$limit` não afetam a forma, nem os estágios além desse ponto.

O Amazon DocumentDB otimiza o pipeline antes de planejá-lo, e um filtro de cache do plano é comparado ao pipeline otimizado. O filtro e a classificação que o planejador vê podem, portanto, diferir dos estágios que você escreveu. Por exemplo, dois `$match` estágios adjacentes na frente de um pipeline são combinados em um único `$and` predicado:

```
db.orders.aggregate([
  { $match: { status: "SHIPPED" } },
  { $match: { region: "us-east-1" } },
  { $group: { _id: "$customerId", total: { $sum: "$amount" } } }
])
```

O planejador vê um `$match` ativado`{$and: [{status: ...}, {region: ...}]}`, então o filtro precisa ser definido nesse predicado combinado.

```
db.runCommand({planCacheSetFilter: "orders",
query: { $and: [ { status: "SHIPPED" }, { region: "us-east-1" } ] },
indexes: [ "status_1_region_1" ]})
```

Um filtro definido na coleção externa de um `$graphLookup` estágio `$lookup` ou conduz o índice usado para escanear essa coleção externa.

Formas de consulta suportadas

A partir do Amazon DocumentDB 5.0.2 e 8.0.2, os seguintes operadores podem aparecer em uma forma de consulta que tenha um filtro de cache de plano:

- `$text`. A string de pesquisa é normalizada, portanto, as consultas que diferem apenas nos termos pesquisados compartilham uma forma.
- `$near` e `$nearSphere`. Ambos os operadores se normalizam para a mesma forma. As coordenadas`$minDistance`, e não `$maxDistance` fazem parte da forma.
- `$geoWithin` e `$geoIntersects`. O tipo de geometria GeoJSON faz parte da forma, portanto, uma `Polygon` consulta e uma `MultiPolygon` consulta têm formas diferentes. As coordenadas não fazem parte da forma.
- `$regex`, quando usado diretamente em um campo. O padrão e as opções são normalizados, então todas as expressões regulares no mesmo campo compartilham a mesma forma.

A configuração de um filtro falha com o código de erro 303 para uma forma de consulta que usa`$jsonSchema`,`$expr`, ou`$sampleRate`. Ele também falha nas seguintes formas. Uma consulta que usa qualquer uma delas é planejada sem um filtro:

- Um `$regex` aninhado dentro`$in`,`$nin`, ou`$all`. Uma matriz que mistura expressões regulares com valores literais não pode ser avaliada em relação a um índice, portanto, um filtro nessa forma nunca poderia ser aplicado.
- Uma espécie de especificação de`{$natural: 1}`. Uma `$natural` classificação solicita uma varredura de coleção, o que é incompatível com a restrição do planejador a um índice.

Configurando, listando e limpando filtros

Para definir um filtro de cache do plano, use o `planCacheSetFilter` comando. Juntos, `sort` os campos `query` e especificam a forma da consulta e `indexes` listam os nomes dos índices que o planejador pode considerar para essa forma.

```
db.runCommand({
planCacheSetFilter: <collection>,
query: <query>,
sort: <sort>, // optional
indexes: [ <index1>, <index2>, ...],
comment: <any> // optional
```

```
})
```

Por exemplo, o comando a seguir restringe o planejador ao a_1 índice da forma `{a: {$eq: "@"}, b: {$eq: "@"}}` sem classificação:

```
db.runCommand({planCacheSetFilter: "foo",
query: { a: 1, b: 1 },
sort: {},
indexes: [ "a_1" ]})
```

Definir um filtro para uma forma que já tem um substitui o filtro existente. Depois de executar os dois comandos a seguir, o filtro na forma `{a: {$eq: "@"}, b: {$eq: "@"}}` é`["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 listar todos os filtros em uma coleção, use o `planCacheListFilters` comando:

```
db.runCommand({planCacheListFilters: <collection>})
```

A saída mostra cada forma armazenada com seus valores normalizados e sua 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
}
```

O operador é preservado e somente o valor é substituído por`"@"`. Uma igualdade implícita, como a, `{a: 1}` `{"a": {"$eq": "@"}}` está listada em sua forma explícita e os elementos de uma `$in` matriz são substituídos como um todo, como. `{"a": {"$in": "@"}}` Um conjunto de filtros sem classificação é listado com um `sort` documento vazio.

Para remover filtros, use o `planCacheClearFilters` comando.

```
db.runCommand({
planCacheClearFilters: <collection>,
query: <query pattern>, // optional
sort: <sort specification>, // optional
comment: <any> // optional
})
```

Para limpar todos os filtros da coleção, omita ambos `query` e`sort`:

```
db.runCommand({planCacheClearFilters: "foo"})
```

Fornecer ambos `query` e `sort` limpar somente o filtro que tem esse tipo exato. Os valores em `query` são normalizados, portanto, qualquer valor representativo funciona. Para limpar somente o filtro que foi definido sem uma classificação, passe um `sort` documento vazio:

```
db.runCommand({planCacheClearFilters: "foo", query: {a: 1}, sort: {a: 1}})

db.runCommand({planCacheClearFilters: "foo", query: {a: 1}, sort: {}})
```

Verificando se um filtro foi aplicado

A `explain` saída inclui dois campos que relatam o status do filtro de cache do plano:

- `indexFilterSet`é `true` quando existe um filtro na coleção para a forma da consulta.
- `indexFilterApplied`é `true` somente quando a consulta foi planejada usando um índice da lista desse filtro.

A partir do Amazon DocumentDB 5.0.2 e 8.0.2, esses campos são relatados para cada comando que suporta filtros de cache de plano e na saída do profiler para esses comandos. Nas versões secundárias anteriores, eles eram relatados apenas para o `find` comando. Para obter mais informações sobre como ler a saída de explicação, consulteAnálise do plano de consulta.

Um filtro pode corresponder a uma forma de consulta sem ser aplicado. Se nenhum dos índices na lista do filtro puder atender à consulta, o planejador planeja novamente sem o filtro e escolhe um índice de custo. Nesse caso, `indexFilterSet` é `true` e `indexFilterApplied` é`false`. Isso acontece quando os índices nomeados não cobrem os campos da consulta, quando os índices nomeados não existem ou quando uma `$regex` forma não pode produzir limites de índice, como um padrão não ancorado ou que não diferencia maiúsculas de minúsculas.

Exemplos

Os exemplos a seguir usam uma `orders` coleção e uma `customers` coleção para o `$lookup` exemplo com esses índices:

```
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
```

Exemplo: fixar uma **`consulta de`** busca em um índice composto

Considere uma consulta que filtra `status` e classifica por`orderDate`:

```
db.orders.find({ status: "SHIPPED" }).sort({ orderDate: 1 })
```

O índice composto `status_1_orderDate_1` satisfaz tanto o predicado quanto a classificação, portanto, nenhuma classificação separada é necessária. Se o planejador preferir`status_1`, a consulta precisará classificar seus resultados, o que adiciona um `SORT` estágio e fica mais caro à medida que o número de pedidos correspondentes aumenta. Para restringir o planejador ao índice composto dessa forma:

```
db.runCommand({planCacheSetFilter: "orders",
query: { status: "SHIPPED" },
sort: { orderDate: 1 },
indexes: [ "status_1_orderDate_1" ]})
```

Como os valores são normalizados fora da forma, esse filtro também cobre`{ status: "PENDING" }`,`{ status: "CANCELLED" }`, e todos os outros valores `status` com o mesmo tipo. Ele não cobre o mesmo predicado com um tipo diferente, ou sem classificação, que são formas separadas. Confirme se o filtro entrou em vigor com`explain`:

```
db.orders.find({ status: "SHIPPED" }).sort({ orderDate: 1 }).explain()
```

Os relatórios de saída `indexFilterSet: true` e`indexFilterApplied: true`, e os nomes `IXSCAN` artísticos`status_1_orderDate_1`.

Exemplo: fixar uma **`atualização em um`** índice seletivo

Uma atualização precisa localizar os documentos a serem modificados antes de poder gravá-los, e essa pesquisa usa um índice da mesma forma que uma leitura. Quando um predicado pode ser servido por mais de um índice, o índice escolhido pelo planejador determina quantos documentos a atualização examina.

```
db.orders.updateMany(
  { customerId: 4815, status: "PENDING" },
  { $set: { status: "CANCELLED" } }
)
```

Ambos `customerId_1` `status_1` podem servir a esse predicado. Para fixar a atualização em`customerId_1`:

```
db.runCommand({planCacheSetFilter: "orders",
query: { customerId: 4815, status: "PENDING" },
indexes: [ "customerId_1" ]})
```

Os filtros nos comandos de gravação exigem o Amazon DocumentDB 5.0.2 ou 8.0.2 e o planner versão 2.0 ou posterior. Verifique com uma `explain` das atualizações:

```
db.runCommand({explain: {update: "orders",
updates: [{ q: { customerId: 4815, status: "PENDING" },
             u: { $set: { status: "CANCELLED" } },
             multi: true }]}})
```

O plano vencedor é um `UPDATE` estágio `IXSCAN` em vez de um`customerId_1`. O mesmo filtro se aplica a `findAndModify` comandos `delete` e comandos que usam essa forma de consulta, porque a forma não inclui o tipo de gravação.

Exemplo: fixar um **`pipeline`** agregado e uma coleção estrangeira **`$lookup`**

Para uma agregação, a forma é retirada do `$match` palco principal e `$sort` imediatamente após ele. Estágios além desse, como `$group` ou`$project`, não fazem parte da forma.

```
db.orders.aggregate([
  { $match: { status: "SHIPPED" } },
  { $group: { _id: "$customerId", total: { $sum: "$amount" } } }
])
```

Um filtro definido no `$match` predicado direciona o índice usado para ler: `orders`

```
db.runCommand({planCacheSetFilter: "orders",
query: { status: "SHIPPED" },
indexes: [ "status_1" ]})
```

Os filtros também se aplicam à coleção à qual um `$graphLookup` estágio `$lookup` ou estágio se junta. O filtro é colocado na coleção externa, não na coleção contra a qual a tubulação passa. No pipeline a seguir, `orders` é a coleção externa e `customers` é a coleção externa:

```
db.orders.aggregate([
  { $lookup: { from: "customers",
               localField: "customerId",
               foreignField: "customerId",
               as: "customer" } }
])
```

A junção sonda `customers` uma vez por documento externo, então o índice escolhido é aplicado repetidamente e seu custo é multiplicado em todo o pipeline. Para fixar esse escaneamento em um índice específico, ative o filtro `customers` para a forma de igualdade da chave de junção:

```
db.runCommand({planCacheSetFilter: "customers",
query: { customerId: 1 },
indexes: [ "customerId_1" ]})
```

Como os valores são normalizados fora da forma, o valor 1 acima é um espaço reservado e qualquer valor produz o mesmo filtro. Quando um filtro de cobrança estrangeira é aplicado, o nível superior `indexFilterSet` e os `indexFilterApplied` campos são relatados`true`, e o índice selecionado pelo filtro aparece no lado externo do plano vencedor. Os filtros ativados `aggregate` exigem o Amazon DocumentDB 8.0.2 e a versão 3.0 do planejador.

Comportamento com dicas e alterações no índice

- Um filtro de cache de plano tem precedência sobre um`hint`. O Amazon DocumentDB aplica o filtro primeiro e depois aplica a dica na lista de índices permitidos do filtro. Se a dica nomear um índice permitido pelo filtro, esse índice será usado. Se a dica nomear um índice que o filtro não permite, a dica será ignorada e o planejador escolherá entre os índices permitidos de acordo com o custo. As duas formas de dica se comportam da mesma maneira, independentemente de você nomear o índice ou fornecer seu padrão de chave.

  Por exemplo, se a coleção `foo` tiver os índices, e `a_1` `b_1a_1_b_1`, e você definir um filtro na forma `{query: {a: {$eq: "@"}, b: {$eq: "@"}}, sort: {a: 1}}` com a lista de índices`["a_1", "a_1_b_1"]`, a execução `db.foo.find({ a: 10, b: 20 }).sort({a: 1}).hint({ a: 1 })` usaráa_1, pois ela é nomeada na dica e permitida pelo filtro.
- A eliminação de um índice não modifica os filtros. O filtro mantém o nome do índice. Embora esse índice esteja ausente, o planejamento o ignora. Se outro índice na lista do filtro puder servir à consulta, o filtro ainda será aplicado. Somente quando nenhum dos índices nomeados está disponível é que o `indexFilterApplied` relatório `false` é feito. Recriar um índice com o mesmo nome o torna elegível novamente.
- Eliminar uma coleção remove todos os filtros dessa coleção.

Práticas recomendadas do Amazon DocumentDB

Aprenda as práticas recomendadas para trabalhar com o Amazon DocumentDB (compatível com MongoDB). Essa seção é continuamente atualizada conforme novas melhores práticas são identificadas.

Tópicos

- Diretrizes operacionais básicas
- Dimensionamento de instância
- Trabalhar com índices
- Práticas recomendadas de segurança
- Otimização de custos
- Uso de métricas para identificar problemas de performance
- Workloads de TTL e séries temporais
- Migrações

- Trabalhar com grupos de parâmetros de cluster
- Consultas de pipeline de agregação
- BatchInsert e BatchUpdate

Diretrizes operacionais básicas

As diretrizes operacionais básicas a seguir devem ser seguidas por todos ao trabalhar com o Amazon DocumentDB. O Acordo de Nível de Serviço do Amazon DocumentDB exige que você siga essas diretrizes.

- Implante um cluster com duas ou mais instâncias do Amazon DocumentDB em duas ou mais zonas de disponibilidade da AWS . Para cargas de trabalho de produção, implante um cluster que consiste em três ou mais instâncias do Amazon DocumentDB em três zonas de disponibilidade.
- Use o serviço dentro dos limites de serviço indicados. Para obter mais informações, consulte Cotas do Amazon DocumentDB.
- Monitore sua memória, CPU, conexões e uso de armazenamento. Para ajudar você a manter o desempenho e a disponibilidade do sistema, configure CloudWatch a Amazon para notificá-lo quando os padrões de uso mudarem ou quando você se aproximar da capacidade de sua implantação.
- Escale suas instâncias quando estiver se aproximando dos limites de capacidade. Suas instâncias devem ser provisionadas com recursos computacionais suficientes (ou seja, RAM, CPU) para acomodar aumentos imprevistos na demanda de suas aplicações.
- Defina seu período de retenção de backup para alinhar com seu objetivo de ponto de recuperação.
- Teste o failover de seu cluster para entender quanto tempo o processo leva para seu caso de uso. Para obter mais informações, consulte Failover do Amazon DocumentDB.
- Conecte-se ao cluster do Amazon DocumentDB com o endpoint do cluster (consulte Endpoints do Amazon DocumentDB) e no modo de conjunto de réplicas (consulte Conectar-se ao Amazon DocumentDB como um conjunto de réplicas) para minimizar o impacto de um failover em sua aplicação.
- Escolha uma configuração de preferência de leitura do driver que maximize a escalabilidade de leitura, sem deixar de atender aos requisitos de consistência de leitura de sua aplicação. A preferência de leitura `secondaryPreferred` permite leituras de réplica e libera a instância primária para trabalhar mais. Para obter mais informações, consulte Opções de preferência de leitura.

- Projete sua aplicação para ser resistente no caso de erros de rede e banco de dados. Use o mecanismo de erro do driver para distinguir entre erros transitórios e persistentes. Repita os erros transitórios usando um mecanismo de recuo exponencial quando for apropriado. Certifique-se de que a sua aplicação considerará a consistência de dados ao implementar a lógica da nova tentativa.
- Habilite a proteção contra exclusão de todos os clusters de produção ou de qualquer cluster que tenha dados valiosos. Antes de excluir um cluster do Amazon DocumentDB, faça um snapshot final. Se você estiver implantando recursos com CloudFormation, ative a proteção contra encerramento. Para obter mais informações, consulte Proteção contra encerramento e exclusão.
- Ao criar um cluster do Amazon DocumentDB, a `--engine-version` é um parâmetro opcional que assume como padrão a versão mais recente do mecanismo principal. A versão padrão atual do mecanismo é 5.0.0 (observação: o Amazon DocumentDB 8.0 está disponível, mas deve ser explicitamente especificado). Quando novas versões principais do mecanismo forem lançadas, a versão padrão `--engine-version` será atualizada para refletir a última versão principal do mecanismo. Como resultado, para cargas de trabalho de produção, especialmente aquelas que dependem de scripts, automação ou CloudFormation modelos, especifique-as explicitamente `--engine-version` para a versão principal pretendida.

Dimensionamento de instância

Um dos aspectos mais importantes da escolha de um tamanho de instância no Amazon DocumentDB é o tamanho de RAM para seu cache. O Amazon DocumentDB reserva um terço da RAM para seus próprios serviços, o que significa que somente dois terços da RAM da instância estão disponíveis para o cache. Assim, é uma prática recomendada de desempenho do Amazon DocumentDB escolher um tipo de instância com RAM suficiente para colocar seu conjunto de trabalho (ou seja, dados e índices) na memória. Ter instâncias dimensionadas adequadamente ajudará a otimizar o desempenho geral e potencialmente minimizar os I/O custos. É possível usar a calculadora de dimensionamento terceirizada do Amazon DocumentDB para estimar o tamanho da instância para um workload específico.

Para determinar se o conjunto de trabalho do seu aplicativo cabe na memória, monitore o `BufferCacheHitRatio` uso da Amazon CloudWatch para cada instância em um cluster que está sob carga.

A `BufferCacheHitRatio` CloudWatch métrica mede a porcentagem de dados e índices fornecidos pelo cache de memória de uma instância (versus o volume de armazenamento). De modo geral, o valor de `BufferCacheHitRatio` deve ser o mais alto possível, pois a leitura de dados da memória

do conjunto de trabalho é mais rápida e econômica do que a leitura do volume de armazenamento. Embora seja desejável manter a `BufferCacheHitRatio` mais próxima de 100%, o melhor valor possível dependerá dos padrões de acesso e dos requisitos de desempenho da aplicação. Para manter a `BufferCacheHitRatio` mais alta possível, é recomendável que as instâncias do cluster sejam provisionadas com RAM suficiente para poder ajustar seus índices e conjunto de dados de trabalho na memória.

Se seus índices não couberem na memória, você verá uma `BufferCacheHitRatio` menor. A leitura contínua do disco gera I/O custos adicionais e não é melhor do que a leitura da memória. Se sua proporção `BufferCacheHitRatio` for menor do que o esperado, aumente o tamanho da instância do cluster para fornecer mais RAM para colocar os dados do conjunto de trabalho na memória. Se a ampliação da classe de instância resultar em um aumento drástico na `BufferCacheHitRatio`, o conjunto de trabalho da aplicação não coube na memória. Continue a expansão até que `BufferCacheHitRatio` não aumente mais consideravelmente após uma operação de escalabilidade. Para obter informações sobre como monitorar as métricas de instância, consulte Métricas do Amazon DocumentDB.

Dependendo dos requisitos de workload e latência, pode ser aceitável que a aplicação tenha valores de `BufferCacheHitRatio` mais altos durante o uso em estado estável, mas que `BufferCacheHitRatio` tenha períodos de queda periodicamente, pois consultas de análise que precisam verificar uma coleção inteira são executadas em uma instância. Esses períodos de queda na `BufferCacheHitRatio` podem se manifestar como latência maior para consultas subsequentes que precisam preencher novamente os dados do conjunto de trabalho do volume de armazenamento de volta para o cache de buffer. Teste suas cargas de trabalho em um ambiente de pré-produção com uma carga de trabalho de produção representativa primeiro para entender as características de desempenho e **`BufferCacheHitRatio`** antes de implantar a carga de trabalho na produção.

A `BufferCacheHitRatio` é uma métrica específica da instância, portanto instâncias diferentes dentro do mesmo cluster podem ter valores de `BufferCacheHitRatio` diferentes dependendo de como as leituras são distribuídas entre as instâncias principal e de réplica. Se o workload operacional não puder lidar com aumentos periódicos de latência de preencher o cache do conjunto de trabalho novamente após a execução de consultas de análise, tente isolar o cache de buffer do workload regular em relação ao das consultas de análise. O isolamento completo da `BufferCacheHitRatio` pode ser obtido direcionando consultas operacionais para a instância principal e consultas de análise somente para as instâncias de réplica. Também é possível obter isolamento parcial direcionando consultas de análise para uma instância de réplica específica, entendendo que um percentual de consultas regulares também será executado nessa réplica, podendo ser afetada.

Os valores de `BufferCacheHitRatio` adequados dependem do seu caso de uso e dos requisitos da aplicação. Não há nenhum valor melhor ou mínimo para essa métrica; somente é possível decidir se a compensação de uma `BufferCacheHitRatio` temporariamente mais baixa é aceitável de uma perspectiva de custo e desempenho.

Trabalhar com índices

Criação de índices

Ao importar dados para o Amazon DocumentDB, você deve criar seus índices antes de importar grandes conjuntos de dados. É possível usar a ferramenta de índice do Amazon DocumentDB para extrair índices de uma instância do MongoDB ou de um diretório `mongodump` em execução e criar esses índices em um cluster do Amazon DocumentDB. Para obter mais orientações sobre migrações, consulte Migração e atualização do Amazon DocumentDB.

Seletividade do índice

Limite a criação de índices aos campos em que o número de valores duplicados seja inferior a 1% do número total de documentos na coleção. Por exemplo, se sua coleção contiver 100.000 documentos, crie índices somente em campos em que o mesmo valor ocorrer 1000 vezes ou menos.

Escolher um índice com um alto número de valores exclusivos (ou seja, uma alta cardinalidade) garante que as operações de filtro retornem um pequeno número de documentos, gerando assim um bom desempenho durante as verificações de índice. Um exemplo de índice de alta cardinalidade é um índice exclusivo, que garante que predicados de igualdade retornem, no máximo, um documento. Exemplos de baixa cardinalidade incluem um índice sobre um campo booliano e um índice sobre o dia da semana. Devido a um desempenho insatisfatório, é improvável que índices de baixa cardinalidade sejam escolhidos pelo otimizador de consultas do banco de dados. Ao mesmo tempo, índices de baixa cardinalidade continuam consumindo recursos como espaço em disco e. I/Os Como regra geral, você deve direcionar índices a campos em que a frequência típica do valor é de 1%, ou menos, do tamanho total da coleção.

Além disso, é recomendável criar apenas índices em campos comumente usados como filtro e procurar regularmente índices não usados. Para obter mais informações, consulte Como analiso o uso do índice e identifico índices não utilizados?.

Impacto dos índices na gravação de dados

Embora os índices possam melhorar o desempenho da consulta evitando a necessidade de digitalizar todos os documentos em uma coleção, essa melhoria tem com uma compensação. Para

cada índice em uma coleção, sempre que um documento é inserido, atualizado ou excluído, o banco de dados deve atualizar a coleção e gravar os campos em cada um dos índices da coleção. Por exemplo, se uma coleção tiver nove índices, o banco de dados deverá executar dez gravações antes de confirmar a operação para o cliente. Assim, cada índice adicional gera latência de gravação adicional e aumento no armazenamento geral utilizado. I/O

As instâncias de cluster precisam ser dimensionadas adequadamente para manter toda a memória do conjunto de trabalho. Isso evita a necessidade de ler continuamente as páginas de índice do volume de armazenamento, o que afeta negativamente o desempenho e gera I/O custos mais altos. Para obter mais informações, consulte Dimensionamento de instância.

Para obter um melhor desempenho, minimize o número de índices em suas coleções, adicionando apenas os índices necessários para melhorar o desempenho de consultas comuns. Embora as workloads variem, uma boa orientação é manter cada coleção com cinco índices ou menos.

Identificar índices ausentes

Como melhor prática, identifique os índices ausentes regularmente. Para obter mais informações, consulte Como identifico índices ausentes?.

Identificar índices não utilizados

Como melhor prática, identifique e remova índices não utilizados regularmente. Para obter mais informações, consulte Como analiso o uso do índice e identifico índices não utilizados?.

Práticas recomendadas de segurança

Para obter as melhores práticas de segurança para o Amazon DocumentDB, consultePráticas recomendadas de segurança para o Amazon DocumentDB.

Otimização de custos

As melhores práticas a seguir podem ajudar você a gerenciar e minimizar os custos ao usar o Amazon DocumentDB. Para obter informações sobre preços, consulte Amazon DocumentDB (compatível com MongoDB) e as perguntas frequentes sobre o Amazon DocumentDB (compatível com MongoDB).

- Crie alertas de faturamento em limites de 50% e 75% de sua fatura esperada para o mês. Para obter mais informações sobre como criar alertas de faturamento, consulte Criar um alarme de faturamento.

- A arquitetura do Amazon DocumentDB separa armazenamento e computação, portanto, até mesmo um cluster de instância única é resiliente. O volume de armazenamento do cluster replica os dados de seis maneiras por três zonas de disponibilidade, proporcionando alta resiliência independentemente da quantidade de instâncias no cluster. Um cluster de produção típico tem três ou mais instâncias para fornecer alta disponibilidade. No entanto, é possível otimizar os custos usando um cluster de desenvolvimento de instância única quando a alta disponibilidade não é necessária.
- Para cenários de desenvolvimento e teste, interrompa um cluster quando ele não for mais necessário e inicie o cluster quando o desenvolvimento for retomado. Para obter mais informações, consulte Interromper e iniciar um cluster do Amazon DocumentDB.
- Tanto o TTL quanto os fluxos de alteração ocorrem quando I/O os dados são gravados, lidos e excluídos. Se você tiver habilitado esses recursos, mas não os estiver utilizando em sua aplicação, desabilitar os recursos pode ajudar a reduzir custos.

Uso de métricas para identificar problemas de performance

Tópicos

- Visualização de métricas de performance
- Configurando um CloudWatch alarme
- Avaliação de métricas de performance
- Avaliação do uso de instâncias do Amazon DocumentDB com métricas CloudWatch
- Ajuste das consultas

Para identificar problemas de desempenho causados por recursos insuficientes e outros gargalos comuns, é possível monitorar as métricas disponíveis para o seu cluster do Amazon DocumentDB.

Visualização de métricas de performance

É necessário monitorar as métricas de desempenho regularmente para ver os valores médio, máximo e mínimo de uma série de intervalos de tempo. Isso ajuda a identificar quando o desempenho está degradado. Você também pode definir CloudWatch alarmes da Amazon para limites métricos específicos para que você seja alertado se eles forem atingidos.

Para solucionar problemas de desempenho, é importante entender o desempenho de linha de base do sistema. Depois de configurar um novo cluster e executá-lo com um workload típica, capture os

valores médio, máximo e mínimo de todas as métricas de desempenho em diferentes intervalos (por exemplo, uma hora, 24 horas, 1 semana, 2 semanas). Isso dá a você uma ideia do que é normal. Isso ajuda a obter comparações para as horas de operação de pico e fora de pico. Você pode usar essas informações para identificar quando a performance está ficando abaixo dos níveis padrão.

Você pode visualizar as métricas de desempenho usando o Console de gerenciamento da AWS ou AWS CLI. Para obter mais informações, consulte Visualização CloudWatch de dados.

Configurando um CloudWatch alarme

Para definir um CloudWatch alarme, consulte Usando CloudWatch alarmes da Amazon no Guia do CloudWatch usuário da Amazon.

Avaliação de métricas de performance

Uma instância tem várias categorias diferentes de métricas. Como você determina valores aceitáveis depende dessa métrica.

CPU

- Utilização da CPU: o percentual da capacidade de processamento computacional utilizada.

Memória

- Memória disponível - quanto de RAM está disponível na instância.
- Uso de troca - Quanto espaço de troca é usado pela instância, em megabytes.

Input/output operações

- IOPS de leitura, IOPS de gravação – o número médio de operações de leitura ou gravação de disco por segundo.
- Latência de leitura, Latência de gravação – o tempo médio de uma operação de leitura ou gravação em milissegundos.
- Throughput de leitura, Throughput de gravação – o número médio de megabytes lido ou gravado no disco por segundo.
- Profundidade da fila de disco — O número de I/O operações que estão aguardando para serem gravadas ou lidas do disco.

Tráfego de rede

- Throughput de recepção de rede, Throughput de transmissão de rede - A taxa de tráfego de rede para e a partir da instância em megabytes por segundo.

Conexões de banco de dados

- Conexões de banco de dados - O número de sessões do cliente que estão conectadas à instância do banco de dados.

De um modo geral, os valores aceitáveis para as métricas de performance dependem do aspecto da linha de base e do que a aplicação está fazendo. Investigue variações consistentes ou tendenciais de sua linha de base.

Veja a seguir recomendações e instruções sobre os tipos específicos de métricas:

- Alto consumo de CPU? valores altos para o consumo de CPU podem ser adequados, desde que estejam de acordo com seus objetivos em relação à aplicação (como throughput ou concorrência). Se o seu consumo de CPU for consistentemente superior a 80%, considere dimensionar suas instâncias.
- Alto consumo de RAM — Se sua métrica `FreeableMemory`frequentemente fica abaixo de 10% da memória total da instância, considere escalar suas instâncias. Para obter mais informações sobre o que acontece quando sua instância do DocumentDB está com alta pressão de memória, consulte Amazon DocumentDB Resource Governance.
- Uso de troca - esta métrica deve permanecer em ou perto de zero. Se o uso de troca for significativo, considere dimensionar suas instâncias.
- Tráfego de rede: em relação ao tráfego de rede, fale com o administrador do sistema para entender qual throughput é esperado para sua rede de domínio e conexão com a internet. Inspecione o tráfego de rede caso o throughput seja consistentemente menor do que a esperada.
- Conexões do banco de dados - considere restringir as conexões do banco de dados caso perceba um alto número de conexões de usuários em conjunto com uma diminuição no desempenho da instância e no tempo de resposta. O melhor número de conexões de usuários para sua instância varia conforme a classe da instância e a complexidade das operações em execução. Para problemas com qualquer métrica de performance, uma das primeiras coisas que é possível fazer para melhorar a performance é ajustar as consultas mais utilizadas e mais caras para ver se isso reduz a pressão sobre os recursos do sistema.

Se suas consultas estiverem ajustadas e o problema persistir, considere atualizar sua classe de instância do Amazon DocumentDB para uma com mais recursos (CPU, RAM, espaço em disco, largura de banda de rede, I/O capacidade) relacionados ao problema que você está enfrentando.

Avaliação do uso de instâncias do Amazon DocumentDB com métricas CloudWatch

Você pode usar CloudWatch métricas para monitorar a taxa de transferência da instância e descobrir se a classe da instância fornece recursos suficientes para seus aplicativos. Para obter informações sobre suas cotas de classe de instância, consulte Cotas de instâncias e localize as especificações da sua classe de instância para encontrar o desempenho da sua rede.

Se o uso da sua instância estiver próximo do limite da classe da instância, a performance poderá começar a diminuir. As CloudWatch métricas podem confirmar essa situação para que você possa planejar a expansão manual para uma classe de instância maior.

Combine os seguintes valores de CloudWatch métricas para descobrir se você está se aproximando do limite da classe de instância:

- NetworkThroughput—A quantidade de taxa de transferência de rede recebida e transmitida pelos clientes para cada instância no cluster Amazon DocumentDB. Esse valor de throughput não inclui o tráfego de rede entre instâncias no cluster e o volume do cluster.
- StorageNetworkThroughput—A quantidade de taxa de transferência de rede recebida e enviada para o volume de armazenamento do cluster Amazon DocumentDB por cada instância no cluster Amazon DocumentDB.

Adicione o NetworkThroughput ao StorageNetworkThroughput para encontrar a taxa de transferência de rede recebida e enviada para o volume de armazenamento do cluster Amazon DocumentDB por cada instância em seu cluster Amazon DocumentDB. O limite da classe de instância para sua instância deve ser maior do que a soma dessas duas métricas combinadas.

É possível usar as seguintes métricas para analisar detalhes adicionais do tráfego de rede das aplicações clientes ao enviar e receber:

- NetworkReceiveThroughput—A quantidade de taxa de transferência de rede recebida dos clientes por cada instância no cluster Amazon DocumentDB. Esse throughput não inclui o tráfego de rede entre instâncias no cluster e o volume de armazenamento do cluster.
- NetworkTransmitThroughput—A quantidade de taxa de transferência de rede enviada aos clientes por cada instância no cluster Amazon DocumentDB. Esse throughput não inclui o tráfego de rede entre instâncias no cluster e o volume de armazenamento do cluster.

- StorageNetworkReceiveThroughput—A quantidade de taxa de transferência de rede recebida do volume de armazenamento do cluster Amazon DocumentDB por cada instância no cluster.
- StorageNetworkTransmitThroughput—A quantidade de taxa de transferência de rede enviada ao volume de armazenamento do cluster Amazon DocumentDB por cada instância no cluster.

Adicione todas essas métricas para avaliar como o uso da rede se compara ao limite da classe da instância. O limite da classe de instância deve ser maior do que a soma dessas métricas combinadas.

Os limites de rede e a utilização da CPU de uma instância são mútuos. Quando o throughput de rede aumenta, a utilização da CPU também aumenta. O monitoramento do uso da CPU e da rede fornece informações sobre como e por que os recursos estão sendo esgotados.

Para ajudar a minimizar o uso da rede, você pode considerar:

- Usar uma classe de instância maior.
- Dividir as solicitações de gravação em lotes para reduzir o total de transações.
- Direcionar a workload somente leitura para uma instância somente leitura.
- Excluir todos os índices não utilizados.

Ajuste das consultas

Um dos melhores jeitos de melhorar o desempenho do cluster é ajustar as consultas mais utilizadas e que requerem mais recursos para baixar o custo de operação delas.

É possível usar o profiler (consulte Definir o perfil das operações do Amazon DocumentDB) para registrar o tempo de execução e detalhes das operações executadas em seu cluster. O Profiler é útil para monitorar as operações mais lentas em seu cluster para ajudar você a melhorar o desempenho de consultas individuais e o desempenho geral do cluster.

Também é possível usar o comando `explain` para aprender a analisar um plano de consulta para uma consulta específica. É possível usar essas informações para modificar uma consulta ou uma coleção subjacente a fim de melhorar o desempenho da consulta (por exemplo, adicionar um índice).

Workloads de TTL e séries temporais

A exclusão de documentos resultante da expiração do índice TTL é um processo de best effort. Não há garantia de que os documentos serão excluídos dentro de um período específico. Fatores

como tamanho de instância, utilização de recursos de instância, tamanho do documento, throughput geral, número de índices e se os índices e o conjunto de trabalho cabem na memória podem afetar o momento em que os documentos expirados são excluídos pelo processo TTL.

Quando o monitor TTL exclui seus documentos, cada exclusão gera I/O custos, o que aumenta sua fatura. Se a taxa de transferência e as taxas de exclusão de TTL aumentarem, você deve esperar uma fatura maior devido ao aumento I/O do uso. No entanto, se você não criar um índice TTL para excluir documentos, mas sim segmentar documentos em coleções com base no tempo e simplesmente descartar essas coleções quando não forem mais necessárias, você não incorrerá em nenhum custo de E/S. Isso pode ser significativamente mais econômico do que usar um índice TTL.

Para cargas de trabalho de séries temporais, você pode considerar a criação de coleções contínuas em vez de um índice TTL, pois as coleções contínuas podem ser uma maneira melhor e menos intensiva de excluir dados. I/O Se você tiver coleções grandes (especialmente coleções acima de 1 TB) ou se I/O os custos de exclusão de TTL forem uma preocupação, particione os documentos em coleções com base no tempo e descarte as coleções quando os documentos não forem mais necessários. É possível criar uma coleção por dia ou uma por semana, dependendo da taxa de ingestão de dados. Embora os requisitos variem dependendo da aplicação, uma boa regra é ter mais coleções menores em vez de algumas coleções grandes. A eliminação dessas coleções não gera I/O custos e pode ser mais rápida e econômica do que usar um índice TTL.

Migrações

Como melhor prática para migrar dados para o Amazon DocumentDB, primeiro crie seus índices no Amazon DocumentDB e depois migre seus dados. Criar os índices primeiro pode reduzir o tempo geral e aumentar a velocidade da migração. Para fazer isso, é possível usar a Ferramenta de índice do Amazon DocumentDB. Para obter mais informações sobre migrações, consulte o. Migração e atualização do Amazon DocumentDB

Também recomendamos que, antes de migrar seu banco de dados de produção, aplique uma prática recomendada de testar totalmente sua aplicação no Amazon DocumentDB, levando em consideração a funcionalidade, o desempenho, as operações e o custo.

Trabalhar com grupos de parâmetros de cluster

Experimente as alterações do grupo de parâmetros do cluster em um cluster de teste antes de aplicar as alterações aos seus clusters de produção. Para obter informações sobre o backup do cluster, consulte Backup e restauração no Amazon DocumentDB.

Consultas de pipeline de agregação

Ao criar uma consulta de pipeline de agregação com vários estágios e avaliar apenas um subconjunto de dados na consulta, use o estágio `$match` como o primeiro estágio ou no início do pipeline. Usar `$match` primeiro reduz o número de documentos que as etapas subsequentes dentro da consulta de pipeline de agregação precisarão processar, melhorando assim o desempenho da consulta.

`BatchInsert` e BatchUpdate

Ao realizar uma alta taxa de `batchInsert` and/or `batchUpdate` operações simultâneas e a quantidade de `FreeableMemory` (CloudWatch métrica) chegar a zero em sua instância primária, você pode reduzir a simultaneidade da carga de trabalho de inserção ou atualização do lote ou, se a simultaneidade da carga de trabalho não puder ser reduzida, aumentar o tamanho da instância para aumentar a quantidade de. `FreeableMemory`

Gerenciar índices do Amazon DocumentDB

Tópicos

- Criação do índice do Amazon DocumentDB
- Manutenção de índices do Amazon DocumentDB

Criação do índice do Amazon DocumentDB

A criação de índices no Amazon DocumentDB exige que várias decisões sejam tomadas:

- Com que rapidez ele precisa ser concluído?
- A coleção pode ficar inacessível enquanto a construção está ocorrendo?
- Quanto da capacidade computacional de uma instância pode ser alocada para a compilação?
- Que tipo de índice deve ser criado?

Esta seção ajuda você a responder a essas perguntas e fornece os comandos e exemplos de monitoramento para criar um índice Amazon DocumentDB em sua coleção de clusters baseada em instâncias.

Diretrizes

As diretrizes a seguir incluem limites básicos e compensações de configuração ao criar novos índices:

- Suporte à versão do Amazon DocumentDB - Embora a indexação de um único trabalhador seja suportada em todas as versões do Amazon DocumentDB, a indexação de vários trabalhadores é suportada somente nas versões 4.0, 5.0 e 8.0 do Amazon DocumentDB.
- Compensação de desempenho - Aumentar o número de trabalhadores no processo de criação do índice aumenta a utilização da CPU e a IO de leitura na instância primária do seu banco de dados Amazon DocumentDB. Os recursos necessários para criar um novo índice não estarão disponíveis para sua workload em execução.
- Clusters elásticos: a indexação paralela não tem suporte nos clusters elásticos do Amazon DocumentDB.
- Clusters sem servidor - A indexação paralela em clusters sem servidor exige uma capacidade mínima de pelo menos 32 DCUs. Abaixo desse limite, o `workers` parâmetro é aceito, mas a construção do índice é executada com um único trabalhador. O número máximo de trabalhadores paralelos é dimensionado de acordo com sua capacidade mínima configurada: aproximadamente `min_DCU / 8` trabalhadores estão disponíveis. Por exemplo, em 32 DCUs você pode usar até 4 trabalhadores e em 64 DCUs você pode usar até 8 trabalhadores.

  > **Note**
  >
  > É necessário reiniciar o cluster após alterar a configuração de capacidade mínima para que o novo limite de trabalhadores paralelos entre em vigor.

- Máximo de trabalhadores (com base na instância) - O número máximo de trabalhadores que você pode configurar depende do tamanho da sua instância primária no cluster de banco de dados. É metade do número total de vCPUs na instância primária do seu cluster de banco de dados. Por exemplo, você pode executar no máximo 32 trabalhadores em uma instância db.r6g.16xlarge que tenha 64 vCPUs.

  > **Note**
  >
  > Trabalhadores paralelos só são suportados em instâncias 2xlarge e superiores.

- Mínimo de trabalhadores - O número mínimo de trabalhadores que você pode configurar é um. A configuração padrão para criação de índices em clusters baseados em instâncias é de dois trabalhadores. No entanto, você pode reduzir o número de trabalhadores para um usando a opção “threads de trabalhadores”. Isso executará o processo com um único trabalhador.
- Compactação de índice - O Amazon DocumentDB não oferece suporte à compactação de índices. O tamanho dos dados para dados e índices armazenados poderá ser maior do que ao usar outras opções.
- Indexação de várias coleções - Metade das vCPUs na instância primária do seu cluster de banco de dados pode ser usada para trabalhadores configurados que realizam a criação de índices em várias coleções.
- Tipos de índice — Para obter uma explicação completa dos tipos de índices compatíveis no Amazon DocumentDB, consulte Como indexar no Amazon DocumentDB no blog do AWS banco de dados.

Introdução

Para iniciar a criação de índices em uma coleção, use o comando `createIndexes`. Por padrão, o comando executará dois trabalhadores paralelos, o que aumenta a velocidade do processo de criação do índice em duas vezes.

Por exemplo, o processo de comando a seguir demonstra como criar um índice para o campo “user_name” em um documento e aumentar a velocidade do processo de indexação para quatro trabalhadores:

1. Crie índices usando dois trabalhadores paralelos no cluster:

```
db.runCommand({"createIndexes":"test","indexes":[{"key": {"user_name":1},
 "name":"username_idx"}]})
```

2. Para otimizar a velocidade do processo de criação do índice, você pode especificar o número de operadores usando a opção “threads de trabalhadores” (`"workers":<number>`) no comando `db.runCommand createIndexes`.

   Aumente a velocidade do processo para quatro trabalhadores paralelos:

```
db.runCommand({"createIndexes":"test","indexes":[{"key": {"user_name":1},
 "name":"username_idx", "workers":4}]})
```

> **Note**
>
> Quanto maior o número de trabalhadores, mais rápido a criação do índice progride. No entanto, quanto maior o número de trabalhadores, maior o aumento da carga nas vCPUs e na E/S de leitura da sua instância primária. Garanta que seu cluster esteja suficientemente provisionado para lidar com o aumento da carga sem degradar outras workloads.

Status do andamento da indexação

O processo de criação de índices funciona inicializando, escaneando coleções, classificando chaves e, finalmente, inserindo chaves por meio de um construtor de índices. O processo tem até seis estágios quando você o executa em primeiro plano e até nove estágios quando você o executa em segundo plano. É possível visualizar métricas de status, como percentual de conclusão, número total de blocos de armazenamento digitalizados, chaves classificadas e chaves inseridas, etapa por etapa.

Monitore o andamento no processo de indexação usando o comando `db.currentOp()` no shell mongo. Uma conclusão de 100% da última etapa mostra que todos os índices foram criados com sucesso:

```
db.currentOp({"command.createIndexes": { $exists : true } })
```

> **Note**
>
> Só há suporte para a visualização do status do andamento da indexação no Amazon DocumentDB 5.0.

Tipos de criação de índice

Os quatro tipos de criação de índices são:

- Primeiro plano: a construção de índices em primeiro plano bloqueia todas as outras operações do banco de dados até que o índice seja criado. A construção em primeiro plano do Amazon DocumentDB é composta por cinco estágios.

- Primeiro plano (exclusivo): a construção de índices de primeiro plano de um único documento (exclusivo) bloqueiam outras operações de banco de dados, como construções regulares de primeiro plano. Ao contrário da construção básica em primeiro plano, a compilação exclusiva usa um estágio adicional (chaves de classificação 2) para procurar chaves duplicadas. A construção em primeiro plano (exclusiva) é composta por seis estágios.
- Plano de fundo: a construção de índices no plano de fundo permite que outras operações do banco de dados sejam executadas em primeiro plano enquanto o índice está sendo criado. A construção em segundo plano do Amazon DocumentDB é composta por oito estágios.
- Plano de fundo (exclusivo): a construção de índices em plano de fundo de um único documento (exclusivo) permitem que outras operações do banco de dados sejam executadas em primeiro plano enquanto o índice está sendo criado. Ao contrário da construção básica em segundo plano, a compilação exclusiva usa um estágio adicional (chaves de classificação 2) para procurar chaves duplicadas. A construção de fundo (única) é composta por nove estágios.

Etapas de construção do índice

Estágio	Primeiro plano	Primeiro plano (exclusivo)	Contexto	Plano de fundo (exclusivo)
Inicializando	1	1	1	1
índice de construção: inicializando	2	2	2	2
índice de construção: varredura da coleção	3	3	3	3
índice de construção: chaves de classificação 1	4	4	4	4
índice de construção:		5		5

Estágio	Primeiro plano	Primeiro plano (exclusivo)	Contexto	Plano de fundo (exclusivo)
chaves de classificação 2				
índice de construção: inserindo chaves	5	6	5	6
validação: índice de digitalização			6	7
validação: classificação de tuplas			7	8
validação: coleta de digitalização			8	9

- inicialização - createIndex está preparando o construtor de índices. Essa fase deve ser muito breve.
- índice de construção: inicialização - O criador de índices está se preparando para criar o índice. Essa fase deve ser muito breve.
- índice de construção: coleção de varredura - O construtor de índices está realizando uma varredura de coleção para coletar chaves de índice. A unidade de medida é “blocos”.

> **Note**
>
> Se mais de um trabalhador estiver configurado para a criação do índice, ele será exibido nesse estágio. O estágio de “coleta de digitalização” é o único estágio que usa vários trabalhadores durante o processo de criação do índice. Todos os outros estágios exibirão um único trabalhador.

- índice de construção: chaves de classificação 1 - O construtor de índices está classificando as chaves de índice coletadas. A unidade de medida é “chaves”.

- índice de construção: chaves de classificação 2 - O construtor de índices está classificando as chaves de índice coletadas que correspondem às tuplas mortas. Essa fase existe apenas para a criação de índices exclusivos. A unidade de medida é “chaves”.
- índice de construção: inserindo chaves - O criador de índices está inserindo chaves de índice no novo índice. A unidade de medida é “chaves”.
- validando: índice de varredura - createIndex está examinando o índice para encontrar chaves que precisam ser validadas. A unidade de medida é “blocos”.
- validando: classificando tuplas - createIndex está classificando a saída da fase de varredura do índice.
- validando: escaneando a coleção - CreateIndex está escaneando a coleção para validar as chaves de índice encontradas nas duas fases anteriores. A unidade de medida é “blocos”.

Exemplo de saída de compilação de índice

No exemplo de saída a seguir (construção do índice em primeiro plano), o status da criação do índice é mostrado. O campo “msg” resume o andamento da compilação indicando o estágio e o percentual de conclusão da construção. O campo “trabalhadores” indica o número de trabalhadores usados durante esse estágio da criação do índice. O campo “andamento” mostra os números reais usados para calcular o percentual de conclusão.

> **Note**
>
> Os campos “currentIndexBuildName”, “msg” e “progress” não são suportados no Amazon DocumentDB versão 4.0.

```
{
    "inprog" : [{
    …
        "command": {
            "createIndexes": "test",
            "indexes": [{
                "v": 2,
                "key": {
                    "user_name": 1
                },
                "name": "user_name_1"
            }],
```

```
            "lsid": {
                "id": UUID("094d0fba-8f41-4373-82c3-7c4c7b5ff13b")
            },
            "$db": "test"
        },
        "currentIndexBuildName": user_name_1,
        "msg": "Index Build: building index number_1, stage 6/6 building index:
 656860/1003520 (keys) 65%",
        "workers": 1,
        "progress": {
            "done": 656861,
            "total": 1003520
        },
    …
    ],

    "ok" : 1
}
```

Manutenção de índices do Amazon DocumentDB

Tópicos

- Sobrecarga de índices
- Manutenção do índice usando reIndex

Sobrecarga de índices

O Amazon DocumentDB usa o Controle de Multi-Version Concorrência (MVCC) para gerenciar transações simultâneas. Quando os documentos são excluídos ou atualizados, suas versões anteriores permanecem em coleções e índices como versões "inativas". O processo de coleta de resíduos recupera automaticamente o espaço dessas versões inativas para futuras operações.

A sobrecarga de índices ocorre quando os índices de uma coleção se tornam maiores devido ao acúmulo de entradas de índice inativas ou obsoletas ou à fragmentação nas páginas. O percentual relatado representa a quantidade de espaço de índices que pode ser usada por futuras entradas de índice. Essa sobrecarga consome espaço no cache e no armazenamento do buffer. Se você quiser remover a sobrecarga, precisará reconstruir os índices.

Example Exemplo

Execute o comando apresentado a seguir para determinar o armazenamento não utilizado do seu índice:

```
db.coll.aggregate({$indexStats:{}});
```

O resultado deve ser semelhante a este:

```
{
    "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
    }
}
```

É possível recriar índices sem tempo de inatividade usando o comando `reIndex`, que exige uma verificação de toda a coleção. Consulte Manutenção do índice usando `reIndex`.

Manutenção do índice usando **`reIndex`**

`reIndex` é um comando usado para reconstruir um índice. Geralmente é usado quando um índice se torna corrompido ou ineficiente. Com o tempo, os índices podem acumular espaço não utilizado devido a muitas atualizações, inserções ou exclusões, levando à degradação do desempenho. A reindexação ajuda a remover esse espaço não utilizado e a restaurar a eficiência do índice.

Diretrizes do ReIndex

- `reIndex` é atualmente compatível somente com Amazon DocumentDB 5.0.
- O Amazon DocumentDB oferece suporte a `reindex` de um único índice em segundo plano, permitindo vários operadores. O índice antigo pode ser usado por consultas quando o processo `reIndex` está em execução.
- O Amazon DocumentDB oferece suporte à indexação de relatórios de andamento por meio de `currentOp`. É possível ver estágios de criação de índice semelhantes ao Etapas de construção do índice visualizado durante a criação do índice. A única diferença é que `reIndex` sempre tem oito estágios, independentemente de ser único ou não. Não há o estágio “criação de índice: chaves de classificação 2”.
- `reIndex` pode ser executado simultaneamente com qualquer comando, exceto comandos relacionados a índices na mesma coleção: `createIndexes`, `dropIndexes`, `collMod` e `renameCollection`.
- `reIndex` atualmente não é compatível com índices textuais, geoespaciais, vetoriais e parciais.

Construção de reindexação

Use o seguinte comando para recriar seu índice:

```
db.runCommand({ reIndex: "collection-name", index: "index-name"})
```

Opcionalmente, você também pode controlar o número de operadores atribuídos ao processo de reconstrução:

```
db.runCommand({ reIndex: "collection-name", index: "index-name", workers: number })
```

Gerenciamento da compactação de documentos a nível de coleção

A compactação de documentos em nível de coleção do Amazon DocumentDB 5.0 permite que você reduza os custos de armazenamento e I/O ao compactar os documentos em suas coleções. É possível ativar a compactação de documentos a um nível de coleção e visualizar as métricas de compactação conforme necessário, medindo os ganhos de armazenamento por meio de métricas de compactação, como tamanho de armazenamento de documentos compactados e status de compactação. O Amazon DocumentDB 5.0 usa o algoritmo de compactação LZ4 para compactar documentos.

O Amazon DocumentDB oferece suporte à compactação de documentos a partir da versão 5.0. A seguir estão as funções de compactação de documentos em nível de coleção:

- Comportamento padrão — A configuração de compactação padrão para novas coleções em um cluster 5.0 é determinada pelo parâmetro clusterdefault_collection_compression. Esse parâmetro é definido como “desativado” por padrão.
- Compactação de coleções existentes: a configuração de compactação de coleções existentes pode ser alterada usando o comando collMod.
- Alteração do limite de compactação: o limite de compactação padrão é de 2 KB. Esse valor pode ser especificado para novas coleções usando o comando createCollection e alterado para coleções existentes usando o comando collMod.

> **Note**
>
> A compactação de documentos do Amazon DocumentDB não tem suporte nas versões 3.6 e 4.0 do Amazon DocumentDB.

Tópicos

- Gerenciamento da compactação de documentos
- Monitoramento da compactação de documentos

Gerenciamento da compactação de documentos

Ativar a compactação de documentos em uma coleção

Ative a compactação de documentos ao criar uma coleção no Amazon DocumentDB 5.0 usando o db.createCollection() método:

```
db.createCollection( sample_collection,{
    storageEngine : {
        documentDB: {
            compression:{enable: <true | false>}
        }
    }
})
```

Ativar a compactação de documentos em um cluster

A compactação de documentos pode ser ativada por padrão para todas as novas coleções em um nível de cluster definindo o parâmetro de cluster `default_collection_compression` como “ativado”. Quando esse parâmetro é definido como “ativado”, as coleções recém-criadas no cluster terão a compactação ativada por padrão com um limite de compactação de 2 KB.

Compactar coleções existentes

Você também pode modificar as configurações de compactação de uma coleção existente usando a operação `collMod` e especificando a configuração `storageEngine` a seguir. Observe que a alteração feita usando esse comando se aplicará somente a inserted/updated documentos novos e a compactação em documentos inseridos anteriormente não será alterada.

```
db.runCommand({
    collMod: "orders",
    storageEngine: {
        documentDB: {compression: {enable: <true | false>} }
    }
})
```

Definir os limites de compactação

Por padrão, o limite de compactação para coleções compactadas é de 2032 bytes. Esse valor limite pode ser definido no comando `createCollection` ao criar uma nova coleção com a compactação ativada:

```
db.createCollection( sample_collection, {
    storageEngine : {
        documentDB: {
            compression: {
                enable: true,
                threshold: <128 - 8000>
            }
        }
    }
})
```

Você também pode modificar o limite de compactação de uma coleção compactada existente usando a operação `collMod` e especificando a seguinte configuração `storageEngine`:

```
db.runCommand({
    collMod: "orders",
    storageEngine: {
        documentDB: {
            compression: {
                enable: true,
                threshold: <128 - 8000>
            }
        }
    }
})
```

Observe que o limite de compactação só pode ser definido para um valor entre 128 e 8000 bytes. Além disso, a opção `enable` precisa ser definida como “verdadeira” ao especificar o limite de compactação.

Monitoramento da compactação de documentos

É possível verificar se uma coleção está compactada e calcular sua taxa de compactação da seguinte maneira.

Visualize as estatísticas de compressão executando o comando `db.printCollectionStats()` ou `db.collection.stats()` no shell do mongo. A saída mostra o tamanho original e o tamanho compactado, que é possível comparar para analisar os ganhos de armazenamento da compactação de documentos. Neste exemplo, são mostradas as estatísticas de uma coleção chamada “sample_collection”: Um fator de escala de 1024*1024 é usado abaixo para gerar os valores `size` e `storageSize` em MB.

```
db.sample_collection.stats(1024*1024)
```

Veja a seguir um exemplo de saída do comando acima:

```
{
    "ns" : "test.sample_collection",
    "count" : 1000000,
    "size" : 3906.3,
    "avgObjSize" : 4096,
    "storageSize" : 1953.1,
    compression:{"enabled" : true,"threshold" : 2032},
    ...
```

```
}
```

- tamanho - O tamanho original da coleção de documentos.
- avg ObjSize - O tamanho médio do documento antes da compactação arredondado para a primeira casa decimal. A unidade de medida é bytes.
- StorageSize - O tamanho de armazenamento da coleção após a compactação. A unidade de medida é bytes.
- habilitada - Indica se a compactação está habilitada ou desabilitada.

Para calcular a taxa de compressão real, divida o tamanho da coleção pelo tamanho do armazenamento (size/storageSize). Para o exemplo acima, o cálculo é 3906. 3/1953.1 que se traduz em uma taxa de compressão de 2:1.

Gerenciando a compactação baseada em dicionário no Amazon DocumentDB 8.0

O Amazon DocumentDB 8.0 apresenta um novo algoritmo de compactação de documentos, zstd, como uma alternativa aprimorada ao LZ4. Quando você ativa a compactação de dicionário em uma coleção do Amazon DocumentDB 8.0 selecionando Zstd como algoritmo de compactação, os documentos da sua coleção são amostrados para criar um dicionário de compactação personalizado. Documentos novos e atualizados são compactados usando esse dicionário e o algoritmo zstd. Essa abordagem geralmente atinge taxas de compactação mais altas do que os métodos de compactação padrão, especialmente para coleções com esquemas de documentos consistentes ou nomes de campo repetidos.

O Lz4 é um algoritmo projetado para compressão e descompressão rápidas. É conhecido por ser leve na CPU e, ao mesmo tempo, obter uma compressão perceptível. O Zstd é um algoritmo de uso geral que, com configurações padrão, usa mais CPU, mas atinge melhores taxas de compressão do que o lz4. O uso de dicionários melhora ainda mais a compactação da maioria dos documentos JSON. Alguns benefícios do algoritmo Zstd são:

- Custos de armazenamento reduzidos: taxas de compressão mais altas significam menos uso de armazenamento e custos mais baixos.
- Inferior I/O: documentos compactados exigem menos I/O, potencialmente melhorando o desempenho.

- Otimizado para sua coleção: o dicionário é treinado especificamente para os padrões de dados da sua coleção.

> **Note**
>
> Dictionary-based a compactação não é suportada nas versões 3.6, 4.0 e 5.0 do Amazon DocumentDB.

Considerações sobre a performance

A compactação Zstd envolve as seguintes compensações:

- Armazenamento versus CPU: a compactação Zstd atinge melhores taxas de compactação, mas pode usar um pouco mais de recursos da CPU em comparação com a compactação LZ4.
- Compressão inicial: novas coleções podem não atingir a compactação ideal até que documentos suficientes sejam inseridos para treinar um dicionário eficaz. Atualmente, um dicionário é treinado se a coleta de pelo menos 100 documentos.
- Tipo de carga de Read-intensive trabalho: cargas de trabalho em que todos os dados cabem no cache de buffer podem sofrer um aumento na latência e no uso da CPU devido à sobrecarga de descompressão.

A compressão Zstd é particularmente eficaz para coleções com documentos pequenos, matrizes de documentos e nomes de campos repetidos.

Habilitando a compactação baseada em dicionário

Para novas coleções, você pode usar o comando abaixo para ativar a compactação Zstd:

```
db.createCollection("myCollection",
    {
        storageEngine: {
            documentDB: {
                compression: {
                    enable: true,
                    algorithm: "zstd"
                }
```

```
            }
        }
    }
)
```

Você também pode ativar ou modificar a compactação em uma coleção existente:

```
db.runCommand({
    collMod: "myCollection",
    storageEngine: {
        documentDB: {
            compression: {
                enable: true,
                algorithm: "zstd"
            }
        }
    }
})
```

Para ativar o algoritmo Zstd em todas as coleções em seu cluster, você pode modificar o grupo de parâmetros do cluster para selecionar “zstd” como o valor do parâmetro “default_collection_compression”.

Conceitos básicos

O Amazon DocumentDB 8.0 vem com a compactação Zstd ativada por padrão. Você sempre pode desativá-lo definindo o valor de 'default_compression' como desativado no grupo de parâmetros do cluster. Deve-se observar que, a partir do Amazon DocumentDB 8.0, 'enabled' não é mais uma opção válida para default_compression, e você deve selecionar entre Zstd e LZ4.

Monitoramento

Você pode visualizar as informações de compactação de uma coleção usando um dos seguintes comandos:

- db.runCommand ({collStats: “myCollection”}) OU
- db.collection.stats ()

Esses comandos retornam as principais estatísticas que você pode usar para calcular a taxa de compactação:

- compression.algorithm: O algoritmo usado (“lz4" ou “zstd”)
- StorageSize: O armazenamento real usado pela coleção, após a compactação. Observe que esse número inclui fragmentação (ou seja, o espaço não utilizado nas páginas do banco de dados)
- avgObjSize: O tamanho lógico médio dos documentos da coleção, descompactado. Observe que, se sua coleção tiver mais de 20 mil documentos, esse número será uma aproximação com base em uma amostra de 20 mil documentos.
- tamanho: o tamanho lógico da coleção sem compressão. Esse número é obtido multiplicando avg ObjSize pelo número total de documentos na coleção, portanto, se avg ObjSize for uma aproximação, esse número também será uma aproximação.
- contagem: Número de documentos na coleção

As CloudWatch métricas a seguir podem ser úteis ao avaliar a compactação baseada em dicionário:

- CPUUtilization
- FreeableMemory
- VolumeBytesUsed
- VolumeReadIOPs
- VolumeWriteIOPs

Métricas do CollStats:

- storageSize
- size

Além disso, pode ser útil acompanhar as métricas específicas do seu aplicativo, como latência e taxa de transferência por tipo de consulta ou API.

Segurança no Amazon DocumentDB

A segurança na nuvem AWS é a maior prioridade. Como AWS cliente, você se beneficia de um data center e de uma arquitetura de rede criados para atender aos requisitos das organizações mais sensíveis à segurança.

A segurança é uma responsabilidade compartilhada entre você AWS e você. Esta documentação ajuda a entender como aplicar o modelo de responsabilidade compartilhada ao usar o Amazon DocumentDB. O modelo de responsabilidade compartilhada descreve isto como segurança da nuvem e segurança na nuvem.

- Segurança da nuvem — AWS é responsável por proteger a infraestrutura que executa AWS serviços na AWS nuvem. AWS também fornece serviços que você pode usar com segurança. Third-party auditores testam e verificam regularmente a eficácia de nossa segurança como parte dos programas de AWS conformidade. Para saber mais sobre os programas de conformidade que se aplicam ao Amazon DocumentDB (compatível com MongoDB), consulte Serviços AWS em Escopo pelo Programa de Conformidade.
- Segurança na nuvem — Sua responsabilidade é determinada pelo AWS serviço que você usa. Também existe a responsabilidade por outros fatores, incluindo a confidencialidade de dados, os requisitos da organização e as leis e regulamentos aplicáveis.

O Amazon DocumentDB é autorizado pelo Programa Federal de Gerenciamento de Riscos e Autorizações (Federal Risk and Authorization Management Program, ou FedRAMP). Ele tem autorização FedRAMP High para regiões AWS GovCloud (EUA) e autorização FedRAMP Moderate para regiões dos EUA. AWS East/West Para obter detalhes sobre os esforços de conformidade AWS e os esforços de conformidade, consulte AWS Serviços no escopo do programa de conformidade.

> **Note**
>
> Este capítulo se aplica tanto a clusters baseados em instâncias quanto a clusters elásticos. Para obter mais informações, consulte os tópicos abaixo:

Você também aprenderá a usar outros AWS serviços que ajudam a monitorar e proteger seus recursos do Amazon DocumentDB. Os tópicos a seguir mostram como configurar o Amazon DocumentDB para atender aos seus objetivos de segurança e compatibilidade.

Tópicos

- Gerenciamento de senhas com o Amazon DocumentDB e AWS Secrets Manager
- Proteção de dados no Amazon DocumentDB
- Gerenciamento de identidade e Gerenciamento de acesso para o Amazon DocumentDB
- Cross-service prevenção adjunta confusa
- Autenticação usando identidade do IAM
- Autenticação usando SCRAM-SHA-256
- Gerenciar usuários do Amazon DocumentDB
- Acesso ao banco de dados usando o controle de Role-Based acesso
- Atualizar seus certificados TLS do Amazon DocumentDB
- Validação de conformidade no Amazon DocumentDB
- Resiliência no Amazon DocumentDB
- Segurança da infraestrutura no Amazon DocumentDB
- API do Amazon DocumentDB e endpoints de VPC de interface (AWS PrivateLink)
- Práticas recomendadas de segurança para o Amazon DocumentDB
- Auditar eventos do Amazon DocumentDB
- Amazon VPC e o Amazon DocumentDB

Gerenciamento de senhas com o Amazon DocumentDB e AWS Secrets Manager

O Amazon DocumentDB se integra ao Secrets Manager para gerenciar senhas de usuário primárias para seus clusters.

Tópicos

- Limitações da integração do Secrets Manager com o Amazon DocumentDB
- Visão geral do gerenciamento de senhas de usuários primários com AWS Secrets Manager
- Impondo o gerenciamento do Amazon DocumentDB da senha do usuário principal em AWS Secrets Manager
- Gerenciamento da senha do usuário principal para um cluster com o Secrets Manager

Limitações da integração do Secrets Manager com o Amazon DocumentDB

O gerenciamento de senhas do usuário principal com o Secrets Manager não oferece suporte aos recursos a seguir:

- Clusters que fazem parte de um banco de dados global do Amazon DocumentDB.
- Réplicas de leitura entre regiões do Amazon DocumentDB

Visão geral do gerenciamento de senhas de usuários primários com AWS Secrets Manager

Com AWS Secrets Manager, você pode substituir credenciais codificadas em seu código, incluindo senhas de banco de dados, por uma chamada de API para o Secrets Manager para recuperar o segredo programaticamente. Para ter mais informações sobre o Secrets Manager, consulte o Guia do usuário do AWS Secrets Manager.

Quando você armazena segredos do banco de dados no Secrets Manager, sua AWS conta incorre em cobranças. Para obter mais informações sobre preços, consulte Preços do AWS Secrets Manager.

É possível especificar que o Amazon DocumentDB gerencie a senha do usuário principal no Secrets Manager para um cluster do Amazon DocumentDB ao realizar uma das operações a seguir:

- Criar um cluster.
- Modificar o cluster

Quando você especifica que o Amazon DocumentDB gerencie a senha do usuário principal no Secrets Manager, o Amazon DocumentDB gera a senha e a armazena no Secrets Manager. É possível interagir diretamente com o segredo para recuperar as credenciais do usuário principal. Você também pode especificar uma chave gerenciada pelo cliente para criptografar o segredo ou usar a chave KMS fornecida pelo Secrets Manager.

O Amazon DocumentDB gerencia as configurações do segredo e o alterna a cada sete dias, por padrão. É possível modificar algumas configurações, como o cronograma de alternância. Se você excluir um cluster que gerencie um segredo no Secrets Manager, o segredo e seus metadados associados também serão excluídos.

Para conectar-se a um cluster com as credenciais em um segredo, é possível recuperar o segredo do Secrets Manager. Para obter mais informações, consulte Obter segredos AWS Secrets Manager e conectar-se a um banco de dados SQL usando JDBC com credenciais em um AWS Secrets Manager segredo no Guia do AWS Secrets Manager usuário.

Impondo o gerenciamento do Amazon DocumentDB da senha do usuário principal em AWS Secrets Manager

É possível usar as chaves de condição do IAM para impor o gerenciamento pelo Amazon DocumentDB da senha do usuário principal no AWS Secrets Manager. A política a seguir não permite que os usuários criem nem restaurem instâncias ou clusters, a menos que a senha do usuário principal seja gerenciada pelo Amazon DocumentDB no Secrets Manager.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "rds:CreateDBCluster"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "rds:ManageMasterUserPassword": false
                }
            }
        }
    ]
}
```

Gerenciamento da senha do usuário principal para um cluster com o Secrets Manager

É possível configurar o gerenciamento pelo Amazon DocumentDB da senha do usuário principal no Secrets Manager com a realização das ações a seguir:

- Criar um cluster do Amazon DocumentDB
- Modificar um cluster do Amazon DocumentDB

Você pode usar o console do Amazon DocumentDB ou o AWS CLI para realizar essas ações.

Proteção de dados no Amazon DocumentDB

O modelo de responsabilidade AWS https://aws.amazon.com/compliance/shared-responsibility-model/ se aplica à proteção de dados no Amazon DocumentDB. Conforme descrito neste modelo, AWS é responsável por proteger a infraestrutura global que opera todos os Nuvem AWS. Você é responsável por manter o controle sobre o conteúdo hospedado nessa infraestrutura. Você também é responsável pelas tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que usa. Para obter mais informações sobre privacidade de dados, consulte Perguntas frequentes sobre privacidade de dados e termos AWS da região da China. Para obter informações sobre proteção de dados na Europa, consulte o Centro de Regulamento Geral sobre a Proteção de Dados (RGPD).

Para fins de proteção de dados, recomendamos que você proteja Conta da AWS as credenciais e configure usuários individuais com Centro de Identidade do AWS IAM ou AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com AWS recursos. Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Configure a API e o registro de atividades do usuário com AWS CloudTrail. Para obter informações sobre como usar CloudTrail trilhas para capturar AWS atividades, consulte Como trabalhar com CloudTrail trilhas no Guia AWS CloudTrail do usuário.
- Use soluções AWS de criptografia, junto com todos os controles de segurança padrão Serviços da AWS.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados sensíveis armazenados no Amazon S3.
- Se você precisar de módulos criptográficos validados pelo FIPS 140-3 ao acessar AWS por meio de uma interface de linha de comando ou de uma API, use um endpoint FIPS. Para saber mais sobre os endpoints FIPS disponíveis, consulte Federal Information Processing Standard (FIPS) 140-3.

É altamente recomendável que nunca sejam colocadas informações confidenciais ou sensíveis, como endereços de e-mail de clientes, em tags ou campos de formato livre, como um campo Nome. Isso inclui quando você trabalha com o Amazon DocumentDB ou outro Serviços da AWS usando o console, a API ou os AWS SDKs. AWS CLI Quaisquer dados inseridos em tags ou em campos de texto de formato livre usados para nomes podem ser usados para logs de faturamento ou de diagnóstico. Se você fornecer um URL para um servidor externo, é fortemente recomendável que não sejam incluídas informações de credenciais no URL para validar a solicitação nesse servidor.

Tópicos

- Client-side criptografia em nível de campo
- Criptografar dados em repouso do Amazon DocumentDB
- Criptografia de dados em trânsito
- Gerenciamento de chaves

Client-side criptografia em nível de campo

A Field Level Encryption (FLE – Criptografia em nível de campo) do lado do cliente do Amazon DocumentDB permite que você criptografe dados confidenciais em aplicações clientes antes de serem transferidos para um cluster do Amazon DocumentDB. Os dados confidenciais permanecem criptografados quando armazenados e processados em um cluster e são descriptografados na aplicação cliente quando recuperados.

Tópicos

- Introdução
- Consulta no FLE do lado do cliente

Introdução

A configuração inicial da FLE do lado do cliente no Amazon DocumentDB é um processo de quatro etapas que inclui a criação de uma chave de criptografia, a associação de um perfil à aplicação, a configuração da aplicação e a definição da operação CRUD com opções de criptografia.

Tópicos

- Etapa 1: criar as chaves de criptografia
- Etapa 2: associar um perfil à aplicação

- Etapa 3: configurar a aplicação
- Etapa 4: definir uma operação CRUD
- Exemplo: arquivo de configuração de criptografia em nível de campo do lado do cliente

Etapa 1: criar as chaves de criptografia

Usando AWS Key Management Service, crie uma chave simétrica gerenciada pelo cliente que criptografa e descriptografa o campo de dados confidenciais e conceda a ele as permissões de uso do IAM necessárias. AWS KMS armazena a chave gerenciada pelo cliente, que é usada para criptografar chaves de dados. Armazenar a chave gerenciada pelo cliente AWS KMS fortalece sua postura de segurança. A chave de dados é a chave secundária, que é armazenada em uma coleção do Amazon DocumentDB e é necessária para criptografar campos confidenciais antes de armazenar o documento no Amazon DocumentDB. A chave gerenciada pelo cliente criptografa a chave de dados, que, por sua vez, criptografa e descriptografa seus dados. Se você usa um cluster global, pode criar uma chave multirregional que diferentes funções de serviço podem usar em diferentes regiões.

Para obter mais informações sobre o AWS Key Management Service, incluindo como criar uma chave, consulte AWS Key Management Service os conceitos no Guia do AWS Key Management Service desenvolvedor.

Etapa 2: associar um perfil à aplicação

Crie uma política do IAM com AWS KMS as permissões apropriadas. Essa política permite que as identidades do IAM às quais ela está anexada criptografem e descriptografem a chave KMS especificada no campo do recurso. Seu aplicativo assume essa função do IAM para se autenticar. AWS KMS

A política deve ter a seguinte aparência:

```
{ "Effect": "Allow",
"Action": ["kms:Decrypt", "kms:Encrypt"],
"Resource": "Customer Key ARN"
}
```

Etapa 3: configurar a aplicação

Você definiu uma chave gerenciada pelo cliente AWS KMS e criou uma função do IAM com as permissões necessárias para acessá-la. Importe os pacotes necessários.

```
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 como o tipo de AWS KMS provedor e insira as credenciais da conta que você recuperou na etapa anterior.

   ```
   provider = "aws"
   kms_providers = {
       provider: {
           "accessKeyId": current_credentials.access_key,
           "secretAccessKey": current_credentials.secret_key
       }
   }
   ```

2. Especifique a chave gerenciada pelo cliente para criptografar a chave de dados:

   ```
   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 o 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. Gere sua chave de dados:

```
data_key_id = client_encryption.create_data_key(provider,
    customer_key,
    key_alt_name = [key_alt_name])
```

5. Recupere sua chave de dados 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})
```

Etapa 4: definir uma operação CRUD

Defina a operação CRUD com opções de criptografia.

1. Defina a coleção para write/read /excluir um único documento:

```
coll = client.gameinfo.users
```

2. Criptografia explícita - criptografe campos e insira:

> **Note**
>
> É necessário fornecer exatamente um “key_id” ou “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"]
})
```

Exemplo: arquivo de configuração de criptografia em nível de campo do lado do cliente

No exemplo a seguir, substitua cada *`placeholder`* por suas próprias informações.

```
# 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()
```

Consulta no FLE do lado do cliente

O Amazon DocumentDB só oferece suporte a consultas de igualdade com valores criptografados FLE do lado do cliente.

Por exemplo, para consultar documentos em que a pontuação do jogador criptografada é igual a 500, o cliente usa um método de criptografia explícito para criptografar o valor da 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 }
} )
```

Criptografar dados em repouso do Amazon DocumentDB

> **Note**
>
> AWS KMS está substituindo o termo chave mestra do cliente (CMK) por AWS KMS key chave KMS. O conceito não mudou. Para evitar alterações significativas, AWS KMS está mantendo algumas variações desse termo.

Você criptografa os dados em repouso em seu cluster do Amazon DocumentDB especificando a opção de criptografia de armazenamento ao criar o cluster. A criptografia de armazenamento é ativada em todo o cluster e é aplicada a todas as instâncias, incluindo a instância principal e todas as réplicas. Ela também é aplicada a volumes de armazenamento, dados, índices, logs, backups automatizados e snapshots do cluster.

O Amazon DocumentDB usa o Advanced Encryption Standard (AES-256) de 256 bits para criptografar seus dados usando chaves de criptografia armazenadas em (). AWS Key Management Service AWS KMS Ao usar um cluster Amazon DocumentDB com criptografia em repouso ativada, você não precisa modificar a lógica da aplicação ou a conexão do cliente. O Amazon DocumentDB lida de forma transparente com a descriptografia de seus dados com um impacto mínimo sobre o desempenho.

O Amazon DocumentDB se integra AWS KMS e usa um método conhecido como criptografia de envelope para proteger seus dados. Quando um cluster do Amazon DocumentDB é criptografado com um AWS KMS, o Amazon DocumentDB solicita o uso de sua chave KMS AWS KMS para gerar uma chave de dados de texto cifrado para criptografar o volume de armazenamento. A chave de dados de texto cifrado é criptografada usando a chave KMS definida e armazenada com os dados criptografados e os metadados de armazenamento. Quando o Amazon DocumentDB precisa acessar seus dados criptografados, ele solicita AWS KMS a descriptografia da chave de dados de texto cifrado usando sua chave KMS e armazena em cache a chave de dados de texto simples na memória para criptografar e descriptografar dados de forma eficiente no volume de armazenamento.

O recurso de criptografia de armazenamento no Amazon DocumentDB está disponível para todos os tamanhos de instância suportados e em todos os Regiões da AWS lugares onde o Amazon DocumentDB está disponível.

Ativar a criptografia em repouso para um cluster do Amazon DocumentDB

Você pode ativar ou desativar a criptografia em repouso em um cluster do Amazon DocumentDB quando o cluster é provisionado usando o Console de gerenciamento da AWS ou o AWS Command Line Interface ().AWS CLI Os clusters criados usando o console têm a criptografia em repouso habilitada por padrão. Os clusters que você cria usando o AWS CLI têm a criptografia em repouso desativada por padrão. Portanto, você deve explicitamente habilitar a criptografia em repouso usando o parâmetro `--storage-encrypted`. Em ambos os casos, após o cluster ser criado, não é possível alterar a opção de criptografia em repouso.

O Amazon DocumentDB usa AWS KMS para recuperar e gerenciar chaves de criptografia e definir as políticas que controlam como essas chaves podem ser usadas. Se você não especificar um identificador de AWS KMS chave, o Amazon DocumentDB usa a chave KMS padrão do serviço AWS gerenciado. O Amazon DocumentDB cria uma chave KMS separada para cada uma Região da AWS em sua. Conta da AWS Para obter mais informações, consulte Conceitos do AWS Key Management Service.

Para começar a criar sua própria chave KMS, consulte Conceitos básicos no Guia do Desenvolvedor da AWS Key Management Service .

> **Important**
>
> É necessário usar uma chave do KMS de criptografia simétrica para criptografar seu cluster, pois o Amazon DocumentDB só oferece suporte a chaves KMS de criptografia simétrica. Não use uma chave KMS assimétrica para tentar criptografar os dados nos clusters do Amazon DocumentDB. Para ter mais informações, consulte Chaves assimétricas do AWS KMS no Guia do desenvolvedor do AWS Key Management Service .

Se o Amazon DocumentDB não puder mais acessar a chave KMS de um cluster — por exemplo, quando o Conta da AWS proprietário da chave é suspenso, a chave é desativada, a chave é programada para exclusão ou a política ou concessão de chaves da qual o Amazon DocumentDB depende é removida — o cluster primeiro passa para o status. `inaccessible-encryption-credentials-recoverable` Enquanto o cluster estiver nesse status, o Amazon DocumentDB interrompe as instâncias do cluster e você não poderá ler ou gravar no cluster, mas o cluster ainda

poderá ser recuperado se o acesso à chave KMS for restaurado em 7 dias. Se o acesso não for restaurado em 7 dias, o cluster passará para o `inaccessible-encryption-credentials` status do terminal. A partir do status do terminal, o cluster não está mais disponível e o estado atual do banco de dados não pode ser recuperado — você só pode restaurar a partir de um backup ou realizar uma restauração pontual usando a chave KMS original. Para o Amazon DocumentDB, os backups estão sempre habilitados por pelo menos 1 dia.

> **Note**
>
> Clusters que fazem parte de um cluster global se comportam de forma diferente. Quando o Amazon DocumentDB detecta que não pode mais acessar a chave KMS, todos os clusters no cluster global passam diretamente para o status do terminal, ignorando o `inaccessible-encryption-credentials` status recuperável. Isso ocorre porque um cluster que faz parte de um cluster global só pode ser interrompido e iniciado quando é o único cluster no cluster global. Para se recuperar, você deve restaurar a partir de um snapshot ou realizar uma restauração pontual. Para excluir os clusters originais, primeiro você deve remover cada cluster do cluster global e depois excluí-lo.

> **Important**
>
> Não é possível alterar a chave KMS para um cluster criptografado depois de já tê-lo criado. Certifique-se de determinar seus requisitos de chave de criptografia antes de criar seu cluster criptografado.

Using the Console de gerenciamento da AWS

Especifique a opção de criptografia em repouso ao criar um cluster. A criptografia em repouso é habilitada por padrão quando você cria um cluster usando o Console de gerenciamento da AWS. Não é possível alterá-la após criar o cluster.

Para especificar a opção de criptografia em repouso ao criar o cluster

1. Crie um cluster do Amazon DocumentDB conforme descrito na seção Conceitos básicos. No entanto, na etapa 6, não selecione Criar cluster.
2. Abaixo da seção Autenticação, escolha Mostrar configurações avançadas.
3. Role para baixo até a seção Encryption-at-rest.

4. Escolha a opção que deseja para a criptografia em repouso. Seja qual for a opção que escolher, não será possível alterá-la após criar o cluster.
   - Para criptografar dados em repouso nesse cluster, selecione Habilitar criptografia.
   - Caso não queira criptografar dados em repouso nesse cluster, selecione Desabilitar criptografia.
5. Escolha a chave primária que você deseja. O Amazon DocumentDB usa o AWS Key Management Service (AWS KMS) para recuperar e gerenciar chaves de criptografia e definir as políticas que controlam como essas chaves podem ser usadas. Se você não especificar um identificador de AWS KMS chave, o Amazon DocumentDB usa a chave KMS padrão do serviço AWS gerenciado. Para obter mais informações, consulte Conceitos do AWS Key Management Service.

   > **Note**
   >
   > Depois de criar um cluster criptografado, não é possível alterar a chave KMS para esse cluster. Certifique-se de determinar seus requisitos de chave de criptografia antes de criar seu cluster criptografado.

6. Complete as outras seções conforme o necessário e crie o cluster.

Using the AWS CLI

Para criptografar um cluster do Amazon DocumentDB usando o AWS CLI, execute o `create-db-cluster` comando e especifique a opção. `--storage-encrypted` Os clusters do Amazon DocumentDB criados usando o AWS CLI não habilitam a criptografia de armazenamento por padrão.

Veja a seguir um exemplo de como criar um cluster do Amazon DocumentDB com a criptografia de armazenamento ativada.

Nos exemplos a seguir, substitua cada um *`user input placeholder`* pelas informações do seu cluster.

Example

Para Linux, macOS ou 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
```

Ao criar um cluster criptografado do Amazon DocumentDB, você pode especificar um identificador de AWS KMS chave, como no exemplo a seguir.

Example

Para Linux, macOS ou 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**
>
> Depois de criar um cluster criptografado, não é possível alterar a chave KMS para esse cluster. Certifique-se de determinar seus requisitos de chave de criptografia antes de criar seu cluster criptografado.

Resolvendo um cluster do Amazon DocumentDB em um estado de criptografia inacessível

Um cluster do Amazon DocumentDB passa para um status de criptografia inacessível quando o Amazon DocumentDB não consegue acessar a chave KMS com a qual o cluster foi criptografado. Há dois desses status, e o caminho de recuperação depende de qual deles o cluster está.

`status recuperável de credenciais de criptografia inacessíveis`

Enquanto o cluster estiver no `inaccessible-encryption-credentials-recoverable` status, você pode retornar ao cluster `available` restaurando o acesso do Amazon DocumentDB à chave KMS e, em seguida, iniciando o cluster. Para resolver esse status, faça o seguinte:

1. Confirme se Conta da AWS o proprietário da chave KMS está ativo. Se a conta for suspensa, reative-a.
2. Confirme se a chave KMS está ativada. Para obter mais informações, consulte Habilitar e desabilitar chaves no AWS Key Management Service Guia do desenvolvedor do .
3. Verifique se a chave KMS está programada para exclusão. Se estiver, cancele a exclusão programada da chave. Para obter mais informações, consulte Agendamento e cancelamento da exclusão de chaves no Guia do AWS Key Management Service desenvolvedor.
4. Confirme se a política de chaves do KMS e todas as concessões das quais o Amazon DocumentDB depende ainda permitem que o Amazon DocumentDB use a chave.
5. Depois que o acesso à chave KMS for restaurado, inicie o cluster usando Console de gerenciamento da AWS ou executando o `start-db-cluster` AWS CLI comando.

   Example

   Para Linux, macOS ou Unix:

   ```
   aws docdb start-db-cluster \
     --db-cluster-identifier example-cluster
   ```

Para Windows:

```
aws docdb start-db-cluster ^
  --db-cluster-identifier example-cluster
```

> **Important**
>
> Se o acesso à chave KMS não for restaurado em 7 dias, o cluster passará para o `inaccessible-encryption-credentials` status do terminal, do qual não poderá ser iniciado.

`status de credenciais de criptografia inacessíveis`

O `inaccessible-encryption-credentials` status é terminal. O cluster não pode ser iniciado e o estado de execução do banco de dados não pode ser recuperado. Para recuperar seus dados, restaure a partir de um snapshot ou execute uma restauração pontual em um novo cluster. Você ainda deve ter acesso à chave KMS original para realizar a restauração. Se a chave KMS tiver sido excluída, os dados não poderão ser recuperados.

Para obter mais informações, consulte Restauração de um snapshot de cluster e Restauração para um ponto no tempo.

> **Note**
>
> Os clusters que fazem parte de um cluster global passam diretamente para o `inaccessible-encryption-credentials` status quando o acesso à chave KMS é perdido, porque um cluster que faz parte de um cluster global só pode ser interrompido e iniciado quando é o único cluster no cluster global. Para excluir um cluster que esteja nesse status, primeiro remova cada cluster do cluster global e, em seguida, exclua os clusters individualmente.

Se você não conseguir excluir um cluster que está no `inaccessible-encryption-credentials` status porque a proteção contra exclusão está ativada, desative a proteção contra exclusão usando o AWS CLI antes de tentar a exclusão novamente.

Example

Para Linux, macOS ou 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
```

Em seguida, você pode excluir o cluster usando o `delete-db-cluster` comando.

Example

Para Linux, macOS ou 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
```

Se o cluster não for excluído depois de executar os comandos anteriores, entre em contato com AWS Support.

Limitações para clusters criptografados do Amazon DocumentDB

As seguintes limitações existem para clusters criptografados do Amazon DocumentDB.

- É possível habilitar ou desabilitar a criptografia em repouso para um cluster do Amazon DocumentDB somente no momento em que ele é criado, e não após a criação ser concluída. No entanto, é possível criar uma cópia criptografada de um cluster decriptografado criando um

snapshot do cluster decriptografado e, em seguida, restaure o snapshot decriptografado como um novo cluster ao especificar a opção de criptografia em repouso.

Para saber mais, consulte os seguintes tópicos:

- Criação de um snapshot manual de cluster
- Restauração de um snapshot de cluster
- Cópia de snapshots de cluster do Amazon DocumentDB

- Os clusters do Amazon DocumentDB com criptografia de armazenamento habilitada não podem ser modificados para desabilitar a criptografia.
- Todas as instâncias, os backups automatizados, os snapshots e os índices em um cluster do Amazon DocumentDB são criptografados com a mesma chave KMS.

Criptografia de dados em trânsito

É possível usar o Transport Layer Security (TLS) para criptografar a conexão entre sua aplicação e um cluster do Amazon DocumentDB. Por padrão, a criptografia em trânsito é ativada para clusters recém-criados do Amazon DocumentDB. É possível desabilitá-la ao criar o cluster ou depois da criação ser concluída. Ao habilitar a criptografia em trânsito, as conexões seguras usando o TLS são obrigatórias para se conectar ao cluster. Para obter mais informações sobre como se conectar ao Amazon DocumentDB usando TLS, consulte Conectar-se de forma programática ao Amazon DocumentDB.

Gerenciar configurações de TLS de cluster do Amazon DocumentDB

A criptografia em trânsito para um cluster do Amazon DocumentDB é gerenciada por meio do parâmetro TLS em um grupo de parâmetros de cluster. Você pode gerenciar suas configurações de TLS do cluster Amazon DocumentDB usando o Console de gerenciamento da AWS ou o AWS Command Line Interface ()AWS CLI. Consulte as seções a seguir para saber como verificar e modificar suas configurações de TLS atuais.

Using the Console de gerenciamento da AWS

Para gerenciar a criptografia usando TLS e o , como identificar grupos de parâmetros, verificar o valor de TLS e fazer as modificações necessárias, use as etapas a seguir.

> **Note**
>
> A menos que você especifique de outra forma ao criar o cluster, ele será criado com o grupo de parâmetros de cluster padrão. Os parâmetros no grupo de parâmetros do `default` cluster não podem ser modificados (por exemplo, `tls` enabled/disabled). Portanto, se o cluster estiver usando um grupo de parâmetros de cluster `default`, será necessário modificar o cluster para usar um grupo de parâmetros de cluster que não seja padrão. Primeiro, será necessário criar um grupo de parâmetros de cluster personalizado. Para obter mais informações, consulte Criando grupos de parâmetros de cluster do Amazon DocumentDB.

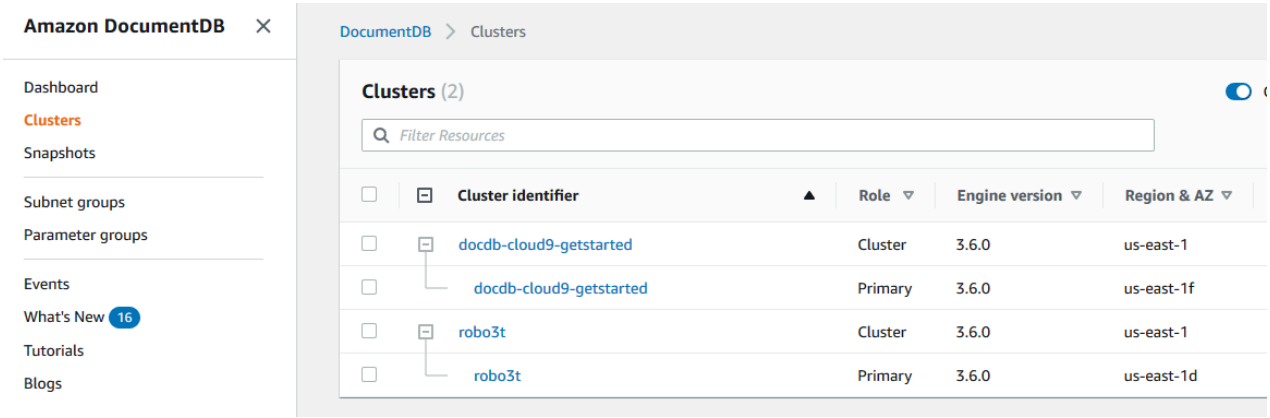
1. Determine o grupo de parâmetros de cluster usado pelo cluster.

   a. Abra o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.

   b. No painel de navegação, escolha Clusters.

   > **Tip**
   >
   > Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

   c. Observe que na caixa de navegação Clusters, a coluna Identificador do cluster mostra clusters e instâncias. As instâncias estão listadas abaixo dos clusters. Veja a captura de tela a seguir para referência.



   d. Escolha o cluster que você deseja usar.

e. Escolha a guia Configuração e role para baixo até a parte inferior dos Detalhes do cluster e localize o Grupo de parâmetros do cluster.. Anote o nome do grupo de parâmetros de cluster.

   Se o nome do grupo de parâmetros do cluster para `default` (por exemplo `default.docdb3.6`), será necessário criar um grupo de parâmetros de cluster personalizado e defini-lo como o grupo de parâmetros do cluster antes de continuar. Para saber mais, consulte:

   1. Criando grupos de parâmetros de cluster do Amazon DocumentDB — Se você não tiver um grupo de parâmetros de cluster personalizado para usar, crie um.
   2. Modificar um cluster do Amazon DocumentDB — Modifique seu cluster para usar o grupo de parâmetros de cluster personalizado.

2. Determine o valor atual do parâmetro **`tls`** de cluster.

   a. Abra o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.

   b. No painel de navegação, escolha Grupos de parâmetros.

   c. Na lista de grupos de parâmetros, selecione o nome do grupo de parâmetros de cluster desejado.

   d. Localize a seção Parâmetros do cluster. Na lista de parâmetros de cluster, localize a linha do parâmetro de cluster `tls`. Nesse momento, as quatro colunas a seguir são importantes:

      - Nome do parâmetro de cluster — O nome dos parâmetros do cluster. Para gerenciar TLS, você está interessado no parâmetro de cluster `tls`.
      - Valores — O valor atual de cada parâmetro do cluster.
      - Valores permitidos — Uma lista de valores que podem ser aplicados a um parâmetro de cluster.
      - Aplicar tipo — estático ou dinâmico. As alterações em parâmetros de cluster estáticos poderão ser aplicadas somente quando as instâncias forem reiniciadas. As alterações feitas em parâmetros de cluster dinâmicos podem ser aplicadas imediatamente ou quando as instâncias são reiniciadas.

3. Modifique o valor do parâmetro **`tls`** do cluster.

Se o valor de `tls` não for o que é necessário, modifique o valor para esse grupo de parâmetros de cluster. Para alterar o valor do parâmetro de cluster `tls`, continue na seção anterior seguindo estas etapas.

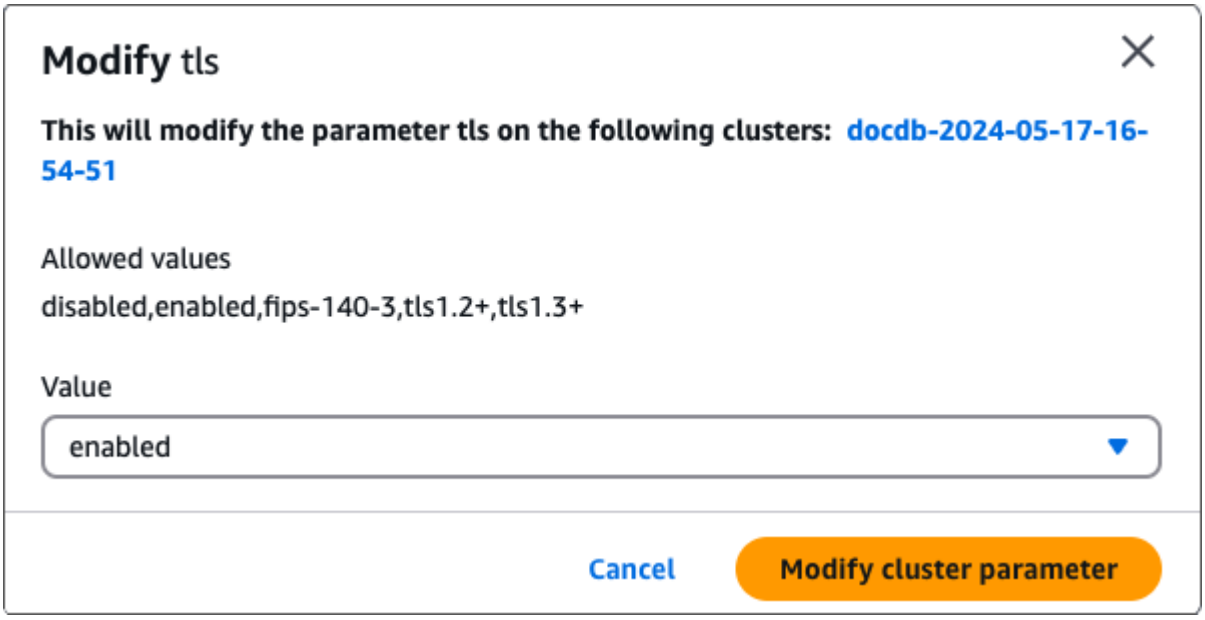
a. Escolha o botão à esquerda do nome do parâmetro de cluster (`tls`).

b. Escolha Editar.

c. Para alterar o valor de `tls`, na caixa de diálogo Modificar **`tls`**, escolha o valor desejado para o parâmetro do cluster na lista suspensa.

   Os valores válidos são:

   - desabilitado — Desativa o TLS
   - ativado — Ativa o TLS. Para o Amazon DocumentDB 5.0 e versões anteriores, isso habilita o TLS 1.0 a 1.3, mas usa o TLS 1.2 ou superior. Somente o TLS 1.2 e superior é compatível com o Amazon DocumentDB 5.0, começando com a versão secundária 5.0.1, Amazon DocumentDB 8.0 e versões mais recentes.
   - fips-140-3 — Ativa o TLS com FIPS. O cluster só aceita conexões seguras de acordo com os requisitos da publicação 140-3 do Federal Information Processing Standards (FIPS). Isso só é suportado a partir dos clusters Amazon DocumentDB 5.0 (engine versão 3.0.3727) nas seguintes regiões: ca-central-1, us-west-2, us-east-1, us-east-2, us-gov-east-1, us-gov-west-1.
   - tls1.2+: habilita as versões 1.2 e posteriores do TLS. Só há suporte para isso a partir do Amazon DocumentDB 4.0 (versão de mecanismo 2.0.10980) e do Amazon DocumentDB (versão de mecanismo 3.0.11051).
   - tls1.3+: habilita as versões 1.3 e posteriores do TLS. Só há suporte para isso a partir do Amazon DocumentDB 4.0 (versão de mecanismo 2.0.10980) e do Amazon DocumentDB (versão de mecanismo 3.0.11051).



d. Escolha Modificar parâmetro de cluster. A alteração será aplicada a cada instância do cluster quando ela for reinicializada.

4. Reinicie a instância do Amazon DocumentDB.

   Reinicialize cada instância do cluster para que a alteração seja aplicada a todas as instâncias no cluster.

   a. Abra o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.

   b. No painel de navegação, escolha Instâncias.

   c. Para especificar a reinicialização de uma instância, localize a instância na lista de instâncias e escolha o botão à esquerda de seu nome.

   d. Escolha Ações e Reiniciar. Confirme se deseja reinicializar, selecionando Reiniciar.

Using the AWS CLI

Para gerenciar a criptografia usando TLS e o AWS CLI, como identificar grupos de parâmetros, verificar o valor de TLS e fazer as modificações necessárias, use as etapas a seguir.

> **Note**
>
> A menos que você especifique de outra forma ao criar o cluster, ele será criado com o grupo de parâmetros de cluster padrão. Os parâmetros no grupo de parâmetros do

`default` cluster não podem ser modificados (por exemplo, `tls` enabled/disabled). Portanto, se o cluster estiver usando um grupo de parâmetros de cluster `default`, será necessário modificar o cluster para usar um grupo de parâmetros de cluster que não seja padrão. Pode ser necessário primeiro criar um grupo de parâmetros de cluster personalizado. Para obter mais informações, consulte Criando grupos de parâmetros de cluster do Amazon DocumentDB.

1. Determine o grupo de parâmetros de cluster usado pelo cluster.

   Execute o comando `describe-db-clusters` com as opções a seguir:

   - `--db-cluster-identifier`
   - `--query`

   No exemplo a seguir, substitua cada um *`user input placeholder`* pelas informações do seu cluster.

   ```
   aws docdb describe-db-clusters \
     --db-cluster-identifier mydocdbcluster \
     --query 'DBClusters[*].[DBClusterIdentifier,DBClusterParameterGroup]'
   ```

   A saída dessa operação é semelhante ao seguinte (formato JSON):

   ```
   [
       [
           "mydocdbcluster",
           "myparametergroup"
       ]
   ]
   ```

   Se o nome do grupo de parâmetros do cluster for `default` (por exemplo, `default.docdb3.6`), será necessário ter um grupo de parâmetros de cluster personalizado e defini-lo como o grupo de parâmetros do cluster antes de continuar. Para saber mais, consulte os seguintes tópicos:

   1. Criando grupos de parâmetros de cluster do Amazon DocumentDB — Se você não tiver um grupo de parâmetros de cluster personalizado para usar, crie um.

2. Modificar um cluster do Amazon DocumentDB — Modifique seu cluster para usar o grupo de parâmetros de cluster personalizado.

2. Determine o valor atual do parâmetro **`tls`** de cluster.

   Para obter mais informações sobre esse grupo de parâmetros de cluster, execute o comando `describe-db-cluster-parameters` com as opções a seguir:

   - `--db-cluster-parameter-group-name`
   - `--query`

     Limita a saída apenas aos campos de interesse: `ParameterName`, `ParameterValue`, `AllowedValues` e `ApplyType`.

   No exemplo a seguir, substitua cada um *`user input placeholder`* pelas informações do seu cluster.

```
aws docdb describe-db-cluster-parameters \
  --db-cluster-parameter-group-name myparametergroup \
  --query 'Parameters[*].[ParameterName,ParameterValue,AllowedValues,ApplyType]'
```

   A saída dessa operação é semelhante ao seguinte (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. Modifique o valor do parâmetro **`tls`** do cluster.

   Se o valor de `tls` for o que ele precisa ser, modifique seu valor para este grupo de parâmetros de cluster. Para alterar o valor do parâmetro de cluster `tls`, execute o comando `modify-db-cluster-parameter-group` com as opções a seguir:

   - `--db-cluster-parameter-group-name` — Obrigatório. O nome do grupo de parâmetros de cluster a ser modificado. Não pode ser um grupo de parâmetros de cluster `default.*`.
   - `--parameters` — Obrigatório. Uma lista de parâmetros do grupo de parâmetros de cluster para modificar.
     - `ParameterName` — Obrigatório. O nome do parâmetro do cluster a ser modificado.
     - `ParameterValue` — Obrigatório. O novo valor desse parâmetro de cluster. Deve ser um dos `AllowedValues` do parâmetro de cluster.
       - `enabled`— O cluster aceita conexões seguras usando TLS. Para o Amazon DocumentDB 5.0 e versões anteriores, isso habilita o TLS 1.0 a 1.3, mas usa o TLS 1.2 ou superior. Somente o TLS 1.2 e superior é compatível com o Amazon DocumentDB 5.0, começando com a versão secundária 5.0.1, Amazon DocumentDB 8.0 e versões mais recentes.
       - `disabled` — O cluster não aceita conexões seguras usando o TLS.
       - `fips-140-3`— O cluster só aceita conexões seguras de acordo com os requisitos da publicação 140-3 do Federal Information Processing Standards (FIPS). Isso só é suportado a partir dos clusters Amazon DocumentDB 5.0 (engine versão 3.0.3727) nas seguintes regiões: ca-central-1, us-west-2, us-east-1, us-east-2, us-gov-east-1, us-gov-west-1.
       - `tls1.2+`: o cluster aceita conexões seguras usando TLS versões 1.2 e posteriores Só há suporte para isso a partir do Amazon DocumentDB 4.0 (versão de mecanismo 2.0.10980) e do Amazon DocumentDB 5.0 (versão de mecanismo 3.0.11051).
       - `tls1.3+`: o cluster aceita conexões seguras usando TLS versões 1.3 e posteriores Só há suporte para isso a partir do Amazon DocumentDB 4.0 (versão de mecanismo 2.0.10980) e do Amazon DocumentDB 5.0 (versão de mecanismo 3.0.11051).
     - `ApplyMethod` — Quando essa modificação deve ser aplicada. Para parâmetros de cluster estáticos, como `tle`, esse valor deve ser `pending-reboot`.

- `pending-reboot` — A alteração é aplicada a uma instância somente depois de ser reinicializada. É necessário reinicializar cada instância de cluster individualmente para que essa mudança ocorra em todas as instâncias do cluster.

Nos exemplos a seguir, substitua cada um *`user input placeholder`* pelas informações do seu cluster.

O código a seguir desabilita o `tls`, aplicando a alteração a cada instância quando ela for reinicializada.

```
aws docdb modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=tls,ParameterValue=disabled,ApplyMethod=pending-
reboot"
```

O código a seguir ativa `tls`, aplicando a alteração a cada instância quando ela é reinicializada.

```
aws docdb modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=tls,ParameterValue=enabled,ApplyMethod=pending-
reboot"
```

O código a seguir habilita o TLS com `fips-140-3`, aplicando a alteração a cada instância quando ela é reinicializada.

```
aws docdb modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup2 \
  --parameters "ParameterName=tls,ParameterValue=fips-140-3,ApplyMethod=pending-
reboot"
```

A saída dessa operação é semelhante ao seguinte (formato JSON):

```
{
    "DBClusterParameterGroupName": "myparametergroup"
}
```

4. Reinicialize sua instância do Amazon DocumentDB.

Reinicialize cada instância do cluster para que a alteração seja aplicada a todas as instâncias no cluster. Para reinicializar uma instância do Amazon DocumentDB, execute o comando `reboot-db-instance` com a opção a seguir:

- `--db-instance-identifier`

O código a seguir reinicializa a instância `mydocdbinstance`.

Nos exemplos a seguir, substitua cada um *`user input placeholder`* pelas informações do seu cluster.

Example

Para Linux, macOS ou Unix:

```
aws docdb reboot-db-instance \
  --db-instance-identifier mydocdbinstance
```

Para Windows:

```
aws docdb reboot-db-instance ^
  --db-instance-identifier mydocdbinstance
```

A saída dessa operação é semelhante ao seguinte (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"
    }
}
```

Demora alguns minutos para sua instância reinicializar. É possível usar a instância somente quando seu status for disponível. É possível monitorar o status da instância usando o console ou a AWS CLI. Para obter mais informações, consulte Monitoramento do status de uma instância do Amazon DocumentDB.

Gerenciamento de chaves

O Amazon DocumentDB usa AWS Key Management Service (AWS KMS) para recuperar e gerenciar chaves de criptografia. AWS KMS combina hardware e software seguros e altamente disponíveis para fornecer um sistema de gerenciamento de chaves dimensionado para a nuvem. Usando AWS KMS, você pode criar chaves de criptografia e definir as políticas que controlam como essas chaves

podem ser usadas. AWS KMS suporta AWS CloudTrail, para que você possa auditar o uso da chave para verificar se as chaves estão sendo usadas adequadamente.

Você pode usar suas chaves KMS em combinação com o Amazon DocumentDB e AWS serviços compatíveis, como Amazon Simple Storage Service (Amazon S3), Amazon Relational Database Service (Amazon RDS), Amazon Elastic Block Store (Amazon EBS) e Amazon Redshift. Para obter uma lista dos serviços que oferecem suporte AWS KMS, consulte Como AWS os serviços são usados AWS KMS no Guia do AWS Key Management Service desenvolvedor. Para obter informações sobre AWS KMS, consulte O que é AWS Key Management Service?

Gerenciamento de identidade e Gerenciamento de acesso para o Amazon DocumentDB

AWS Identity and Access Management (IAM) é um AWS service (Serviço da AWS) que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. Os administradores do IAM controlam quem pode ser autenticado (conectado) e autorizado (ter permissões) para utilizar os recursos do Amazon DocumentDB. O IAM é um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

Tópicos

- Público
- Autenticação com identidades
- Gerenciar o acesso usando políticas
- Como o Amazon DocumentDB funciona com o IAM
- Identity-based exemplos de políticas para o Amazon DocumentDB
- Solução de problemas de identidade e acesso da Amazon DocumentDB
- Gerenciar permissões de acesso aos recursos do Amazon DocumentDB
- Usar políticas baseadas em identidade (políticas do IAM) para o Amazon DocumentDB
- AWS políticas gerenciadas para o Amazon DocumentDB
- Permissões da API do Amazon DocumentDB: referência de ações, recursos e condições

Público

A forma como você usa AWS Identity and Access Management (IAM) difere com base na sua função:

- Usuário do serviço: solicite permissões ao seu administrador se você não conseguir acessar os atributos (consulte Solução de problemas de identidade e acesso da Amazon DocumentDB).
- Administrador do serviço: determine o acesso do usuário e envie solicitações de permissão (consulte Como o Amazon DocumentDB funciona com o IAM)
- Administrador do IAM: escreva políticas para gerenciar o acesso (consulte Identity-based exemplos de políticas para o Amazon DocumentDB)

Autenticação com identidades

A autenticação é como você faz login AWS usando suas credenciais de identidade. Você deve ser autenticado como usuário do IAM ou assumindo uma função do IAM. Usuário raiz da conta da AWS

Você pode entrar como uma identidade federada usando credenciais de uma fonte de identidade, como Centro de Identidade do AWS IAM (IAM Identity Center), autenticação de login único ou credenciais. Google/Facebook Para ter mais informações sobre como fazer login, consulte Como fazer login em sua Conta da AWS no Guia do usuário do Início de Sessão da AWS .

Para acesso programático, AWS fornece um SDK e uma CLI para assinar solicitações criptograficamente. Para ter mais informações, consulte AWS Signature Version 4 para solicitações de API no Guia do usuário do IAM.

Conta da AWS usuário-raiz

Ao criar um Conta da AWS, você começa com uma identidade de login chamada usuário Conta da AWS raiz que tem acesso completo a todos Serviços da AWS os recursos. É altamente recomendável não usar o usuário-raiz em tarefas diárias. Consulte as tarefas que exigem credenciais de usuário-raiz em Tarefas que exigem credenciais de usuário-raiz no Guia do usuário do IAM.

Identidade federada

Como melhor prática, exija que usuários humanos usem a federação com um provedor de identidade para acessar Serviços da AWS usando credenciais temporárias.

Uma identidade federada é um usuário do seu diretório corporativo, provedor de identidade da web ou Directory Service que acessa Serviços da AWS usando credenciais de uma fonte de identidade. As identidades federadas assumem funções que oferecem credenciais temporárias.

Para o gerenciamento de acesso centralizado, recomendamos Centro de Identidade do AWS IAM. Para saber mais, consulte O que é o IAM Identity Center? no Guia do usuário do Centro de Identidade do AWS IAM .

Usuários e grupos do IAM

Um usuário do IAM é uma identidade com permissões específicas para uma única pessoa ou aplicação. É recomendável usar credenciais temporárias, em vez de usuários do IAM com credenciais de longo prazo. Para obter mais informações, consulte Exigir que usuários humanos usem a federação com um provedor de identidade para acessar AWS usando credenciais temporárias no Guia do usuário do IAM.

Um grupo do IAM especifica um conjunto de usuários do IAM e facilita o gerenciamento de permissões para grandes conjuntos de usuários. Para ter mais informações, consulte Casos de uso de usuários do IAM no Guia do usuário do IAM.

Perfis do IAM

Uma perfil do IAM é uma identidade com permissões específicas que oferece credenciais temporárias. Você pode assumir uma função mudando de um usuário para uma função do IAM (console) ou chamando uma operação de AWS API AWS CLI ou. Para saber mais, consulte Métodos para assumir um perfil no Manual do usuário do IAM.

Os perfis do IAM são úteis para acesso de usuário federado, permissões de usuário do IAM temporárias, acesso entre contas, acesso entre serviços e aplicações em execução no Amazon EC2. Consulte mais informações em Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Gerenciar o acesso usando políticas

Você controla o acesso AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política define permissões quando associada a uma identidade ou recurso. AWS avalia essas políticas quando um diretor faz uma solicitação. A maioria das políticas é armazenada AWS como documentos JSON. Para ter mais informações sobre documentos de política JSON, consulte Visão geral das políticas JSON no Guia do usuário do IAM.

Por meio de políticas, os administradores especificam quem tem acesso a que, definindo qual entidade principal pode realizar ações em quais recursos e sob quais condições.

Por padrão, usuários e perfis não têm permissões. Um administrador do IAM cria políticas do IAM e as adiciona aos perfis, os quais os usuários podem então assumir. As políticas do IAM definem permissões, independentemente do método usado para realizar a operação.

Identity-based políticas

Identity-based políticas são documentos de política de permissões JSON que você anexa a uma identidade (usuário, grupo ou função). Essas políticas controlam quais ações as identidades podem realizar, em quais recursos e sob quais condições. Para saber como criar uma política baseada em identidade, consulte Definir permissões personalizadas do IAM com as políticas gerenciadas pelo cliente no Guia do Usuário do IAM.

Identity-based as políticas podem ser políticas em linha (incorporadas diretamente em uma única identidade) ou políticas gerenciadas (políticas autônomas anexadas a várias identidades). Para saber como escolher entre uma política gerenciada e políticas em linha, consulte Escolher entre políticas gerenciadas e políticas em linha no Guia do usuário do IAM.

Resource-based políticas

Resource-based políticas são documentos de política JSON que você anexa a um recurso. Entre os exemplos estão políticas de confiança de perfil do IAM e políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. É necessário especificar uma entidade principal em uma política baseada em recursos.

Resource-based políticas são políticas em linha que estão localizadas nesse serviço. Você não pode usar políticas AWS gerenciadas do IAM em uma política baseada em recursos.

Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais que podem definir as permissões máximas concedidas por tipos de políticas mais comuns:

- Limites de permissões: definem o número máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM. Para saber mais sobre limites de permissões, consulte Limites de permissões para identidades do IAM no Guia do usuário do IAM.

- Políticas de Controle de Serviços (SCPs): as SCPs especificam o número máximo de permissões para uma organização ou uma unidade organizacional no AWS Organizations. Para saber mais, consulte Políticas de controle de serviço no Guia do usuário do AWS Organizations .
- Políticas de controle de recursos (RCPs): definem o número máximo de permissões disponíveis para recursos em suas contas. Consulte mais informações em Resource control policies (RCPs) no Guia do usuário do AWS Organizations .
- Políticas de sessão: políticas avançadas transmitidas como um parâmetro durante a criação de uma sessão temporária para um perfil ou um usuário federado. Para saber mais, consulte Políticas de sessão no Guia do usuário do IAM.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como AWS determinar se uma solicitação deve ser permitida quando vários tipos de políticas estão envolvidos, consulte Lógica de avaliação de políticas no Guia do usuário do IAM.

Como o Amazon DocumentDB funciona com o IAM

Antes de usar o IAM para gerenciar o acesso ao Amazon DocumentDB, entenda que atributos do IAM estão disponíveis para uso com o Amazon DocumentDB.

Atributos do IAM que você pode usar com o Amazon DocumentDB

Recurso do IAM	Instance-based aglomerados	Clusters elásticos
Identity-based políticas	Sim	Sim
Resource-based políticas	Não	Não
Ações de políticas	Sim	Sim
Recursos de políticas	Sim	Sim
Chaves de condição de política (específicas do serviço)	Sim	Sim

Recurso do IAM	Instance-based aglomerados	Clusters elásticos
ACLs	Não	Não
ABAC (tags em políticas)	Parcial	Sim
Credenciais temporárias	Sim	Sim
Permissões de entidade principal	Sim	Sim
Perfis de serviço	Sim	Sim
Service-linked papéis	Não	Sim

Para obter uma visão de alto nível de como o Amazon DocumentDB e outros AWS serviços funcionam com a maioria dos recursos do IAM, consulte AWS os serviços que funcionam com o IAM no Guia do usuário do IAM.

Identity-based políticas para o Amazon DocumentDB

Compatível com políticas baseadas em identidade: sim

Identity-based políticas são documentos de política de permissões JSON que você pode anexar a uma identidade, como um usuário do IAM, grupo de usuários ou função. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais atributos e em que condições. Para saber como criar uma política baseada em identidade, consulte Definir permissões personalizadas do IAM com as políticas gerenciadas pelo cliente no Guia do Usuário do IAM.

Com as políticas baseadas em identidade do IAM, é possível especificar ações e recursos permitidos ou negados, assim como as condições sob as quais as ações são permitidas ou negadas. Para saber mais sobre todos os elementos que podem ser usados em uma política JSON, consulte Referência de elemento de política JSON do IAM no Guia do usuário do IAM.

Identity-based exemplos de políticas para o Amazon DocumentDB

Para visualizar exemplos de políticas baseadas em identidade do Amazon DocumentDB, consulte Identity-based exemplos de políticas para o Amazon DocumentDB.

Resource-based políticas no Amazon DocumentDB

Compatibilidade com políticas baseadas em recursos: não

Resource-based políticas são documentos de política JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de perfil do IAM e as políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o atributo ao qual a política está anexada, a política define quais ações uma entidade principal especificado pode executar nesse atributo e em que condições. É necessário especificar uma entidade principal em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Para permitir o acesso entre contas, é possível especificar uma conta inteira ou as entidades do IAM em outra conta como a entidade principal em uma política baseada em recursos. Consulte mais informações em Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Ações políticas para Amazon DocumentDB

Compatível com ações de políticas: sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso a quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Action` de uma política JSON descreve as ações que podem ser usadas para permitir ou negar acesso em uma política. Incluem ações em uma política para conceder permissões para executar a operação associada.

> **Note**
>
> Para alguns recursos de gerenciamento, o Amazon DocumentDB usa a tecnologia operacional que é compartilhada com o Amazon Relational Database Service (Amazon RDS).
> Para ver uma lista das ações do Amazon RDS, consulte Ações definidas pelo serviço de banco de dados relacional da Amazon na Referência de autorização de serviço.
> Para visualizar ações de políticas para clusters elásticos do Amazon DocumentDB, consulte Ações definidas pelos clusters elásticos do Amazon DocumentDB na Referência de autorização de serviço.

As ações de política no Amazon DocumentDB usam o seguinte prefixo antes da ação:

```
aws
```

Para especificar várias ações em uma única declaração, separe-as com vírgulas.

```
"Action": [
      "aws:action1",
      "aws:action2"
         ]
```

Para visualizar exemplos de políticas baseadas em identidade do Amazon DocumentDB, consulte Identity-based exemplos de políticas para o Amazon DocumentDB.

Recursos de política para Amazon DocumentDB

Compatível com recursos de políticas: sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso a quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento de política JSON `Resource` especifica o objeto ou os objetos aos quais a ação se aplica. Como prática recomendada, especifique um recurso usando seu nome do recurso da Amazon (ARN). Para ações que não oferecem compatibilidade com permissões em nível de recurso, use um curinga (*) para indicar que a instrução se aplica a todos os recursos.

```
"Resource": "*"
```

> **Note**
>
> Para alguns recursos de gerenciamento, o Amazon DocumentDB usa a tecnologia operacional que é compartilhada com o Amazon Relational Database Service (Amazon RDS).
> Para ver uma lista dos tipos de recursos RDS e seus ARNs, consulte Recursos definidos pelo serviço de banco de dados relacional da Amazon na Referência de autorização do serviço. Para saber com quais ações você pode especificar o ARN de cada recurso, consulte Ações definidas pelo serviço do banco de dados relacional da Amazon.

Para visualizar os tipos de recursos dos clusters elásticos do Amazon DocumentDB, consulte Tipos de recursos definidos pelos clusters elásticos do Amazon DocumentDB na Referência de Autorização de Serviço.

Para visualizar exemplos de políticas baseadas em identidade do Amazon DocumentDB, consulte Identity-based exemplos de políticas para o Amazon DocumentDB.

Chaves de condição de política para o Amazon DocumentDB

Compatível com chaves de condição de política específicas de serviço: sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso a quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Condition` especifica quando as instruções são executadas com base em critérios definidos. É possível criar expressões condicionais que usem agentes de condição, como “igual a” ou “menor que”, para fazer a condição da política corresponder aos valores na solicitação. Para ver todas as chaves de condição AWS globais, consulte as chaves de contexto de condição AWS global no Guia do usuário do IAM.

Note

Para alguns recursos de gerenciamento, o Amazon DocumentDB usa a tecnologia operacional que é compartilhada com o Amazon Relational Database Service (Amazon RDS).
Para ver uma lista de chaves de condição do RDS, consulte Chaves de condição para o serviço do banco de dados relacional da Amazon na Referência de autorização de serviço.
Para saber com quais ações e recursos você pode usar a chave de condição, consulte Ações definidas pelo serviço do banco de dados relacional da Amazon.
Para visualizar as chaves de condição para clusters elásticos do Amazon DocumentDB, consulte Chaves de condição para clusters elásticos do Amazon DocumentDB na Referência de autorização de serviço.

Para visualizar exemplos de políticas baseadas em identidade do Amazon DocumentDB, consulte Identity-based exemplos de políticas para o Amazon DocumentDB.

ACLs no Amazon DocumentDB

Compatível com ACLs: não

As listas de controle de acesso (ACLs) controlam quais entidades principais (membros, usuários ou perfis da conta) têm permissões para acessar um recurso. As ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

ABAC com o Amazon DocumentDB

> **Note**
>
> O ABAC tem suporte apenas parcialmente para clusters baseados em instâncias, mas é compatível com clusters elásticos.

Attribute-based o controle de acesso (ABAC) é uma estratégia de autorização que define permissões com base em atributos chamados tags. Você pode anexar tags a entidades e AWS recursos do IAM e, em seguida, criar políticas ABAC para permitir operações quando a tag do principal corresponder à tag no recurso.

Para controlar o acesso baseado em tags, forneça informações sobre as tags no elemento de condição de uma política usando as `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* ou chaves de condição `aws:TagKeys`.

Se um serviço for compatível com as três chaves de condição para cada tipo de recurso, o valor será Sim para o serviço. Se um serviço for compatível com as três chaves de condição somente para alguns tipos de recursos, o valor será Parcial

Para saber mais sobre o ABAC, consulte Definir permissões com autorização do ABAC no Guia do usuário do IAM. Para visualizar um tutorial com etapas para configurar o ABAC, consulte Usar controle de acesso por atributo (ABAC) no Guia do usuário do IAM.

Usar credenciais temporárias com o Amazon DocumentDB

Compatível com credenciais temporárias: sim

As credenciais temporárias fornecem acesso de curto prazo aos AWS recursos e são criadas automaticamente quando você usa federação ou troca de funções. AWS recomenda que você gere

dinamicamente credenciais temporárias em vez de usar chaves de acesso de longo prazo. Para ter mais informações, consulte Credenciais de segurança temporárias no IAM e Serviços da Serviços da AWS que funcionam com o IAM no Guia do usuário do IAM.

Cross-service permissões principais para o Amazon DocumentDB

Compatibilidade com o recurso de encaminhamento de sessões de acesso (FAS): sim

As sessões de acesso direto (FAS) usam as permissões do principal que chama an AWS service (Serviço da AWS), combinadas com a solicitação AWS service (Serviço da AWS) para fazer solicitações aos serviços downstream. Para obter detalhes da política ao fazer solicitações de FAS, consulte Sessões de acesso direto.

Perfis de serviço para Amazon DocumentDB

Compatível com perfis de serviço: sim

O perfil de serviço é um perfil do IAM que um serviço assume para executar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para saber mais, consulte Criar um perfil para delegar permissões a um AWS service (Serviço da AWS) no Guia do Usuário do IAM.

> **Warning**
>
> A alteração das permissões de um perfil de serviço pode interromper a funcionalidade do Amazon DocumentDB. Edite perfis de serviço somente quando o Amazon DocumentDB fornecer orientação para isso.

Service-linked funções para o Amazon DocumentDB

> **Note**
>
> Service-linked as funções não são compatíveis com clusters baseados em instâncias, mas são compatíveis com clusters elásticos.

Uma função vinculada a serviços é um tipo de função de serviço vinculada a um. AWS service (Serviço da AWS) O serviço pode assumir a função de realizar uma ação em seu nome. Service-

linked as funções aparecem em você Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não editar as permissões para perfis vinculados ao serviço.

Para obter detalhes sobre como criar ou gerenciar perfis vinculados a serviços, consulte Serviços da AWS que funcionam com o IAM. Encontre um serviço na tabela que inclui um `Yes` na coluna de Service-linked função. Escolha o link Sim para visualizar a documentação do perfil vinculado a serviço desse serviço.

Identity-based exemplos de políticas para o Amazon DocumentDB

Por padrão, usuários e perfis não têm permissão para criar ou modificar recursos do Amazon DocumentDB. Para conceder permissão aos usuários para executar ações nos recursos que eles precisam, um administrador do IAM pode criar políticas do IAM.

Para aprender a criar uma política baseada em identidade do IAM ao usar esses documentos de política em JSON de exemplo, consulte Criar políticas do IAM (console) no Guia do usuário do IAM.

Para obter detalhes sobre ações e tipos de recursos definidos pelo Amazon DocumentDB, incluindo o formato dos ARNs para cada um dos tipos de recursos, consulte Ações, recursos e chaves de condição para o serviço do banco de dados relacional da Amazon na Referência de autorização do serviço.

> **Note**
>
> Para alguns recursos de gerenciamento, o Amazon DocumentDB usa a tecnologia operacional que é compartilhada com o Amazon Relational Database Service (Amazon RDS).
> Para ações de políticas para clusters elásticos do Amazon DocumentDB, consulte Ações, recursos e chaves de condição para clusters elásticos do Amazon DocumentDB na Referência de autorização de serviço.

Tópicos

- Práticas recomendadas de política
- Usar o console do Amazon DocumentDB
- Permitir que os usuários visualizem suas próprias permissões

Práticas recomendadas de política

Identity-based as políticas determinam se alguém pode criar, acessar ou excluir recursos do Amazon DocumentDB em sua conta. Essas ações podem incorrer em custos para sua Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com políticas AWS gerenciadas e avance para permissões com privilégios mínimos — Para começar a conceder permissões a seus usuários e cargas de trabalho, use as políticas AWS gerenciadas que concedem permissões para muitos casos de uso comuns. Eles estão disponíveis em seu Conta da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo AWS cliente que sejam específicas para seus casos de uso. Para saber mais, consulte Políticas gerenciadas pela AWS ou Políticas gerenciadas pela AWS para funções de trabalho no Guia do usuário do IAM.
- Aplique permissões de privilégio mínimo: ao definir permissões com as políticas do IAM, conceda apenas as permissões necessárias para executar uma tarefa. Você faz isso definindo as ações que podem ser executadas em recursos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para saber mais sobre como usar o IAM para aplicar permissões, consulte Políticas e permissões no IAM no Guia do usuário do IAM.
- Use condições nas políticas do IAM para restringir ainda mais o acesso: é possível adicionar uma condição às políticas para limitar o acesso a ações e recursos. Por exemplo, é possível escrever uma condição de política para especificar que todas as solicitações devem ser enviadas usando SSL. Você também pode usar condições para conceder acesso às ações de serviço se elas forem usadas por meio de um específico AWS service (Serviço da AWS), como CloudFormation. Para saber mais, consulte Elementos da política JSON do IAM: condição no Guia do usuário do IAM.
- Use o IAM Access Analyzer para validar suas políticas do IAM a fim de garantir permissões seguras e funcionais: o IAM Access Analyzer valida as políticas novas e existentes para que elas sigam a linguagem de política do IAM (JSON) e as práticas recomendadas do IAM. O IAM Access Analyzer oferece mais de cem verificações de política e recomendações práticas para ajudar a criar políticas seguras e funcionais. Para saber mais, consulte Validação de políticas do IAM Access Analyzer no Guia do Usuário do IAM.
- Exigir autenticação multifator (MFA) — Se você tiver um cenário que exija usuários do IAM ou um usuário raiz no seu Conta da AWS, ative o MFA para obter mais segurança. Para exigir MFA quando as operações de API forem chamadas, adicione condições de MFA às suas políticas. Para saber mais, consulte Configuração de acesso à API protegido por MFA no Guia do Usuário do IAM.

Para saber mais sobre as práticas recomendadas do IAM, consulte Práticas recomendadas de segurança no IAM no Guia do usuário do IAM.

Usar o console do Amazon DocumentDB

Para acessar o console da Amazon DocumentDB (compatível com MongoDB), você deve ter um conjunto mínimo de permissões. Essas permissões devem permitir que você liste e visualize detalhes sobre os recursos do Amazon DocumentDB em seu Conta da AWS. Caso crie uma política baseada em identidade mais restritiva que as permissões mínimas necessárias, o console não funcionará como pretendido para entidades (usuários ou perfis) com essa política.

Você não precisa permitir permissões mínimas de console para usuários que estão fazendo chamadas somente para a API AWS CLI ou para a AWS API. Em vez disso, permita o acesso somente a ações que correspondam à operação de API que estiverem tentando executar.

Para garantir que usuários e funções ainda possam usar o console do Amazon DocumentDB, anexe também o Amazon DocumentDB *ConsoleAccess* ou a política *ReadOnly* AWS gerenciada às entidades. Para obter informações, consulte Adicionar permissões a um usuário no Guia do usuário do IAM.

Permitir que os usuários visualizem suas próprias permissões

Este exemplo mostra como criar uma política que permita que os usuários do IAM visualizem as políticas gerenciadas e em linha anexadas a sua identidade de usuário. Essa política inclui permissões para concluir essa ação no console ou usando programaticamente a API AWS CLI ou 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": "*"
        }
    ]
}
```

Solução de problemas de identidade e acesso da Amazon DocumentDB

Use as informações a seguir para ajudar a diagnosticar e corrigir problemas comuns que você possa encontrar ao trabalhar com a Amazon DocumentDB e o IAM.

Tópicos

- Não tenho autorização para executar uma ação no Amazon DocumentDB
- Não estou autorizado a realizar iam: PassRole
- Quero permitir que pessoas saiam da minha Conta da AWS para acessar meus recursos do Amazon DocumentDB

Não tenho autorização para executar uma ação no Amazon DocumentDB

Se você receber uma mensagem de erro informando que não tem autorização para executar uma ação, suas políticas deverão ser atualizadas para permitir que você realize a ação.

O erro do exemplo a seguir ocorre quando o usuário do IAM `mateojackson` tenta usar o console para visualizar detalhes sobre um atributo *`my-example-widget`* fictício, mas não tem as permissões `aws:`*`GetWidget`* fictícias.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 aws:GetWidget on resource: my-example-widget
```

Nesse caso, a política do usuário `mateojackson` deve ser atualizada para permitir o acesso ao recurso *`my-example-widget`* usando a ação `aws:GetWidget`.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Não estou autorizado a realizar iam: PassRole

Caso receba uma mensagem de erro informando que você não tem autorização para executar a ação `iam:PassRole`, as políticas deverão ser atualizadas para permitir a transmissão de um perfil ao Amazon DocumentDB.

Alguns Serviços da AWS permitem que você passe uma função existente para esse serviço em vez de criar uma nova função de serviço ou uma função vinculada ao serviço. Para fazer isso, é preciso ter permissões para passar o perfil para o serviço.

O erro exemplificado a seguir ocorre quando um usuário do IAM chamada `marymajor` tenta usar o console para executar uma ação no Amazon DocumentDB. No entanto, a ação exige que o serviço tenha permissões concedidas por um perfil de serviço. Mary não tem permissões para passar o perfil para o serviço.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Nesse caso, as políticas de Mary devem ser atualizadas para permitir que ela realize a ação `iam:PassRole`.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Quero permitir que pessoas saiam da minha Conta da AWS para acessar meus recursos do Amazon DocumentDB

É possível criar um perfil que os usuários de outras contas ou pessoas fora da organização podem usar para acessar seus recursos. É possível especificar quem é confiável para assumir o perfil. Para serviços que oferecem compatibilidade com políticas baseadas em recursos ou listas de controle de acesso (ACLs), é possível usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte:

- Para saber se o Amazon DocumentDB é compatível com esses atributos, consulte Como o Amazon DocumentDB funciona com o IAM.
- Para saber como fornecer acesso aos recursos Contas da AWS que você possui, consulte Como fornecer acesso a um usuário do IAM em outro Conta da AWS usuário de sua propriedade no Guia do usuário do IAM.
- Para saber como fornecer acesso aos seus recursos a terceiros Contas da AWS, consulte Fornecendo acesso Contas da AWS a terceiros no Guia do usuário do IAM.
- Para saber como conceder acesso por meio da federação de identidades, consulte Conceder acesso a usuários autenticados externamente (federação de identidades) no Guia do usuário do IAM.
- Para saber a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Gerenciar permissões de acesso aos recursos do Amazon DocumentDB

Cada AWS recurso é de propriedade de um Conta da AWS, e as permissões para criar ou acessar os recursos são regidas por políticas de permissões. Um administrador de conta pode anexar políticas de permissões às identidades do IAM (ou seja, usuários, grupos e funções), e alguns serviços (como AWS Lambda) também oferecem suporte à anexação de políticas de permissões aos recursos.

> **Note**
>
> O administrador de uma conta (ou o usuário administrador) é um usuário com permissões de administrador. Para obter mais informações, consulte Práticas recomendadas de segurança no IAM no Guia do usuário do IAM.

Tópicos

- Recursos e operações do Amazon DocumentDB
- Informações sobre propriedade de recursos
- Gerenciar acesso aos recursos da
- Especificar elementos da política: ações, efeitos, recursos e entidades principais

- Especificar condições em uma política

Recursos e operações do Amazon DocumentDB

No Amazon DocumentDB, o principal recurso é um cluster. O Amazon DocumentDB oferece suporte a outros recursos que podem ser usados com o recurso principal, como instâncias, grupos de parâmetros, e assinaturas de eventos. Esses recursos são chamados de sub-recursos.

Esses recursos e sub-recursos têm Nomes de recurso da Amazon (ARNs) exclusivos associados a eles, conforme mostrado na tabela a seguir.

Tipo de recurso	Formato de Nome de região da Amazon (ARN)
Cluster	`arn:aws:rds:` *`region`*`:`*`account-id`* `:cluster:` *`db-cluster-name`*
Grupo de parâmetros do cluster	`arn:aws:rds:` *`region`*`:`*`account-id`* `:cluster-pg:` *`cluster-parameter-group-name`*
Snapshot de cluster	`arn:aws:rds:` *`region`*`:`*`account-id`* `:cluster-snapshot:` *`cluster-snapshot-name`*
Instância	`arn:aws:rds:` *`region`*`:`*`account-id`* `:db:`*`db-instance-name`*
Grupo de segurança	`arn:aws:rds:` *`region`*`:`*`account-id`* `:secgrp:`*`security-group-name`*
Grupo de sub-redes	`arn:aws:rds:` *`region`*`:`*`account-id`* `:subgrp:`*`subnet-group-name`*

O Amazon DocumentDB fornece um conjunto de operações para trabalhar com recursos do Amazon DocumentDB. Para obter uma lista das operações disponíveis, consulte Ações.

Informações sobre propriedade de recursos

O proprietário do recurso é Conta da AWS aquele que criou um recurso. Ou seja, o proprietário Conta da AWS do recurso é a entidade principal (a conta raiz, um usuário do IAM ou uma função do IAM) que autentica a solicitação que cria o recurso. Os seguintes exemplos mostram como isso funciona:

- Se você usar as credenciais da sua conta raiz Conta da AWS para criar um recurso do Amazon DocumentDB, como uma instância, você Conta da AWS é o proprietário do recurso Amazon DocumentDB.
- Se você criar um usuário do IAM em seu Conta da AWS e conceder permissões para criar recursos do Amazon DocumentDB para esse usuário, o usuário poderá criar recursos do Amazon DocumentDB. No entanto, você Conta da AWS, ao qual o usuário pertence, possui os recursos do Amazon DocumentDB.
- Se você criar uma função do IAM em sua Conta da AWS com permissões para criar recursos do Amazon DocumentDB, qualquer pessoa que possa assumir a função poderá criar recursos do Amazon DocumentDB. Você Conta da AWS, ao qual a função pertence, é dono dos recursos do Amazon DocumentDB.

Gerenciar acesso aos recursos da

Uma política de permissões descreve quem tem acesso a quê. A seção a seguir explica as opções disponíveis para a criação de políticas de permissões.

> **Note**
>
> Esta seção discute o uso do IAM no contexto do Amazon DocumentDB. Não são fornecidas informações detalhadas sobre o serviço IAM. Para obter a documentação completa do IAM, consulte O que é o IAM? no Guia do usuário do IAM. Para obter mais informações sobre a sintaxe e as descrições da política do IAM, consulte a Referência de política do AWS IAM no Guia do usuário do IAM.

As políticas anexadas a uma identidade do IAM são chamadas de políticas baseadas em identidade (políticas do IAM). As políticas anexadas a um recurso são chamadas de políticas baseadas em recursos. O Amazon DocumentDB oferece suporte apenas a políticas baseadas em identidade (políticas do IAM).

Tópicos

- Identity-based políticas (políticas do IAM)
- Resource-based políticas

Identity-based políticas (políticas do IAM)

É possível anexar políticas a identidades do IAM. Por exemplo, você pode fazer o seguinte:

- Anexar uma política de permissões a um usuário ou a um grupo em sua conta – um administrador da conta pode usar uma política de permissões associada a um determinado usuário a fim de conceder permissões para que o usuário crie um recurso do Amazon DocumentDB, como uma instância.
- Anexar uma política de permissões a uma função: você pode anexar uma política de permissões baseada em identidade a um perfil do IAM para conceder permissões entre contas. Por exemplo, um administrador pode criar uma função para conceder permissões entre contas a outra pessoa Conta da AWS ou a um AWS serviço da seguinte forma:
  1. Um administrador da Conta A cria uma função do IAM e anexa uma política de permissões à função que concede permissões em recursos da Conta A.
  2. Um administrador da Conta A anexa uma política de confiança à função identificando a Conta B como a entidade principal, que pode assumir a função.
  3. O administrador da Conta B pode então delegar permissões para assumir a função a qualquer usuário na Conta B. Isso permite que os usuários da Conta B criem ou acessem recursos na Conta A. O principal na política de confiança também pode ser um diretor de AWS serviço se você quiser conceder permissões a um AWS serviço para assumir a função.

  Para obter mais informações sobre o uso do IAM para delegar permissões, consulte Gerenciamento de acesso no Guia do usuário do IAM.

Veja a seguir um exemplo de política que permite ao usuário com o ID `123456789012` para criar instâncias para o seu Conta da AWS. A nova instância deve usar um grupo de opções e um parameter group que começa com `default` e deve usar o grupo de sub-redes `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 obter mais informações sobre o uso de políticas baseadas em identidade com o Amazon DocumentDB, consulte Usar políticas baseadas em identidade (políticas do IAM) para o Amazon DocumentDB. Para obter mais informações sobre usuários, grupos, funções e permissões, consulte Identidades (usuários, grupos e funções) no Guia do usuário do IAM.

Resource-based políticas

Outros serviços, como o Amazon Simple Storage Service (Amazon S3), são compatíveis com políticas de permissões baseadas em recursos. Por exemplo, você pode anexar uma política a um bucket do Amazon S3 para gerenciar permissões de acesso a esse bucket. O Amazon DocumentDB não oferece suporte a políticas baseadas em recursos.

Especificar elementos da política: ações, efeitos, recursos e entidades principais

Para cada recurso do Amazon DocumentDB, (consulte Recursos e operações do Amazon DocumentDB), o serviço define um conjunto de operações da API. Para obter mais informações, consulte Ações. Para conceder permissões para essas operações de API, o Amazon DocumentDB define um conjunto de ações que você pode especificar em uma política. A execução de uma operação de API pode exigir permissões para mais de uma ação.

Estes são os elementos de política básicos:

- Recurso: em uma política, você usa um Amazon Resource Name (ARN – Nome do recurso da Amazon) para identificar o recurso a que a política se aplica.
- Ação: você usa palavras-chave de ação para identificar operações de recursos que deseja permitir ou negar. Por exemplo, a permissão `rds:DescribeDBInstances` permite que o usuário execute a operação `DescribeDBInstances`.

- Efeito: você especifica o efeito quando o usuário solicita a ação específica, que pode ser permitir ou negar. Se você não conceder (permitir) explicitamente acesso a um recurso, o acesso estará implicitamente negado. Você também pode negar explicitamente o acesso a um recurso, para ter certeza de que um usuário não consiga acessá-lo, mesmo que uma política diferente conceda acesso.
- Entidade principal: em políticas baseadas em identidade (políticas do IAM), o usuário ao qual a política é anexada é a entidade principal implícita. Para as políticas baseadas em recursos, você especifica quais usuários, contas, serviços ou outras entidades deseja que recebam permissões (isso se aplica somente a políticas baseadas em recursos). O Amazon DocumentDB não oferece suporte a políticas baseadas em recursos.

Para saber mais sobre a sintaxe e as descrições da política do IAM, consulte a Referência de política do AWS IAM no Guia do usuário do IAM.

Para obter uma tabela que mostra todas as ações da API do Amazon DocumentDB e os recursos aos quais se aplicam, consulte Permissões da API do Amazon DocumentDB: referência de ações, recursos e condições.

Especificar condições em uma política

Ao conceder permissões, você pode usar a linguagem da política do IAM para especificar as condições de quando uma política deverá entrar em vigor. Por exemplo, é recomendável aplicar uma política somente após uma data específica. Para obter mais informações sobre como especificar condições em uma linguagem de política, consulte Condition no Guia do usuário do IAM.

Para expressar condições, você usa chaves de condição predefinidas. O Amazon DocumentDB não tem chaves de contexto de serviço específicas que possam ser usadas em uma política do IAM;. Para obter uma lista de chaves de contexto de condição global que estão disponíveis para todos os serviços, consulte Chaves de condição globais disponíveis no Guia do usuário do IAM.

Usar políticas baseadas em identidade (políticas do IAM) para o Amazon DocumentDB

> **Important**
>
> Para determinados atributos de gerenciamento, o Amazon DocumentDB usa a tecnologia operacional que é compartilhada com o Amazon RDS. As chamadas de console e API do

Amazon DocumentDB são registradas como chamadas feitas para a API do Amazon RDS.
AWS CLI
Primeiro, revise os tópicos introdutórios que explicam os conceitos básicos e as opções para gerenciar o acesso aos seus recursos do Amazon DocumentDB. Para obter mais informações, consulte Gerenciar permissões de acesso aos recursos do Amazon DocumentDB.

Este tópico fornece exemplos de políticas baseadas em identidade em que um administrador de conta pode anexar políticas de permissões a identidades do IAM (ou seja, usuários, grupos e funções).

A seguir há um exemplo de uma política do 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"
            ]
        }
    ]
}
```

A política inclui uma única instrução que especifica as seguintes permissões para o usuário do IAM:

- A política permite que o usuário do IAM crie uma instância usando a ação createDBInstance (isso também se aplica à operação https://docs.aws.amazon.com/cli/latest/reference/rds/create-db-instance.html AWS CLI create-db-instance e à). Console de gerenciamento da AWS
- O elemento `Resource` especifica que o usuário pode realizar ações em ou com recursos. Você especifica recursos usando um nome de recurso da Amazon (ARN). Esse ARN inclui o nome do serviço ao qual o recurso pertence (`rds`), o Região da AWS (`*`indica qualquer região neste exemplo), o número da conta do usuário (`123456789012`é a ID do usuário neste exemplo) e o tipo de recurso.

  O elemento `Resource` neste exemplo especifica as restrições da política a seguir em recursos para o usuário:

  - O identificador de instância para a nova instância deve começar com `test` (por exemplo, `testCustomerData1`, `test-region2-data`).
  - O grupo de parâmetros de cluster para a nova instância deve começar com `default`.
  - O grupo de sub-redes para a nova instância deve ser o grupo de sub-redes `default`.

A política não especifica o elemento `Principal` porque, em uma política baseada em identidade, a entidade principal que obtém as permissões não é especificada. Quando você anexar uma política um usuário, o usuário será a entidade principal implícita. Quando você anexa uma política de permissões a um perfil do IAM, a entidade principal identificada na política de confiança do perfil obtém as permissões.

Para obter uma tabela lista mostrando todas as operações da API do Amazon DocumentDB e os recursos aos quais elas se aplicam, consulte Permissões da API do Amazon DocumentDB: referência de ações, recursos e condições.

Permissões necessárias para usar o console do Amazon DocumentDB

Para um usuário trabalhar com o console Amazon DocumentDB, esse usuário deve ter um conjunto de permissões mínimo. Essas permissões permitem que o usuário descreva seus recursos do Amazon DocumentDB Conta da AWS e forneça outras informações relacionadas, incluindo informações de segurança e de rede do Amazon EC2.

Se você criar uma política do IAM que seja mais restritiva que as permissões mínimas necessárias, o console do não funcionará como pretendido para os usuários com essa política do IAM. Para garantir que esses usuários ainda consigam usar o console Amazon DocumentDB, associe também a política

gerenciada `AmazonDocDBConsoleFullAccess` ao usuário, conforme descrito em AWS políticas gerenciadas para o Amazon DocumentDB.

Você não precisa permitir permissões mínimas de console para usuários que estão fazendo chamadas somente para a API do Amazon DocumentDB AWS CLI ou para a API do Amazon DocumentDB.

Exemplos de política gerenciada pelo cliente

Nesta seção, você pode encontrar exemplos de políticas de usuário que concedem permissões para várias ações do Amazon DocumentDB. Essas políticas funcionam quando você está usando ações de API do Amazon DocumentDB, AWS SDKs ou o. AWS CLI Ao usar o console, você precisa conceder permissões adicionais específicas ao console, o que é abordado em Permissões necessárias para usar o console do Amazon DocumentDB.

Para alguns atributos de gerenciamento, o Amazon DocumentDB usa a tecnologia operacional que é compartilhada com o Amazon Relational Database Service (Amazon RDS) e o Amazon Neptune..

> **Note**
>
> Todos os exemplos usam a Leste dos EUA (N. da Virgínia) (`us-east-1`) e contêm IDs de conta fictícios.

Exemplos

- Exemplo 1: permitir que um usuário execute qualquer ação de descrição em qualquer recurso do Amazon DocumentDB
- Exemplo 2: impedir que um usuário exclua uma instância
- Exemplo 3: impedir que um usuário crie um cluster, a menos que a criptografia de armazenamento esteja ativada

Exemplo 1: permitir que um usuário execute qualquer ação de descrição em qualquer recurso do Amazon DocumentDB

A seguinte política de permissões concede permissões a um usuário para executar todas as ações que começam com `Describe`. Essas ações mostram informações sobre um recurso do Amazon DocumentDB, como uma instância. O caractere curinga (*) no elemento `Resource` indica que as ações são permitidas para todos os recursos do Amazon DocumentDB que pertencem à conta.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowRDSDescribe",
         "Effect":"Allow",
         "Action":"rds:Describe*",
         "Resource":"*"
      }
   ]
}
```

Exemplo 2: impedir que um usuário exclua uma instância

A seguinte política de permissões concede permissões para impedir que um usuário exclua uma instância específica. Por exemplo, você pode querer negar a capacidade de excluir suas instâncias de produção a qualquer usuário que não seja um 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"
      }
   ]
}
```

Exemplo 3: impedir que um usuário crie um cluster, a menos que a criptografia de armazenamento esteja ativada

A política de permissões a seguir nega a permissão de um usuário criar um cluster Amazon DocumentDB, a menos que a criptografia de armazenamento esteja habilitada.

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 gerenciadas para o Amazon DocumentDB

Para adicionar permissões a usuários, grupos e funções, é mais fácil usar políticas AWS gerenciadas do que criar políticas você mesmo. É necessário tempo e experiência para criar políticas gerenciadas pelo cliente do IAM que fornecem à sua equipe apenas as permissões de que precisam. Para começar rapidamente, você pode usar nossas políticas AWS gerenciadas. Essas políticas abrangem casos de uso comuns e estão disponíveis na sua Conta da AWS. Para obter mais informações sobre políticas AWS gerenciadas, consulte políticas AWS gerenciadas no Guia do usuário do IAM.

AWS os serviços mantêm e atualizam as políticas AWS gerenciadas. Você não pode alterar as permissões nas políticas AWS gerenciadas. Ocasionalmente, os serviços adicionam permissões adicionais a uma política AWS gerenciada para oferecer suporte a novos recursos. Esse tipo de atualização afeta todas as identidades (usuários, grupos e funções) em que a política está anexada. É mais provável que os serviços atualizem uma política AWS gerenciada quando um novo recurso é lançado ou quando novas operações são disponibilizadas. Os serviços não removem as permissões de uma política AWS gerenciada, portanto, as atualizações da política não quebrarão suas permissões existentes.

Além disso, AWS oferece suporte a políticas gerenciadas para funções de trabalho que abrangem vários serviços. Por exemplo, a política `ViewOnlyAccess` AWS gerenciada fornece acesso somente para leitura a vários AWS serviços e recursos. Quando um serviço lança um novo recurso, AWS adiciona permissões somente de leitura para novas operações e recursos. Para obter uma lista e descrições das políticas de perfis de trabalho, consulte Políticas gerenciadas pela AWS para perfis de trabalho no Guia do usuário do IAM.

As seguintes políticas AWS gerenciadas, que você pode anexar aos usuários em sua conta, são específicas do Amazon DocumentDB:

- AmazonDocDBFullAccess— Concede acesso total a todos os recursos do Amazon DocumentDB para a raiz Conta da AWS.
- AmazonDocDBReadOnlyAccess— Concede acesso somente para leitura a todos os recursos do Amazon DocumentDB para a raiz. Conta da AWS
- AmazonDocDBConsoleFullAccess – Concede acesso total para gerenciar os recursos de cluster elástico do Amazon DocumentDB e do Amazon DocumentDB usando o Console de gerenciamento da AWS.
- AmazonDocDBElasticReadOnlyAccess— Concede acesso somente para leitura a todos os recursos de cluster elástico do Amazon DocumentDB para a raiz. Conta da AWS
- AmazonDocDBElasticFullAccess— Concede acesso total a todos os recursos de cluster elástico do Amazon DocumentDB para a raiz Conta da AWS.

AmazonDocDBFullAccess

Essa política concede permissões administrativas que permitem que a entidade principal tenha acesso total a todas as ações do Amazon DocumentDB. As permissões nessa política são agrupadas da seguinte forma:

- As permissões do Amazon DocumentDB permitem todas as ações do Amazon DocumentDB.
- Algumas das permissões do Amazon EC2 nessa política são necessárias para validar os recursos transmitidos em uma solicitação de API. Isso serve para garantir que o Amazon DocumentDB seja capaz de usar adequadamente os recursos com um cluster. O restante das permissões do Amazon EC2 nesta política permitem que o Amazon DocumentDB crie os AWS recursos necessários para possibilitar a conexão com seus clusters.

- As permissões do Amazon DocumentDB são usadas para validar os recursos transmitidos em uma solicitação durante as chamadas de API. Elas são necessárias para que o Amazon DocumentDB consiga usar a chave transmitida com o cluster do Amazon DocumentDB.
- Os CloudWatch registros são necessários para que o Amazon DocumentDB possa garantir que os destinos de entrega de logs estejam acessíveis e que sejam válidos para uso de logs do broker.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "rds:AddRoleToDBCluster",
                "rds:AddSourceIdentifierToSubscription",
                "rds:AddTagsToResource",
                "rds:ApplyPendingMaintenanceAction",
                "rds:CopyDBClusterParameterGroup",
                "rds:CopyDBClusterSnapshot",
                "rds:CopyDBParameterGroup",
                "rds:CreateDBCluster",
                "rds:CreateDBClusterParameterGroup",
                "rds:CreateDBClusterSnapshot",
                "rds:CreateDBInstance",
                "rds:CreateDBParameterGroup",
                "rds:CreateDBSubnetGroup",
                "rds:CreateEventSubscription",
                "rds:DeleteDBCluster",
                "rds:DeleteDBClusterParameterGroup",
                "rds:DeleteDBClusterSnapshot",
                "rds:DeleteDBInstance",
                "rds:DeleteDBParameterGroup",
                "rds:DeleteDBSubnetGroup",
                "rds:DeleteEventSubscription",
                "rds:DescribeAccountAttributes",
                "rds:DescribeCertificates",
                "rds:DescribeDBClusterParameterGroups",
                "rds:DescribeDBClusterParameters",
                "rds:DescribeDBClusterSnapshotAttributes",
                "rds:DescribeDBClusterSnapshots",
                "rds:DescribeDBClusters",
```

```
                "rds:DescribeDBEngineVersions",
                "rds:DescribeDBInstances",
                "rds:DescribeDBLogFiles",
                "rds:DescribeDBParameterGroups",
                "rds:DescribeDBParameters",
                "rds:DescribeDBSecurityGroups",
                "rds:DescribeDBSubnetGroups",
                "rds:DescribeEngineDefaultClusterParameters",
                "rds:DescribeEngineDefaultParameters",
                "rds:DescribeEventCategories",
                "rds:DescribeEventSubscriptions",
                "rds:DescribeEvents",
                "rds:DescribeOptionGroups",
                "rds:DescribeOrderableDBInstanceOptions",
                "rds:DescribePendingMaintenanceActions",
                "rds:DescribeValidDBInstanceModifications",
                "rds:DownloadDBLogFilePortion",
                "rds:FailoverDBCluster",
                "rds:ListTagsForResource",
                "rds:ModifyDBCluster",
                "rds:ModifyDBClusterParameterGroup",
                "rds:ModifyDBClusterSnapshotAttribute",
                "rds:ModifyDBInstance",
                "rds:ModifyDBParameterGroup",
                "rds:ModifyDBSubnetGroup",
                "rds:ModifyEventSubscription",
                "rds:PromoteReadReplicaDBCluster",
                "rds:RebootDBInstance",
                "rds:RemoveRoleFromDBCluster",
                "rds:RemoveSourceIdentifierFromSubscription",
                "rds:RemoveTagsFromResource",
                "rds:ResetDBClusterParameterGroup",
                "rds:ResetDBParameterGroup",
                "rds:RestoreDBClusterFromSnapshot",
                "rds:RestoreDBClusterToPointInTime"
            ],
            "Effect": "Allow",
            "Resource": [
                "*"
            ]
        },
        {
            "Action": [
                "cloudwatch:GetMetricStatistics",
```

```
                "cloudwatch:ListMetrics",
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeVpcs",
                "kms:ListAliases",
                "kms:ListKeyPolicies",
                "kms:ListKeys",
                "kms:ListRetirableGrants",
                "logs:DescribeLogStreams",
                "logs:GetLogEvents",
                "sns:ListSubscriptions",
                "sns:ListTopics",
                "sns:Publish"
            ],
            "Effect": "Allow",
            "Resource": [
                "*"
            ]
        },
        {
            "Action": "iam:CreateServiceLinkedRole",
            "Effect": "Allow",
            "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
            "Condition": {
                "StringLike": {
                    "iam:AWS ServiceName": "rds.amazonaws.com"
                }
            }
        }
    ]
}
```

AmazonDocDBReadOnlyAccess

Essa política concede permissões de acesso somente leitura que permitem que os usuários visualizem informações no Amazon DocumentDB. As entidades principais com essa política anexada não podem fazer nenhuma atualização ou excluir recursos existentes, nem criar novos recursos do Amazon DocumentDB. Por exemplo, entidades principais com essas permissões podem visualizar

a lista de clusters e configurações associadas à conta, mas não podem alterar a configuração ou as definições de nenhum cluster. As permissões nessa política são agrupadas da seguinte forma:

- As permissões do Amazon DocumentDB permitem que você liste os recursos do Amazon DocumentDB, descreva e obtenha informações sobre eles.
- As permissões do Amazon EC2 são usadas para descrever a Amazon VPC, sub-redes, grupos de segurança e ENIs associados a um cluster.
- A permissão do Amazon DocumentDB é usada para descrever a chave associada ao cluster.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "rds:DescribeAccountAttributes",
                "rds:DescribeCertificates",
                "rds:DescribeDBClusterParameterGroups",
                "rds:DescribeDBClusterParameters",
                "rds:DescribeDBClusterSnapshotAttributes",
                "rds:DescribeDBClusterSnapshots",
                "rds:DescribeDBClusters",
                "rds:DescribeDBEngineVersions",
                "rds:DescribeDBInstances",
                "rds:DescribeDBLogFiles",
                "rds:DescribeDBParameterGroups",
                "rds:DescribeDBParameters",
                "rds:DescribeDBSubnetGroups",
                "rds:DescribeEventCategories",
                "rds:DescribeEventSubscriptions",
                "rds:DescribeEvents",
                "rds:DescribeOrderableDBInstanceOptions",
                "rds:DescribePendingMaintenanceActions",
                "rds:DownloadDBLogFilePortion",
                "rds:ListTagsForResource"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
```

```
        {
            "Action": [
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeInternetGateways",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeVpcs"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "kms:ListKeys",
                "kms:ListRetirableGrants",
                "kms:ListAliases",
                "kms:ListKeyPolicies"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "logs:DescribeLogStreams",
                "logs:GetLogEvents"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:logs:*:*:log-group:/aws/rds/*:log-stream:*",
                "arn:aws:logs:*:*:log-group:/aws/docdb/*:log-stream:*"
            ]
        }
    ]
```

```
}
```

AmazonDocDBConsoleFullAccess

Concede acesso total para gerenciar os recursos do Amazon DocumentDB usando Console de gerenciamento da AWS o seguinte:

- As permissões do Amazon DocumentDB permitem todas as ações de cluster do Amazon DocumentDB e do Amazon DocumentDB.
- Algumas das permissões do Amazon EC2 nessa política são necessárias para validar os recursos transmitidos em uma solicitação de API. Isso serve para garantir que o Amazon DocumentDB seja capaz de usar adequadamente os recursos para provisionar e manter o cluster. O restante das permissões do Amazon EC2 nesta política permitem que o Amazon DocumentDB crie AWS os recursos necessários para possibilitar a conexão com seus clusters, como o VPCendpoint.
- AWS KMS as permissões são usadas durante chamadas de API AWS KMS para validar os recursos passados em uma solicitação. Elas são necessárias para que o Amazon DocumentDB consiga usar a chave transmitida para criptografar e descriptografar os dados em repouso com o cluster do Amazon DocumentDB.
- Os CloudWatch registros são necessários para que o Amazon DocumentDB possa garantir que os destinos de entrega de logs estejam acessíveis e que sejam válidos para auditoria e criação de perfil de uso de logs.
- As permissões do Secrets Manager são necessárias para validar determinado segredo e usá-lo para configurar o usuário administrador para clusters elásticos do Amazon DocumentDB.
- As permissões do Amazon RDS são necessárias para ações de gerenciamento de clusters do Amazon DocumentDB. Para determinados atributos de gerenciamento, o Amazon DocumentDB usa a tecnologia operacional que é compartilhada com o Amazon RDS.
- Permissões SNS permitem que as entidades principais acessem assinaturas e tópicos do Amazon Simple Notification Service (Amazon SNS) e publiquem mensagens do Amazon SNS.
- As permissões do IAM são necessárias para criar as funções vinculadas ao serviço necessárias para publicação de métricas e logs.

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

Essa política concede permissões de acesso somente para leitura que oferecem acesso à informação do cluster elástico no Amazon DocumentDB. As entidades principais com essa política anexada não podem fazer nenhuma atualização ou excluir recursos existentes, nem criar novos recursos do Amazon DocumentDB. Por exemplo, entidades principais com essas permissões podem visualizar a lista de clusters e configurações associadas à conta, mas não podem alterar a configuração ou as definições de nenhum cluster. As permissões nessa política são agrupadas da seguinte forma:

- As permissões de cluster elástico do Amazon DocumentDB permitem que você liste os recursos do cluster elástico Amazon DocumentDB, descreva-os e obtenha informações sobre eles.
- CloudWatch as permissões são usadas para verificar as métricas do serviço.

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

Essa política concede permissões administrativas que permitem que a entidade principal tenha acesso total a todas as ações do Amazon DocumentDB para cluster elástico do Amazon DocumentDB.

Essa política usa AWS tags (https://docs.aws.amazon.com/tag-editor/latest/userguide/tagging.html) dentro de condições para definir o escopo do acesso aos recursos. Se você estiver usando um segredo, ele deverá ser marcado com uma chave de tag `DocDBElasticFullAccess` e um valor de tag. Se você estiver usando uma chave gerenciada pelo cliente, ela deverá ser marcada com uma chave de tag `DocDBElasticFullAccess` e um valor de tag.

As permissões nessa política são agrupadas da seguinte forma:

- As permissões de cluster elástico do Amazon DocumentDB permitem todas as ações do Amazon DocumentDB.
- Algumas das permissões do Amazon EC2 nessa política são necessárias para validar os recursos transmitidos em uma solicitação de API. Isso serve para garantir que o Amazon DocumentDB seja capaz de usar adequadamente os recursos para provisionar e manter o cluster. O restante das permissões do Amazon EC2 nesta política permitem que o Amazon DocumentDB crie AWS os recursos necessários para possibilitar que você se conecte aos seus clusters, como um endpoint VPC.
- AWS KMS permissões são necessárias para que o Amazon DocumentDB possa usar a chave passada para criptografar e descriptografar os dados em repouso dentro do cluster elástico do Amazon DocumentDB.

> **Note**
>
> A chave gerenciada pelo cliente deve ter uma tag com chave `DocDBElasticFullAccess` e um valor de tag.

- SecretsManager permissões são necessárias para validar determinado segredo e usá-lo para configurar o usuário administrador para clusters elásticos do Amazon DocumentDB.

> **Note**
>
> O segredo usado deve ter uma tag com chave `DocDBElasticFullAccess` e um valor de tag.

- As permissões do IAM são necessárias para criar as funções vinculadas ao serviço necessárias para publicação de métricas e logs.

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

Você não pode se vincular `AmazonDocDBElasticServiceRolePolicy` às suas AWS Identity and Access Management entidades. Essa política é anexada a uma função vinculada ao serviço que permite ao Amazon DocumentDB realizar ações em seu nome. Para obter mais informações, consulte Service-linked funções em clusters elásticos.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": [
                        "AWS/DocDB-Elastic"
                    ]
                }
            }
        }
    ]
}
```

Atualizações do Amazon DocumentDB para AWS políticas gerenciadas

Alteração	Descrição	Data
AmazonDocDBElasticFullAccess, AmazonDocDBConsoleFullAccess - Alteração	Políticas atualizadas para adicionar ações de manutenção pendentes.	2/11/2025
AmazonDocDBElasticFullAccess, AmazonDocDBConsoleFullAccess - Alteração	Políticas atualizadas para adicionar start/stop cluster e copiar ações de snapshot de cluster.	2/21/2024
AmazonDocDBElasticReadOnlyAccess, AmazonDocDBElasticFullAccess - Alteração	Políticas atualizadas para adicionar `cloudwatch:GetMetricData` ações.	6/21/2023
AmazonDocDBElasticReadOnlyAccess: nova política	Nova política gerenciada para clusters elásticos do Amazon DocumentDB.	6/8/2023

Alteração	Descrição	Data
AmazonDocDBElastic FullAccess: nova política	Nova política gerenciada para clusters elásticos do Amazon DocumentDB.	6/5/2023
AmazonDocDB-Elasti cServiceRolePolicy – Nova política	O Amazon DocumentDB cria uma nova função vinculada a AWS ServiceRoleForDocD B-Elastic serviços para clusters elásticos do Amazon DocumentDB.	11/30/2022
AmazonDocDBConsole FullAccess - Alteração	Política atualizada para adicionar permissões de cluster globais e elásticas do Amazon DocumentDB.	11/30/2022
AmazonDocDBConsole FullAccess, AmazonDoc DBFullAccess, AmazonDoc DBReadOnlyAccess - Nova política	Inicialização do serviço.	1/19/2017

Permissões da API do Amazon DocumentDB: referência de ações, recursos e condições

Use as seções a seguir como referência ao configurar Usar políticas baseadas em identidade (políticas do IAM) para o Amazon DocumentDB e escrever políticas de permissões que você pode anexar a uma identidade do IAM (políticas com base em identidade).

O conteúdo a seguir lista cada operação da API do Amazon DocumentDB. Incluídas na lista estão as ações correspondentes para as quais você pode conceder permissões para realizar a ação, o AWS recurso para o qual você pode conceder as permissões e as chaves de condição que você pode incluir para um controle de acesso refinado. Você especifica as ações no campo `Action` da política, o valor de recurso no campo `Resource` da política e as condições no campo `Condition` da política. Para obter mais informações sobre as condições, consulte Especificar condições em uma política.

Você pode usar chaves AWS de condição abrangentes em suas políticas do Amazon DocumentDB para expressar condições. Para obter uma lista completa AWS de chaves amplas, consulte Chaves disponíveis no Guia do usuário do IAM.

É possível testar as políticas do IAM com o simulador de políticas do IAM Ele fornece automaticamente uma lista de recursos e parâmetros necessários para cada AWS ação, incluindo ações do Amazon DocumentDB. O simulador de políticas do IAM determina as permissões que são necessárias para cada uma das ações especificadas por você. Para obter informações sobre o simulador de políticas do IAM, consulte Teste de políticas do IAM com o simulador de políticas do IAM no Guia do usuário do IAM.

> **Note**
>
> Para especificar uma ação, use o prefixo `rds:` seguido do nome da operação da API (por exemplo, `rds:CreateDBInstance`).

O conteúdo a seguir lista operações de API do Amazon RDS e as ações, os recursos e as chaves de condição relacionados.

Tópicos

- Ações do Amazon DocumentDB que dão suporte a permissões no nível do recurso
- Ações do Amazon DocumentDB que não dão suporte a permissões no nível do recurso

Ações do Amazon DocumentDB que dão suporte a permissões no nível do recurso

Resource-level as permissões fornecem a capacidade de especificar os recursos nos quais os usuários podem realizar ações. O Amazon DocumentDB oferece suporte parcial para permissões em nível de recurso. Isso significa que, para determinadas ações do Amazon DocumentDB, é possível controlar quando os usuários têm permissão para usar essas ações com base em condições que precisam ser concluídas, ou em recursos específicos que os usuários têm permissão para usar. Por exemplo, você pode conceder a usuários permissão para modificar somente instâncias específicas.

O conteúdo a seguir lista operações de API do Amazon DocumentDB e as ações, os recursos e as chaves de condição relacionados.

> **Note**
>
> Para determinados atributos de gerenciamento, o Amazon DocumentDB usa a tecnologia operacional que é compartilhada com o Amazon RDS. Para obter mais ações e permissões do Amazon DocumentDB, consulte Ações, recursos e chaves de condição para o Amazon RDS na Referência de autorização do serviço.

Operações e ações da API do Amazon DocumentDB	Recursos	Chaves de condição
AddTagsToResource `rds:AddTagsToResource`	Instância `arn:aws:rds:` *`region`*`:`*`account-id`* `:db:`*`db-instance-name`*	`rds:db-tag`
	Grupo de sub-redes `arn:aws:rds:` *`region`*`:`*`account-id`* `:subgrp:`*`subnet-group-name`*	`rds:subgrp-tag`
ApplyPendingMaintenanceAction `rds:ApplyPendingMaintenanceAction`	Instância `arn:aws:rds:` *`region`*`:`*`account-id`* `:db:`*`db-instance-name`*	`rds:db-tag`
CopyDBClusterSnapshot `rds:CopyDBClusterSnapshot`	Snapshot de cluster `arn:aws:rds:` *`region`*`:`*`account-id`* `:cluster-snapshot:` *`cluster-snapshot-name`*	`rds:cluster-snapshot-tag`
CreateDBCluster	Cluster	`rds:cluster-tag`

Operações e ações da API do Amazon DocumentDB	Recursos	Chaves de condição
rds:CreateDBCluster	arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-name*	
	Grupo de parâmetros do cluster arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
	Grupo de sub-redes arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
CreateDBClusterParameterGroup rds:CreateDBClusterParameterGroup	Grupo de parâmetros do cluster arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
CreateDBClusterSnapshot rds:CreateDBClusterSnapshot	Cluster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-name*	rds:cluster-tag
	Snapshot de cluster arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag

Operações e ações da API do Amazon DocumentDB	Recursos	Chaves de condição
CreateDBInstance rds:CreateDBInstance	Instância arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:DatabaseClass rds:db-tag
	Cluster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-name*	rds:cluster-tag
CreateDBSubnetGroup rds:CreateDBSubnetGroup	Grupo de sub-redes arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
DeleteDBInstance rds:DeleteDBInstance	Instância arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
DeleteDBSubnetGroup rds:DeleteDBSubnetGroup	Grupo de sub-redes arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
DescribeDBClusterParameterGroups rds:DescribeDBClusterParameterGroups	Grupo de parâmetros do cluster arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag

Operações e ações da API do Amazon DocumentDB	Recursos	Chaves de condição
DescribeDBClusterParameters rds:DescribeDBClusterParameters	Grupo de parâmetros do cluster arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
DescribeDBClusters rds:DescribeDBClusters	Cluster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-instance-name*	rds:cluster-tag
DescribeDBClusterSnapshotAttributes rds:DescribeDBClusterSnapshotAttributes	Snapshot de cluster arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag
DescribeDBSubnetGroups rds:DescribeDBSubnetGroups	Grupo de sub-redes arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
DescribePendingMaintenanceActions rds:DescribePendingMaintenanceActions	Instância arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:DatabaseClass rds:db-tag

Operações e ações da API do Amazon DocumentDB	Recursos	Chaves de condição
FailoverDBCluster rds:FailoverDBCluster	Cluster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-instance-name*	rds:cluster-tag
ListTagsForResource rds:ListTagsForResource	Instância arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
	Grupo de sub-redes arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
ModifyDBCluster rds:ModifyDBCluster	Cluster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-name*	rds:cluster-tag
	Grupo de parâmetros do cluster arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
ModifyDBClusterParameterGroup rds:ModifyDBClusterParameterGroup	Grupo de parâmetros do cluster arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag

Operações e ações da API do Amazon DocumentDB	Recursos	Chaves de condição
ModifyDBClusterSnapshotAttribute rds:ModifyDBClusterSnapshotAttribute	Snapshot de cluster arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag
ModifyDBInstance rds:ModifyDBInstance	Instância arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:DatabaseClass rds:db-tag
RebootDBInstance rds:RebootDBInstance	Instância arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
RemoveTagsFromResource rds:RemoveTagsFromResource	Instância arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
	Grupo de sub-redes arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
ResetDBClusterParameterGroup rds:ResetDBClusterParameterGroup	Grupo de parâmetros do cluster arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag

Operações e ações da API do Amazon DocumentDB	Recursos	Chaves de condição
RestoreDBClusterFromSnapshot rds:RestoreDBClusterFromSnapshot	Cluster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-instance-name*	rds:cluster-tag
	Snapshot de cluster arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag
RestoreDBClusterToPointInTime rds:RestoreDBClusterToPointInTime	Cluster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-instance-name*	rds:cluster-tag
	Grupo de sub-redes arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag

Ações do Amazon DocumentDB que não dão suporte a permissões no nível do recurso

É possível usar todas as ações do em uma política do Amazon DocumentDB em uma política do IAM para conceder ou negar aos usuários permissão para usar essa ação. Contudo, nem todas as ações do Amazon DocumentDB dão suporte a permissões no nível do recurso, que permitem especificar os recursos nos quais uma ação pode ser realizada. As ações de API do Amazon DocumentDB a seguir não oferecem suporte a permissões em nível de recurso no momento. Por isso, para usar essas ações em uma política do IAM, você deve conceder a usuários permissão para usar todos os recursos para a ação usando um caractere curinga * para o elemento `Resource` na instrução.

- `rds:DescribeDBClusterSnapshots`
- `rds:DescribeDBInstances`

Cross-service prevenção adjunta confusa

O problema do “confused deputy” é um problema de segurança em que uma entidade que não tem permissão para executar uma ação pode coagir uma entidade mais privilegiada a executar a ação. Em AWS, a falsificação de identidade entre serviços pode resultar no confuso problema do delegado. Cross-service a representação pode ocorrer quando um serviço (o serviço de chamada) chama outro serviço (o serviço chamado). O serviço de chamada pode ser manipulado de modo a usar suas permissões para atuar nos recursos de outro cliente de uma forma na qual ele não deveria ter permissão para acessar. Para evitar isso, a AWS fornece ferramentas que ajudam você a proteger seus dados para todos os serviços com entidades principais de serviço que receberam acesso aos recursos em sua conta.

Use as chaves de contexto de condição global `aws:SourceAccount` e `aws:SourceArn` nas políticas de recursos para limitar as permissões que o Amazon DocumentDB oferece a outro serviço ao recurso. Use `aws:SourceArn` se quiser que apenas um recurso seja associado ao acesso entre serviços. Use `aws:SourceAccount` se quiser permitir que qualquer recurso nessa conta seja associado ao uso entre serviços.

A maneira mais eficaz de se proteger contra o problema do substituto confuso é usar a chave de contexto de condição global `aws:SourceArn` com o ARN completo do recurso. Se você não souber o ARN completo do recurso ou especificar vários recursos, use a chave de condição de contexto global `aws:SourceArn` com caracteres curinga (*) para as partes desconhecidas do ARN. Por exemplo, .`arn:aws:`*`servicename`*`:*:`*`123456789012`*`:*`

Se o valor de `aws:SourceArn` não contiver o ID da conta, como um ARN de bucket do Amazon S3, você deverá usar ambas as chaves de contexto de condição global para limitar as permissões.

O valor de `aws:SourceArn` deve ser cluster.

O exemplo a seguir mostra como você pode usar as chaves de contexto de condição `aws:SourceAccount` global `aws:SourceArn` e as chaves de contexto no Amazon DocumentDB para evitar o confuso problema do deputado.

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

Autenticação usando identidade do IAM

Os usuários e aplicações do Amazon DocumentDB podem usar usuários e funções do IAM para se autenticar em um cluster do Amazon DocumentDB. A autenticação de IAM do Amazon DocumentDB é um método de autenticação sem senha. Além disso, os aplicativos clientes não enviam os segredos da senha para o cluster Amazon DocumentDB ao usar o IAM roles/users. Em vez disso, as conexões do cliente são autenticadas AWS STS usando tokens de segurança temporários. Non-administrative usuários e aplicativos agora podem usar o mesmo ARN de identidade do IAM ao se conectar a diferentes clusters e outros AWS serviços do Amazon DocumentDB.

Você também pode optar por usar tanto a autenticação baseada em senha quanto a do IAM para autenticar usuários e aplicações em um cluster do Amazon DocumentDB. A autenticação do IAM está disponível somente na versão 5.0 do cluster baseado em instâncias do Amazon DocumentDB. A autenticação do IAM usando ARNs de identidade do IAM não tem suporte pelo usuário principal do Amazon DocumentDB.

> **Note**
>
> O usuário principal só pode ser autenticado usando a autenticação baseada em senha existente.

Tópicos

- Introdução à autenticação usando usuários e funções do IAM
- Configurar AWS tipos de computação para se autenticar no Amazon DocumentDB usando AWS IAM
- Monitorar solicitações de autenticação do IAM
- Usar a autenticação do IAM
- Drivers que oferecem suporte ao IAM
- Perguntas frequentes sobre autenticação de identidade do IAM

Introdução à autenticação usando usuários e funções do IAM

Os usuários e funções do Amazon DocumentDB com identidades do IAM são criados e gerenciados em um banco de dados. `$external`

Criar um usuário

Conecte-se como usuário principal e, em seguida, crie um usuário e um perfil do IAM:

```
use $external;
db.createUser(
    {
        user: "arn:aws:iam::123456789123:user/iamuser",
        mechanisms: ["MONGODB-AWS"],
        roles: [ { role: "readWrite", db: "readWriteDB" } ]
    }
);
```

Como alternativa, adicione um usuário do Amazon DocumentDB usando um perfil do IAM:

```
use $external;
db.createUser(
    {
        user: "arn:aws:iam::123456789123:role/iamrole",
        mechanisms: ["MONGODB-AWS"],
        roles: [ { role: "readWrite", db: "readWriteDB" } ]
    }
);
```

Modificar um usuário ou perfil do IAM

Modificar um usuário do IAM existente:

```
use $external;
db.updateUser(
  "arn:aws:iam::123456789123:user/iamuser",
  {
    roles: [ { role: "read", db: "readDB" } ]
  }
);
```

Modificar um perfil do IAM existente:

```
use $external;
db.updateUser(
  "arn:aws:iam::123456789123:role/iamrole",
  {
    roles: [ { role: "read", db: "readDB" } ]
  }
);
```

Para conceder ou revogar perfis de um usuário do 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" } ]
);
```

Para conceder ou revogar perfis de um perfil do 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" } ]
);
```

Eliminar um usuário ou perfil do IAM

Para descartar um usuário do IAM existente:

```
use $external;
db.dropUser("arn:aws:iam::123456789123:user/iamuser");
```

Para descartar um perfil do IAM existente:

```
use $external;
db.dropUser("arn:aws:iam::123456789123:role/iamrole");
```

Configurar um URI de conexão para autenticar usando o IAM AWS

Para autenticar usando o AWS IAM, use os seguintes parâmetros de URI: `authSource` as `$external` e `authMechanism` as`MONGODB-AWS`. Se você estiver usando um usuário do IAM, os campos de nome de usuário e senha serão substituídos por uma chave de acesso e uma chave secreta, respectivamente. Se você estiver assumindo um perfil do IAM, vinculada ao ambiente em que está (por exemplo, função AWS Lambda , instância do Amazon EC2). Não será necessário passar nenhuma credencial específica ao fazer a autenticação usando o mecanismo `MONGODB-AWS`. Se você estiver usando drivers do MongoDB que suportam o mecanismo de autenticação `MONGODB-AWS`, os drivers também podem recuperar as credenciais do perfil do IAM da instância de computação (por exemplo, Amazon EC2, função do Lambda e outras). O exemplo a seguir usa um shell mongo para autenticar usando `MONGODB-AWS`, passando manualmente uma chave de acesso e uma chave secreta (de um usuário do IAM) para demonstrar a autenticação no Amazon DocumentDB.

O exemplo a seguir usa o código Python para autenticar usando `MONGODB-AWS` sem passar explicitamente nenhuma credencial (usando uma função do IAM anexada ao ambiente) para demonstrar a autenticação no 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')
```

O exemplo a seguir usa um shell mongo para autenticar usando o mecanismo `MONGODB-AWS`, passando manualmente uma chave de acesso e uma chave secreta (de um usuário do IAM) para demonstrar a autenticação no Amazon DocumentDB.

```
$ mongo 'mongodb://<access_key>:<secret_key>@<cluster_endpoint>:<db_port>/test?
authSource=%24external&authMechanism=MONGODB-AWS'
```

O exemplo a seguir usa um shell mongo para autenticar usando `MONGODB-AWS` sem passar explicitamente nenhuma credencial (usando o perfil do IAM anexado ao ambiente) para demonstrar a autenticação no Amazon DocumentDB.

```
$ mongo 'mongodb://<cluster_endpoint>:<db_port>/test?authSource=
%24external&authMechanism=MONGODB-AWS'
```

Configurar AWS tipos de computação para se autenticar no Amazon DocumentDB usando AWS IAM

Usar o Amazon EC2/AWS Lambda/AWS Fargate

O Amazon EC2 usa as seguintes variáveis de ambiente. Se você tiver um perfil do IAM anexado à instância do EC2 ou um perfil do IAM de execução associado a uma função do Lambda ou a uma tarefa do Amazon ECS, essas variáveis serão preenchidas automaticamente e o driver poderá buscar esses valores do ambiente:

```
AWS_ACCESS_KEY_ID
AWS_SECRET_ACCESS_KEY
AWS_SESSION_TOKEN
```

Para obter informações sobre como usar variáveis de ambiente, consulte Usar variáveis de ambiente do Lambda no Guia do desenvolvedor do AWS Lambda .

Usar o Amazon EKS

A atribuição de um perfil aos pods do Amazon Elastic Kubernetes Service (Amazon EKS) configurará automaticamente as duas variáveis de ambiente a seguir:

```
AWS_WEB_IDENTITY_TOKEN_FILE - path of web identity token file
AWS_ROLE_ARN - Name of IAM role to connect with
```

Com a ajuda dessas variáveis, assuma manualmente a função do seu código usando a chamada do AWS SDK para`AssumeRoleWithWebIdentity`:

- Omita o parâmetro `ProviderID`.
- Encontre o valor do parâmetro `WebIdentityToken` no arquivo descrito na variável de ambiente `AWS_WEB_IDENTITY_TOKEN_FILE`.

Para obter mais informações, consulte O que é o Amazon EKS no Guia do usuário do Amazon EKS.

Monitorar solicitações de autenticação do IAM

Usar a auditoria do Amazon DocumentDB

Acesse a pasta de registros de auditoria na Amazon CloudWatch e use padrões de pesquisa diferentes para obter os registros para autenticação do IAM. Por exemplo, use `{ $.param.mechanism = "MONGODB-AWS" }` como padrão de pesquisa “Pesquisar todos os fluxos de log”.

Para obter mais informações sobre eventos compatíveis com a auditoria, consulte Auditar eventos do Amazon DocumentDB.

Usando CloudWatch métricas da Amazon

`StsGetCallerIdentityCalls`: essa métrica mostra quantas `GetCallerIdentity` chamadas uma instância do Amazon DocumentDB está fazendo para o endpoint regionalized AWS Security Token Service ()AWS STS. Consulte a especificação de `MONGODB-AWS` autenticação sobre por que as instâncias de banco de dados precisam fazer `GetCallerIdentity` chamadas STS.

Usar a autenticação do IAM

Quando você não quiser gerenciar o nome de usuário e a senha em seu próprio banco de dados, você pode usar a autenticação do IAM. A autenticação do IAM está disponível somente na versão 5.0 do cluster baseado em instâncias do Amazon DocumentDB.

A autenticação do IAM depende de AWS Security Token Service (AWS STS). Se você receber uma exceção de AWS STS limitação ao usar a autenticação IAM, reduza sua taxa de conexão.

Para cotas do IAM, consulte IAM e AWS STS cotas no Guia do usuário do IAM.

Drivers que oferecem suporte ao IAM

Os drivers que suportam o Amazon DocumentDB 5.0 e o mecanismo de autenticação do `MONGODB-AWS` devem funcionar com a implementação de autenticação do IAM no Amazon DocumentDB.

> **Important**
>
> Há uma limitação conhecida com Node.js drivers anteriores à versão 6.13.1, que atualmente não são suportados pela autenticação de identidade IAM para o Amazon DocumentDB. Node.js drivers e ferramentas que usam Node.js driver (por exemplo, mongosh) devem ser atualizados para usar a versão 6.13.1 ou Node.js superior do driver.

Perguntas frequentes sobre autenticação de identidade do IAM

Há alguma amostra que eu possa consultar?

Consulte estas páginas para exemplos de casos de uso e configurações:

- Como usuários humanos podem se autenticar no Amazon DocumentDB usando usuários e perfis do IAM?
- Password-less autenticação no Amazon DocumentDB usando funções do IAM

Estou recebendo um erro ao usar meu driver Python: “pymongo.errors. ConfigurationError: A AWS autenticação MONGODB- requer pymongo-auth-aws”. Como posso resolver isso?

Certifique-se de usar a seguinte declaração ao instalar o driver Python com autenticação IAM:

```
pip install 'pymongo[aws]'
```

Isso instalará as AWS dependências adicionais necessárias para que a autenticação do IAM funcione.

Minha conexão cairá quando minhas credenciais temporárias de perfil do IAM expirarem?

Não, as credenciais temporárias do IAM são usadas somente para estabelecer conexão e autenticação. Em seguida, todas as autorizações adicionais acontecem no cluster do Amazon DocumentDB. Mesmo com as credenciais do IAM rotate/expire, a conexão não cairá nem ficará obsoleta.

Autenticação usando SCRAM-SHA-256

O Amazon DocumentDB (com compatibilidade com MongoDB) oferece suporte ao Salted Challenge Response Authentication Mechanism (SCRAM) para autenticação baseada em senha. Com o SCRAM, o servidor verifica se um cliente sabe a senha de um usuário sem transmitir a senha ao servidor. O Amazon DocumentDB oferece suporte a dois mecanismos SCRAM: e. `SCRAM-SHA-1` `SCRAM-SHA-256` `SCRAM-SHA-256`usa a função de SHA-256 hash mais forte e funciona com o modo FIPS.

Você atribui um ou ambos os mecanismos SCRAM a um usuário ao criar ou atualizar esse usuário. Os usuários que foram criados com `SCRAM-SHA-1` continuam trabalhando sem nenhuma alteração. SCRAM-SHA-256 a autenticação está disponível nas versões secundárias do cluster baseado em instância do Amazon DocumentDB 5.0.1+ e 8.0.1+.

> **A autenticação SCRAM e IAM são mutuamente exclusivas**
>
> Um usuário pode ter um mecanismo SCRAM ou o `MONGODB-AWS` mecanismo, mas não ambos. Para autenticação de identidade do IAM, consulteAutenticação usando identidade do IAM.

Tópicos

- Criação de usuários com mecanismos SCRAM
- Atualizando usuários existentes
- Autenticando com SCRAM-SHA-256

Criação de usuários com mecanismos SCRAM

Ao criar um usuário, você pode especificar os mecanismos de SCRAM a serem habilitados com o `mechanisms` campo. Os exemplos a seguir criam usuários com `SCRAM-SHA-1` apenas, `SCRAM-SHA-256` somente e ambos os 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 padrão

Se você não especificar o `mechanisms` campo, o Amazon DocumentDB habilitará ambos `SCRAM-SHA-1` e `SCRAM-SHA-256` para o usuário. Os dois comandos a seguir são 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']
});
```

A ativação de ambos os mecanismos por padrão mantém a compatibilidade com drivers MongoDB mais antigos que suportam apenas. `SCRAM-SHA-1` Os drivers mais novos e o modo FIPS preferem. `SCRAM-SHA-256` Se você não quiser que nenhum usuário se autentique`SCRAM-SHA-1`, especifique somente `SCRAM-SHA-256` no `mechanisms` campo para todos os seus usuários.

Atualizando usuários existentes

Os usuários criados com SCRAM-SHA-1 continuam recebendo suporte sem alterações. Para mudar um usuário existente de SCRAM-SHA-1 paraSCRAM-SHA-256, atualize o usuário com uma nova senha e o SCRAM-SHA-256 mecanismo:

```
db.runCommand({
    updateUser: 'existing',
    pwd: passwordPrompt(),
    mechanisms: ['SCRAM-SHA-256']
});
```

Autenticando com SCRAM-SHA-256

Seu driver se autentica automaticamente usando os parâmetros da cadeia de caracteres de conexão antes de enviar o primeiro comando em cada conexão. Para selecionar SCRAM-SHA-256 um URI de conexão, defina o authMechanism parâmetro comoSCRAM-SHA-256:

```
mongodb://username:password@hostname:port/authenticationDatabase?authMechanism=SCRAM-
SHA-256
```

Você também pode enviar uma solicitação de autenticação explícita do mongosh:

```
db.auth({user: 'sha256user', pwd: passwordPrompt(), mechanism: 'SCRAM-SHA-256'})
Enter password:
1
```

Se você não especificar um mecanismo de autenticação nas opções de conexão, o driver selecionará automaticamente um mecanismo suportado pelo usuário do banco de dados de destino.

Gerenciar usuários do Amazon DocumentDB

No Amazon DocumentDB, os usuários se autenticam em um cluster em conjunto com uma senha. Cada cluster tem credenciais primárias de login que são estabelecidas durante a criação do cluster.

> **Note**
>
> Todos os usuários criados antes de 26 de março de 2020 receberam as funções dbAdminAnyDatabase, readWriteAnyDatabase e clusterAdmin. Recomenda-se que

você reavalie todos os usuários e modifique as funções conforme necessário para impor o privilégio mínimo para todos os usuários em seus clusters.
Para obter mais informações, consulte Acesso ao banco de dados usando o controle de Role-Based acesso.

Usuário primário e `administrador de serviços`

Um cluster do Amazon DocumentDB recém-criado tem dois usuários: o usuário primário e o usuário `serviceadmin`.

O usuário primário é um usuário único e privilegiado que pode executar tarefas administrativas e criar usuários adicionais com perfis. Ao se conectar a um cluster Amazon DocumentDB pela primeira vez, você deve se autenticar usando as credenciais primárias de login. O usuário primário recebe essas permissões administrativas para um cluster do Amazon DocumentDB quando esse cluster é criado e recebe a função de `root`.

O usuário `serviceadmin` é criado implicitamente quando o cluster é criado. Cada cluster do Amazon DocumentDB tem um `serviceadmin` usuário que fornece AWS a capacidade de gerenciar seu cluster. Não é possível fazer login, descartar, renomear, alterar a senha nem alterar permissões do `serviceadmin`. Qualquer tentativa de fazer isso resulta em um erro.

> **Note**
>
> Os usuários primário e `serviceadmin` de um cluster do Amazon DocumentDB não podem ser excluídos, e a função `root` do usuário primário não pode ser revogada.
> Se você esquecer sua senha de usuário principal, poderá redefini-la usando o Console de gerenciamento da AWS ou AWS CLI o.

Criar usuários adicionais

Depois de conectar-se como usuário primário (ou qualquer usuário que tenha a função `createUser`), você pode criar um novo usuário, como mostrado abaixo.

```
db.createUser(
    {
        user: "sample-user-1",
        pwd: "password123",
```

```
        roles:
            [{"db":"admin", "role":"dbAdminAnyDatabase" }]
    }
)
```

Para visualizar detalhes do usuário, você pode usar o comando `show users` da seguinte maneira. Você também pode remover usuários com o comando `dropUser`. Para obter mais informações, consulte Comandos comuns.

```
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"
```

```
     }
    ]
}
```

Neste exemplo, o novo usuário `sample-user-1` é atribuído ao banco de dados `admin`. Esse é sempre o caso de um novo usuário. O Amazon DocumentDB não tem o conceito de um `authenticationDatabase` e, portanto, toda autenticação é realizada no contexto do banco de dados `admin`.

Ao criar usuários, se você omitir o campo db ao especificar a função, o Amazon DocumentDB atribuirá implicitamente a função ao banco de dados no qual a conexão está sendo emitida. Por exemplo, se sua conexão for emitida no banco de dados `sample-database` e você executar o comando a seguir, o usuário `sample-user-2` será criado no banco de dados `admin` e terá permissões `readWrite` no banco de dados `sample-database`.

```
db.createUser(
    {
        user: "sample-user-2",
        pwd: "password123",
        roles:
            ["readWrite"]
    }
)
```

A criação de usuários com funções com escopo em todos os bancos de dados (por exemplo, `readInAnyDatabase`) exige que você esteja no contexto do banco de dados `admin` ao criar o usuário ou indicar o banco de dados explicitamente para a função ao criar o usuário.

Para alternar o contexto do banco de dados, você pode usar o seguinte comando.

```
use admin
```

Para saber mais sobre o Controle de acesso baseado em função e impor o mínimo privilégio entre os usuários em seu cluster, consulte Acesso ao banco de dados usando o controle de Role-Based acesso.

Alternância automática de senhas para o Amazon DocumentDB

Com AWS Secrets Manager, você pode substituir as credenciais codificadas em seu código (incluindo senhas) por uma chamada de API para o Secrets Manager para recuperar o segredo programaticamente. Isso ajuda a garantir que o segredo não será comprometido por alguém que esteja examinando seu código, pois o segredo simplesmente não está ali. Além disso, configure o Secrets Manager para alterar automaticamente o segredo para você de acordo com a programação que você especificar. Isso permite substituir segredos de longo prazo por outros de curto prazo, ajudando a reduzir de maneira significativa o risco de comprometimento.

Usando o Secrets Manager, você pode alternar automaticamente suas senhas do Amazon DocumentDB (ou seja, segredos) usando uma AWS Lambda função fornecida pelo Secrets Manager.

Para obter mais informações sobre AWS Secrets Manager a integração nativa com o Amazon DocumentDB, consulte o seguinte:

- Blog: como fazer a rotação das credenciais do Amazon DocumentDB e Amazon Redshift no Secrets Manager AWS
- O que é o AWS Secrets Manager?
- Alternar segredos do AWS Secrets Manager
- Credenciais do Amazon DocumentDB no Secrets Manager

Acesso ao banco de dados usando o controle de Role-Based acesso

É possível restringir o acesso às ações que os usuários podem executar nos bancos de dados usando o role-based access control (RBAC) no Amazon DocumentDB (compatível com MongoDB). O RBAC funciona concedendo uma ou mais funções a um usuário. Estas funções determinam as operações que um usuário pode realizar nos recursos do banco de dados. Atualmente, o Amazon DocumentDB oferece suporte a funções integradas com o escopo no nível do banco de dados, como `read`, `readWrite`, `readAnyDatabase`, `clusterAdmin`, e funções definidas pelo usuário que podem ter como escopo ações específicas e recursos granulares, como coleções com base em seus requisitos.

Casos de uso comuns para RBAC incluem a imposição de privilégios mínimos criando usuários com acesso somente para leitura aos bancos de dados ou coleções em um cluster e designs de

aplicações de vários locatários que permitem que um único usuário acesse um determinado banco de dados ou coleção em um cluster.

> **Note**
>
> Todos os usuários criados antes de 26 de março de 2020 receberam as funções `dbAdminAnyDatabase`, `readWriteAnyDatabase` e `clusterAdmin`. Recomenda-se que você reavalie todos os usuários existentes e modifique as funções conforme necessário para impor privilégios mínimos para seus clusters.

Tópicos

- Conceitos do RBAC
- Introdução às funções integradas do RBAC
- Introdução às funções definidas pelo usuário do RBAC
- Conectar-se ao Amazon DocumentDB como um usuário
- Comandos comuns
- Diferenças funcionais
- Limites
- Acesso ao banco de dados usando o controle de Role-Based acesso

Conceitos do RBAC

Veja a seguir os termos e conceitos importantes relacionados ao controle de acesso baseado em função. Para obter mais informações sobre usuários do Amazon DocumentDB, consulte Gerenciar usuários do Amazon DocumentDB.

- Usuário — Uma entidade individual que pode se autenticar no banco de dados e realizar operações.
- Senha — Um segredo usado para autenticar o usuário.
- Função — Autoriza um usuário a realizar ações em um ou mais bancos de dados.
- Banco de dados administrativo — O banco de dados no qual os usuários são armazenados e autorizados.
- Banco de dados (**db**): o namespace dentro dos clusters que contém coleções para armazenar documentos.

O comando a seguir cria um cofre chamado `sample-user`.

```
db.createUser({user: "sample-user", pwd: "abc123", roles: [{role: "read", db: "sample-database"}]})
```

Neste exemplo:

- `user: "sample-user"` — Indica o nome do usuário.
- `pwd: "abc123"` — Indica a senha do usuário.
- `role: "read", "db: "sample-database"` — Indica que o usuário `sample-user` terá permissões de leitura em`sample-database`.



O exemplo a seguir mostra a saída depois que você obtiver o usuário `sample-user` com `db.getUser(sample-user)`. Neste exemplo, o usuário `sample-user` reside no banco de dados `admin`, mas tem a função de leitura no banco de dados `sample-database`.



Ao criar usuários, se você omitir o campo db ao especificar a função, o Amazon DocumentDB atribuirá implicitamente a função ao banco de dados no qual a conexão está sendo emitida. Por exemplo, se sua conexão for emitida no banco de dados `sample-database` e você executar o comando a seguir, o usuário `sample-user` será criado no banco de dados `admin` e terá permissões `readWrite` no banco de dados `sample-database`.

```
db.createUser({user: "sample-user", pwd: "abc123", roles: ["readWrite"]})
```

A saída dessa operação é semelhante à seguinte.

```
{
   "user":"sample-user",
   "roles":[
      {
         "db":"sample-database",
         "role":"readWrite"
      }
   ]
}
```

A criação de usuários com funções com escopo em todos os bancos de dados (por exemplo, `readAnyDatabase`) exige que você esteja no contexto do banco de dados `admin` ao criar o usuário ou que indique explicitamente o banco de dados para a função ao criar o usuário. Para emitir comandos no banco de dados `admin`, use o comando `use admin`. Para obter mais informações, consulte Comandos comuns.

Introdução às funções integradas do RBAC

Para ajudar você a começar a usar o controle de acesso baseado em função, esta seção demonstra um cenário de exemplo de aplicação de privilégios mínimos criando funções para três usuários com funções de trabalho diferentes.

- O `user1` é um novo gerente que precisa poder visualizar e acessar todos os bancos de dados em um cluster.
- O `user2` é um novo funcionário que precisa acessar apenas um banco de dados, `sample-database-1`, nesse mesmo cluster.
- O `user3` é um funcionário existente que precisa visualizar e acessar um banco de dados diferente, `sample-database-2`, ao qual ele não tinha acesso anteriormente, no mesmo cluster.

Em um ponto mais tarde, tanto o `user1` como o `user2` saem da empresa e, portanto, seu acesso deve ser revogado.

Para criar usuários e conceder funções, o usuário com o qual você se autentica no cluster deve ter uma função associada que possa executar ações para `createUser` e `grantRole`. Por exemplo, as funções `admin` e `userAdminAnyDatabase` podem conceder essas capacidades, por exemplo. Para obter ações por função, consulte Acesso ao banco de dados usando o controle de Role-Based acesso.

> **Note**
>
> No Amazon DocumentDB, todas as operações de usuário e função (por exemplo `create`, `get`, `drop`, `grant`, `revoke`, etc.) são implicitamente executadas no banco de dados `admin`independentemente de você estar emitindo comandos ao banco de dados `admin`.

Primeiro, para entender quais são os usuários e funções atuais no cluster, você pode executar o comando `show users`, como no exemplo a seguir. Você verá dois usuários, o `serviceadmin` e o usuário principal do cluster. Esses dois usuários sempre existem e não podem ser excluídos. Para obter mais informações, consulte Gerenciar usuários do Amazon DocumentDB.

```
show users
```

Para o `user1`, crie uma função com acesso de leitura e gravação a todos os bancos de dados em todo o cluster com o seguinte comando.

```
db.createUser({user: "user1", pwd: "abc123", roles: [{role: "readWriteAnyDatabase", db:
 "admin"}]})
```

A saída dessa operação é semelhante à seguinte.

```
{
   "user":"user1",
   "roles":[
      {
         "role":"readWriteAnyDatabase",
         "db":"admin"
      }
   ]
}
```

Para o `user2`, crie uma função com acesso somente leitura ao banco de dados `sample-database-1` com o seguinte comando.

```
db.createUser({user: "user2", pwd: "abc123", roles: [{role: "read", db: "sample-
database-1"}]})
```

A saída dessa operação é semelhante à seguinte.

```
{
   "user":"user2",
   "roles":[
      {
         "role":"read",
         "db":"sample-database-1"
      }
   ]
}
```

Para simular o cenário em que o `user3` é um usuário existente, primeiro crie o usuário `user3` e atribua uma nova função ao `user3`.

```
db.createUser({user: "user3", pwd: "abc123", roles: [{role: "readWrite", db: "sample-
database-1"}]})
```

A saída dessa operação é semelhante à seguinte.

```
{
   "user":"user3",
   "roles":[
      {
         "role":"readWrite",
         "db":"sample-database-1"
      }
   ]
}
```

Agora que o usuário `user3` foi criado, atribua ao `user3` a função `read` no `sample-database-2`.

```
db.grantRolesToUser("user3", [{role: "read", db: "sample-database-2"}])
```

Por fim, ambos `user1` e `user2` saem da empresa e precisam que seu acesso ao cluster seja revogado. Você pode fazer isso descartando os usuários, da seguinte forma.

```
db.dropUser("user1")
db.dropUser("user2")
```

Para garantir que todos os usuários tenham as funções apropriadas, você pode listar todos os usuários com o comando a seguir.

```
show users
```

A saída dessa operação é semelhante à seguinte.

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

Introdução às funções definidas pelo usuário do RBAC

Para ajudar você a começar a usar o controle de acesso baseado em função, esta seção demonstra um cenário de exemplo de aplicação de privilégios mínimos criando funções para três usuários com funções de trabalho diferentes.

Neste exemplo, aplica-se o seguinte:

- O `user1` é um novo gerente que precisa poder visualizar e acessar todos os bancos de dados em um cluster.
- O `user2` é um funcionário novo que precisa somente a ação ‘buscar’ em apenas um banco de dados, `sample-database-1`, nesse mesmo cluster.
- O `user3` é um funcionário existente que precisa visualizar e acessar uma coleção específica, col2 em um banco de dados diferente, `sample-database-2` ao qual ele não tinha acesso anteriormente, no mesmo cluster.
- Para o `user1`, crie uma função com acesso de leitura e gravação a todos os bancos de dados em todo o cluster com o seguinte comando.

```
db.createUser(
 {
     user: "user1", pwd: "abc123",
     roles: [{role: "readWriteAnyDatabase", db: "admin"}]
 }
 )
```

A saída dessa operação é semelhante à seguinte.

```
{
   "user":"user1",
   "roles":[
      {
         "role":"readWriteAnyDatabase",
         "db":"admin"
      }
   ]
}
```

Para`user2`, crie uma função com privilégios de 'busca' para todas as coleções no banco de dados `sample-database-1` com o comando a seguir. Observe que essa função garantiria que qualquer usuário associado só pudesse executar consultas de busca.

```
db.createRole(
{
    role: "findRole",
    privileges: [
    {
        resource: {db: "sample-database-1", collection: ""}, actions: ["find"]
    }],
    roles: []
}
)
```

A saída dessa operação é semelhante à seguinte.

```
{
   "role":"findRole",
   "privileges":[
      {
         "resource":{
            "db":"sample-database-1",
            "collection":""
         },
         "actions":[
            "find"
         ]
      }
   ],
   "roles":[

   ]
}
```

Em seguida, crie o usuário (`user2`) e associe a função criada recentemente `findRole` ao usuário.

```
db.createUser(
{
    user: "user2",
    pwd: "abc123",
    roles: []
```

```
})

 db.grantRolesToUser("user2",["findRole"])
```

Para simular o cenário em que o `user3` é um usuário existente, primeiro crie o usuário `user3` e, em seguida, atribua uma nova função chamada collectionRole que atribuiremos a `user3` na próxima etapa.

Agora você pode atribuir uma nova função a `user3`. Esta nova função permitirá `user3` a inserir, atualizar, excluir e encontrar acesso a uma coleção específica col2 em `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: []
}
)
```

A saída dessa operação é semelhante à seguinte.

```
{
   "role":"collectionRole",
   "privileges":[
      {
         "resource":{
            "db":"sample-database-2",
            "collection":"col2"
         },
         "actions":[
            "find",
```

```
            "update",
            "insert",
            "remove"
         ]
      }
   ],
   "roles":[

   ]
}
```

Agora que o usuário `user3` foi criado, você pode conceder ao `user3` a função `collectionFind`.

```
db.grantRolesToUser("user3",["collectionRole"])
```

Por fim, ambos `user1` e `user2` saem da empresa e precisam que seu acesso ao cluster seja revogado. Você pode fazer isso descartando os usuários, da seguinte forma.

```
db.dropUser("user1")
db.dropUser("user2")
```

Para garantir que todos os usuários tenham as funções apropriadas, você pode listar todos os usuários com o comando a seguir.

```
show users
```

A saída dessa operação é semelhante à seguinte.

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

Conectar-se ao Amazon DocumentDB como um usuário

Ao conectar-se a um cluster do Amazon DocumentDB, você se conecta no contexto de um banco de dados específico. Por padrão, se não especificar um banco de dados em sua string de conexão, você será automaticamente conectado ao cluster no contexto do banco de dados `test`. Todos os comandos de nível de coleção, como `insert` e `find`, são emitidos em coleções no banco de dados `test`.

Para ver o banco de dados em cujo contexto você está ou, em outras palavras, para o qual você está emitindo comandos, use o comando db no shell do mongo, da seguinte forma.

Consulta:

```
db
```

Saída:

```
test
```

Embora a conexão padrão possa estar no contexto do banco de dados `test`, isso não significa necessariamente que o usuário associado à conexão está autorizado a executar ações no banco de dados `test`. No cenário de exemplo anterior, se você se autenticar como o usuário `user3`, que tem a função `readWrite` no banco de dados `sample-database-1`, o contexto padrão da conexão será o banco de dados `test`. No entanto, se você tentar inserir um documento em uma coleção no banco de dados `test`, receberá uma mensagem de erro de falha de autorização. Isso ocorre porque esse usuário não está autorizado a executar esse comando nesse banco de dados, como mostrado abaixo.

Consulta:

```
db
```

Saída:

```
test
```

Consulta:

```
db.col.insert({x:1})
```

Saída:

```
WriteCommandError({ "ok" : 0, "code" : 13, "errmsg" : "Authorization failure" })
```

Se você alterar o contexto da sua conexão com o banco de dados `sample-database-1`, poderá gravar na coleção na qual o usuário tem autorização para fazê-lo.

Consulta:

```
use sample-database-1
```

Saída:

```
switched to db sample-database-1
```

Consulta:

```
db.col.insert({x:1})
```

Saída:

```
WriteResult({ "nInserted" : 1})
```

Ao autenticar-se em um cluster com um usuário específico, você também pode especificar o banco de dados na string de conexão. Isso remove a necessidade de executar o comando `use` depois que o usuário foi autenticado no banco de dados `admin`.

A seguinte string de conexão autentica o usuário no banco de dados `admin`, mas o contexto da conexão será com o banco de dados `sample-database-1`.

```
mongo "mongodb://user3:abc123@sample-cluster.node.us-east-1.docdb.amazonaws.com:27017/
sample-database-2"
```

Comandos comuns

Esta seção fornece exemplos de comandos comuns usando o controle de acesso baseado em função no Amazon DocumentDB. Você deve estar no contexto do banco de dados `admin` para criar e modificar usuários e funções. Você pode usar o comando `use admin` para alternar para o banco de dados `admin`.

> **Note**
>
> As modificações em usuários e funções ocorrerão implicitamente no banco de dados `admin`. A criação de usuários com funções que têm escopo em todos os bancos de dados (por exemplo, `readAnyDatabase`) requer que você esteja no contexto do banco de dados `admin` (ou seja, `use admin`) ao criar o usuário, ou que indique explicitamente o banco de dados para a função ao criar o usuário (como mostrado no Exemplo 2 nesta seção).

Exemplo 1: criar um usuário com a função `read` para o banco de dados `foo`.

```
db.createUser({user: "readInFooBar", pwd: "abc123", roles: [{role: "read", db:
 "foo"}]})
```

A saída dessa operação é semelhante à seguinte.

```
{
   "user":"readInFooBar",
   "roles":[
      {
         "role":"read",
         "db":"foo"
      }
   ]
}
```

Exemplo 2: criar um usuário com acesso para leitura em todos os bancos de dados.

```
db.createUser({user: "readAllDBs", pwd: "abc123", roles: [{role: "readAnyDatabase", db:
 "admin"}]})
```

A saída dessa operação é semelhante à seguinte.

```
{
   "user":"readAllDBs",
   "roles":[
      {
         "role":"readAnyDatabase",
         "db":"admin"
      }
   ]
}
```

Exemplo 3: conceder a função `read` a um usuário existente em um banco de dados novo.

```
db.grantRolesToUser("readInFooBar", [{role: "read", db: "bar"}])
```

Exemplo 4: atualizar a função de um usuário.

```
db.updateUser("readInFooBar", {roles: [{role: "read", db: "foo"}, {role: "read", db:
 "baz"}]})
```

Exemplo 5: revogar o acesso a um banco de dados de um usuário.

```
db.revokeRolesFromUser("readInFooBar", [{role: "read", db: "baz"}])
```

Exemplo 6: descrever uma função interna.

```
db.getRole("read", {showPrivileges:true})
```

A saída dessa operação é semelhante à seguinte.

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

Exemplo 7: descartar um usuário do cluster.

```
db.dropUser("readInFooBar")
```

A saída dessa operação é semelhante à seguinte.

```
true
```

Exemplo 8: criar uma função com acesso para leitura e gravação a uma coleção específica

```
db.createRole(
{
    role: "collectionRole",
    privileges: [
    {
        resource: {db: "sample-database-2", collection: "col2"}, actions: ["find",
 "update", "insert", "remove"]
    }],
    roles: []
}
)
```

A saída dessa operação é semelhante à seguinte.

```
{
   "role":"collectionRole",
   "privileges":[
      {
         "resource":{
            "db":"sample-database-2",
```

```
            "collection":"col2"
         },
         "actions":[
            "find",
            "update",
            "insert",
            "remove"
         ]
      }
   ],
   "roles":[

   ]
}
```

Exemplo 9: criar um usuário e atribuir uma função definida pelo usuário

```
db.createUser(
{
    user: "user3",
    pwd: "abc123",
    roles: []
})

db.grantRolesToUser("user3",["collectionRole"])
```

Exemplo 10: conceder privilégios adicionais a uma função definida pelo usuário

```
db.grantPrivilegesToRole(
  "collectionRole",
  [
    {
      resource: { db: "sample-database-1", collection: "col1" },
      actions: ["find", "update", "insert", "remove"]
    }
  ]
)
```

Exemplo 11: remover privilégios de uma função definida pelo usuário

```
db.revokePrivilegesFromRole(
  "collectionRole",
```

```
  [
    {
      resource: { db: "sample-database-1", collection: "col2" },
      actions: ["find", "update", "insert", "remove"]
    }
  ]
)
```

Exemplo 12: atualizar uma função existente definida pelo usuário

```
db.updateRole(
  "collectionRole",
  {
    privileges: [
    {
        resource: {db: "sample-database-3", collection: "sample-collection-3"},
 actions: ["find", "update", "insert", "remove"]
    }],
    roles: []
  }
)
```

Diferenças funcionais

No Amazon DocumentDB, as definições de usuário e função são armazenadas no banco de dados `admin`, e os usuários são autenticados no banco de dados `admin`. Essa funcionalidade difere do MongoDB Community Edition, mas é consistente com o MongoDB Atlas.

O Amazon DocumentDB também oferece suporte a fluxos de alterações, que fornecem uma sequência ordenada pelo tempo de eventos de alterações que ocorrem nas coleções do cluster. A ação `listChangeStreams` é aplicada no nível do cluster (ou seja, em todos os bancos de dados) e a ação `modifyChangeStreams` pode ser aplicada no nível do banco de dados e no nível do cluster.

Limites

A tabela a seguir contém os limites do controle de Role-Based acesso no Amazon DocumentDB.

Description	Limite
Número de usuários por cluster	1000

Description	Limite
Número de funções associadas a um usuário	1000
Número de funções definidas pelo usuário	100
Número de recursos associados a um privilégio	100

Acesso ao banco de dados usando o controle de Role-Based acesso

Com o controle de acesso baseado em função, você pode criar um usuário e conceder a ele uma ou mais funções para determinar quais operações esse usuário pode executar em um banco de dados ou cluster.

Veja a seguir uma lista de funções integradas que são compatíveis atualmente com o Amazon DocumentDB.

> **Note**
>
> No Amazon DocumentDB 4.0 e 5.0, os comandos `ListCollection` e `ListDatabase`podem, opcionalmente, usar os parâmetros `authorizedCollections` e `authorizedDatabases` para listar as coleções e bancos de dados que o usuário tem permissão de acessar exigindo as funções `listCollections` e `listDatabase`, respectivamente. Além disso, os usuários agora podem matar seus próprios cursores sem precisar da função `KillCursor`.

Database user

Nome do perfil	Description	Ações

`read`	Concede acesso de leitura a um usuário ao banco de dados especificado.	`changeStreams`
		`collStats`
		`dbStats`
		`find`

Nome do perfil	Description	Ações

		`killCursors`
		`listIndexes`
		`listCollections`

`readWrite`	Concede ao usuário acesso de leitura e gravação ao banco de dados especificado.	Todas as ações de permissões `read`.
		`createCollection`
		`dropCollection`
		`createIndex`
		`dropIndex`
		`insert`
		`killCursors`
		`listIndexes`
		`listCollections`
		`remove`
		`update`

Cluster user

Nome do perfil	Description	Ações
ler AnyDatabase	Concede acesso de leitura a todos os bancos de dados do cluster a um usuário.	Todas as ações de permissões `read`.

Nome do perfil	Description	Ações
		`listChangeStreams` `listDatabases`

`readWriteAnyDatabase`	Concede a um usuário acesso de leitura e gravação a todos os bancos de dados no cluster.	Todas as ações de permissões `readWrite` .
		`listChangeStreams`
		`listDatabases`

`userAdminAnyDatabase`	Concede a um usuário a capacidade de atribuir e modificar as funções ou privilégios que qualquer usuário tem ao banco de dados especificado.	`changeCustomData`
		`changePassword`
		`createUser`
		`dropRole`
		`dropUser`
		`grantRole`
		`listDatabases`
		`revokeRole`
		`viewRole`
		`viewUser`

Nome do perfil	Description	Ações

`dbAdminAnyDatabase`	Concede a um usuário a capacidade de executar funções de administração de banco de dados em qualquer banco de dados especificado.	Todas as ações de permissões `dbAdmin`.
		
		`dropCollection`
		
		`listDatabases`
		
		`listChangeStreams`
		
		`modifyChangeStreams`

Superuser

Nome do perfil	Description	Ações
`root`	Concede a um usuário acesso aos recursos e operações de todas as seguintes funções combinadas: `readWriteAnyDatabase` , `dbAdminAnyDatabase` , `userAdminAnyDatabase` , `clusterAdmin` , `restore` e `backup`.	Todas as ações de `readWriteAnyDatabase` , `dbAdminAnyDatabase` , `userAdminAnyDatabase` , `clusterAdmin` , `restore` e `backup`.

Database administrator

Nome do perfil	Description	Ações
`dbAdmin`	Concede a um usuário a capacidade de executar tarefas administrativas no banco de dados especificado.	`bypassDocumentValidation`

Nome do perfil	Description	Ações

		`collMod`
		`collStats`
		`createCollection`
		`createIndex`
		`dropCollection`
		`dropDatabase`
		`dropIndex`
		`dbStats`
		`find`
		`killCursors`
		`listIndexes`
		`listCollections`
		`modifyChangeStreams`

`dbOwner`	Concede a um usuário a capacidade de executar quaisquer tarefas administrativas no banco de dados especificado combinando as funções `dbAdmin` e `readWrite` .	Todas as ações de `dbAdmin` e `readWrite` .

Cluster administrator

Nome da função	Description	Ações

`clusterAdmin`	Concede a um usuário o maior acesso de gerenciamento de cluster combinando as funções `clusterManager` , `clusterMonitor` e `hostManager` .	Todas as ações de `clusterManager` , `clusterMonitor` e `hostManager` .
		`listChangeStreams`
		`dropDatabase`
		`modifyChangeStreams`

`clusterManager`	Concede a um usuário a capacidade de executar ações de gerenciamento e monitoramento no cluster especificado.	`listChangeStreams`
		`listSessions`
		`modifyChangeStreams`
		`replSetGetConfig`

`clusterMonitor`	Concede a um usuário a capacidade de ter acesso somente leitura a ferramentas de monitoramento.	`collStats`
		`dbStats`
		`find`
		`getParameter`
		`hostInfo`
		`indexStats`
		`killCursors`

Nome da função	Description	Ações

		`listChangeStreams`
		`listCollections`
		`listDatabases`
		`listIndexes`
		`listSessions`
		`replSetGetConfig`
		`serverStatus`
		`top`

`hostManager`	Concede a um usuário a capacidade de monitorar e gerenciar servidores.	`auditConfigure`
		`killCursors`
		`killAnyCursor`
		`killAnySession`
		`killop`

Backup administrator

Nome do perfil	Description	Ações

`backup`	Concede a um usuário o acesso necessário para fazer backup de dados.	`getParameter`
		`insert`
		`find`

Nome do perfil	Description	Ações

		`listChangeStreams`
		`listCollections`
		`listDatabases`
		`listIndexes`
		`update`

Nome do perfil	Description	Ações

`restore`	Concede a um usuário o acesso necessário para restaurar dados.	`bypassDocumentValidation`
		`changeCustomData`
		`changePassword`
		`collMod`
		`createCollection`
		`createIndex`
		`createUser`
		`dropCollection`
		`dropRole`
		`dropUser`
		`getParameter`
		`grantRole`
		`find`
		`insert`
		`listCollections`
		`modifyChangeStreams`
		`revokeRole`

Nome do perfil	Description	Ações

		`remove`
		`viewRole`
		`viewUser`
		`update`

Atualizar seus certificados TLS do Amazon DocumentDB

Tópicos

- Atualizar sua aplicação e seu cluster do Amazon DocumentDB
- Alternância automática de certificados do servidor
- Solução de problemas
- Perguntas frequentes

O certificado da autoridade de certificação (CA) para clusters Amazon DocumentDB foi atualizado em agosto de 2024. Se estiver usando clusters do Amazon DocumentDB com o Transport Layer Security (TLS) habilitado (a configuração padrão) e não tiver alternado a aplicação cliente e os certificados de servidor, as seguintes etapas serão necessárias para minimizar problemas de conectividade entre sua aplicação e seus clusters do Amazon DocumentDB.

- Etapa 1: Fazer download do novo certificado de CA e atualizar seu aplicativo
- Etapa 2: atualizar o certificado do servidor

Os certificados CA e de servidor foram atualizados como parte das melhores práticas de manutenção e segurança padrão do Amazon DocumentDB. As aplicações cliente devem adicionar os novos certificados de CA em seus armazenamentos de confiança, e as instâncias do Amazon DocumentDB existentes devem ser atualizadas para usar os novos certificados de CA antes dessa data de validade.

Atualizar sua aplicação e seu cluster do Amazon DocumentDB

Siga as etapas nesta seção para atualizar o pacote de certificados CA da aplicação (Etapa 1) e os certificados de servidor do cluster (Etapa 2). Antes de aplicar as alterações nos seus ambientes de produção, recomendamos testar estas etapas em um ambiente de desenvolvimento ou teste.

> **Note**
>
> Você deve concluir as etapas 1 e 2 Região da AWS em cada uma das quais você tem clusters do Amazon DocumentDB.

Etapa 1: Fazer download do novo certificado de CA e atualizar seu aplicativo

Faça download do novo certificado da CA e atualize sua aplicação para usá-la na criação de conexões TLS com o Amazon DocumentDB. Faça download do novo pacote de certificados CA em https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem. Essa operação faz download de um arquivo chamado `global-bundle.pem`.

> **Note**
>
> Se você estiver acessando o armazenamento de chaves que contém o certificado CA antigo (`rds-ca-2019-root.pem`) e os novos certificados CA (`rds-ca-rsa2048-g1`, `rds-ca-rsa4096-g1`, `rds-ca-ecc384-g1`), certifique-se de que o armazenamento de chaves seleciona `global-bundle`.

```
wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
```

Depois, atualize suas aplicações para usar o novo pacote de certificados. O novo pacote de CA contém o certificado CA antigo (rds-ca-2019) e os novos certificados CA (rds-ca-rsa2048-g1, rds-ca-rsa4096-g1, rds-ca-ecc384-g1). Com os dois certificados da autoridade de certificação (CA) no novo pacote, é possível atualizar sua aplicação e cluster em duas etapas.

Para aplicações Java, crie um novo armazenamento confiável com o novo certificado CA. Para obter instruções, consulte a guia Java no tópico Conectar-se com o TLS habilitado.

Para verificar se a aplicação está usando o pacote de certificados CA mais recente, consulte Como garantir que estou usando o pacote da CA mais recente?. Se você já estiver usando o pacote de certificados CA mais recente em sua aplicação, poderá avançar para a Etapa 2.

Para obter exemplos de como usar um pacote CA com a sua aplicação, consulte Criptografia de dados em trânsito e Conectar-se com o TLS habilitado.

> **Note**
>
> Atualmente, o MongoDB Go Driver 1.2.1 aceita somente um certificado de servidor CA em `sslcertificateauthorityfile`. Consulte Conectar-se com o TLS habilitado para se conectar ao Amazon DocumentDB usando Go quando o TLS está habilitado.

Etapa 2: atualizar o certificado do servidor

Depois que a aplicação foi atualizada para usar o novo pacote de CA, o próximo passo é atualizar o certificado de servidor modificando cada instância em um cluster do Amazon DocumentDB. Para modificar as instâncias para usarem o novo certificado de servidor, consulte as instruções a seguir.

O Amazon DocumentDB fornece as CAs a seguir para assinar o certificado do servidor de banco de dados para uma instância de banco de dados:

- rds-ca-ecc384-g1: utiliza uma autoridade de certificação com algoritmo de chave privada ECC 384 e algoritmo de assinatura SHA384. Essa CA é compatível com a alternância automática de certificados do servidor. Isso tem suporte somente no Amazon DocumentDB 4.0 e 5.0.
- rds-ca-rsa2048-g1 — usa uma autoridade de certificação com algoritmo de chave privada RSA 2048 e algoritmo de assinatura SHA256 na maioria. Regiões da AWS Essa CA é compatível com a alternância automática de certificados do servidor.
- rds-ca-rsa4096-g1-Utiliza uma autoridade de certificação com algoritmo de chave privada RSA 4096 e algoritmo de assinatura SHA384. Essa CA é compatível com a alternância automática de certificados do servidor.

> **Note**
>
> Se você estiver usando o AWS CLI, poderá ver as validades das autoridades de certificação listadas na seção anterior usando describe-certificates. https://docs.aws.amazon.com/cli/latest/reference/docdb/describe-certificates.html

Esses certificados CA estão incluídos no pacote de certificados regionais e globais. Quando você usa a CA rds-ca-rsa2048-g1, rds-ca-rsa4096-g1 ou rds-ca-ecc384-g1 com um banco de dados, o Amazon DocumentDB gerencia o certificado do servidor de banco de dados no banco de dados. O Amazon DocumentDB alterna automaticamente o certificado do servidor de banco de dados antes que ele expire.

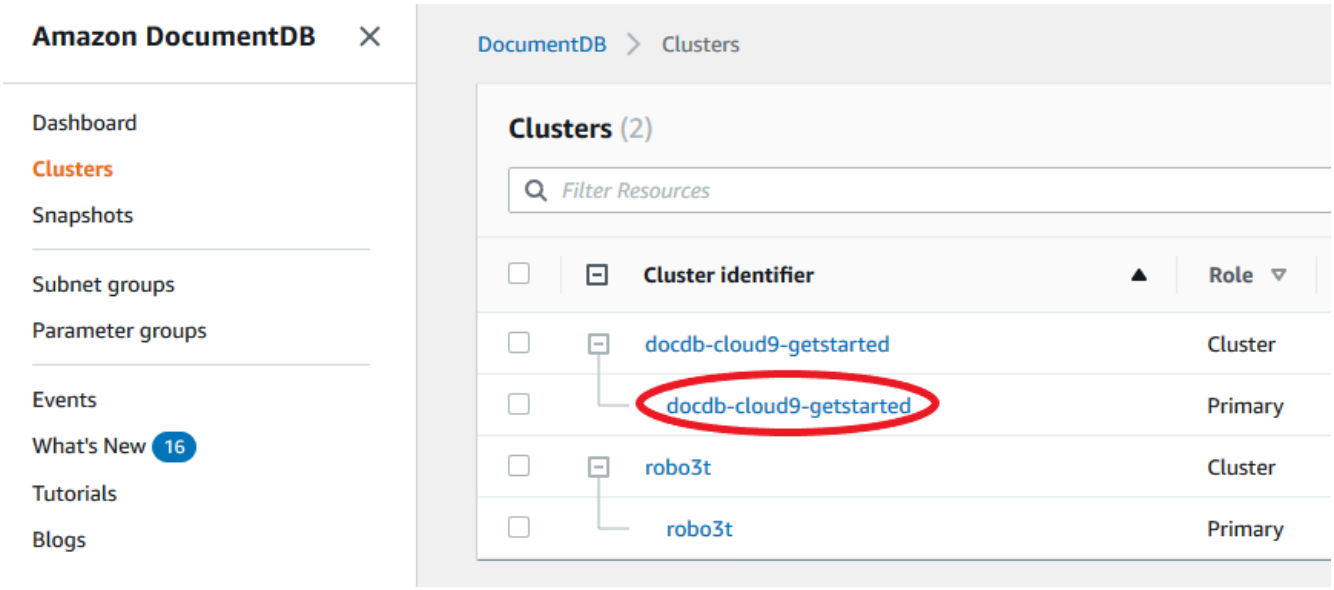
> **Note**
>
> O Amazon DocumentDB não exigirá reinicialização para alternar certificados se seu cluster estiver em execução nas seguintes versões de patch do mecanismo:
>
> - Amazon DocumentDB 3.6: 1.0.208662 ou superior
> - Amazon DocumentDB 4.0: 2.0.10179 ou superior
> - Amazon DocumentDB 5.0: 3.0.4780 ou superior
>
> É possível determinar a versão atual do patch do mecanismo do Amazon DocumentDB executando o seguinte comando: `db.runCommand({getEngineVersion: 1})`. Antes de atualizar o certificado de servidor, verifique se você concluiu a Etapa 1.

Using the Console de gerenciamento da AWS

Conclua as etapas a seguir para identificar e alternar o certificado de servidor antigo para suas instâncias existentes do Amazon DocumentDB usando a Console de gerenciamento da AWS.

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. Na lista de regiões no canto superior direito da tela, escolha Região da AWS em que seus clusters residem.
3. No painel de navegação no lado esquerdo do console, em DAX, selecione Clusters.

4. Talvez seja necessário identificar quais instâncias ainda estão no certificado antigo do servidor (`rds-ca-2019`). É possível fazer isso na coluna Autoridade de certificação, localizada na extremidade direita da tabela Clusters.
5. Na tabela Clusters, você verá a coluna Identificador do cluster na extremidade esquerda. Suas instâncias estão listadas em clusters, semelhante à captura de tela a seguir.



6. Marque a caixa à esquerda da instância na qual você está interessado.
7. Escolha Ação e Modificar.
8. Em Autoridade de certificação, selecione o novo certificado do servidor (`rds-ca-rsa2048-g1`) para esta instância.
9. Você verá um resumo das alterações na próxima página. Há um alerta adicional para lembrar você de garantir que a aplicação esteja usando o pacote de certificados da CA mais recente antes de modificar a instância, para evitar interrupções na conectividade.
10. É possível optar por aplicar a modificação durante a próxima janela de manutenção ou imediatamente. Se sua intenção é modificar o certificado do servidor imediatamente, use a opção Aplicar imediatamente.
11. Escolha Modificar instância para concluir a atualização.

Using the AWS CLI

Conclua as etapas a seguir para identificar e girar o certificado de servidor antigo para suas instâncias existentes do Amazon DocumentDB usando a AWS CLI.

1. Para modificar as instâncias imediatamente, execute o seguinte comando para cada instância do cluster.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier>
 --ca-certificate-identifier rds-ca-rsa2048-g1 --apply-immediately
```

2. Para modificar as instâncias nos clusters para usarem o novo certificado CA na próxima janela de manutenção do cluster, execute o seguinte comando para cada instância no cluster.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier>
 --ca-certificate-identifier rds-ca-rsa2048-g1 --no-apply-immediately
```

Alternância automática de certificados do servidor

O Amazon DocumentDB oferece suporte à alternância automática de certificados do servidor. O certificado do servidor é o certificado de entidade final emitido para cada instância do cluster. Ao contrário dos certificados CA raiz, os certificados do servidor têm validade curta (12 meses) e o Amazon DocumentDB gerencia automaticamente sua alternância sem nenhuma ação de sua parte. O Amazon DocumentDB usa a mesma CA raiz para essa alternância automática, então você não precisa baixar um novo pacote de CA.

> **Important**
>
> Ao se conectar ao seu cluster Amazon DocumentDB, confie no pacote de CA raiz em vez de confiar diretamente em cada certificado de servidor. Isso evitará erros de conexão após a alternância do certificado do servidor. Consulte Conectar-se com o TLS habilitado.

O Amazon DocumentDB tenta alternar o certificado do servidor em sua janela de manutenção preferida na meia-vida do respectivo certificado. O novo certificado do servidor é válido por 12 meses.

Use o comando `describe-db-engine-versions` e inspecione o sinalizador `SupportsCertificateRotationWithoutRestart` para identificar se a versão do mecanismo oferece suporte à alternância de certificado sem reinicialização.

> **Note**
>
> O Amazon DocumentDB oferece suporte à alternância de certificados do servidor sem reinicialização se o seu cluster estiver em execução nas versões de patch do mecanismo a seguir:
>
> - Amazon DocumentDB 3.6: 1.0.208662 ou superior
> - Amazon DocumentDB 4.0: 2.0.10179 ou superior
> - Amazon DocumentDB 5.0: 3.0.4780 ou superior
>
> É possível determinar a versão atual do patch do mecanismo do Amazon DocumentDB executando este comando: `db.runCommand({getEngineVersion: 1})`.
> Se você estiver usando uma versão de patch de mecanismo mais antiga, o Amazon DocumentDB alternará o certificado do servidor e agendará um evento de reinicialização do banco de dados na janela de manutenção de sua preferência.

Solução de problemas

Se você estiver tendo problemas para se conectar ao seu cluster como parte da rotação de certificados, tente o seguinte:

- Verifique se seus clientes estão usando o pacote de certificado mais recente. Consulte Como garantir que estou usando o pacote da CA mais recente?.
- Verifique se suas instâncias estão usando o certificado mais recente. Consulte Como sei quais das minhas instâncias do Amazon DocumentDB estão usando o certificado do old/new servidor?.
- Verifique se a CA de certificado mais recente está sendo utilizada por sua aplicação. Alguns drivers, como Java e Go, exigem código extra para importar vários certificados de um pacote de certificados para o armazenamento confiável. Para obter mais informações sobre como se conectar ao Amazon DocumentDB usando TLS, consulte Conectar-se de forma programática ao Amazon DocumentDB.
- Entre em contato com o suporte. Se você tiver dúvidas ou problemas, entre em contato com o Suporte.

Perguntas frequentes

Veja a seguir as respostas a algumas perguntas comuns sobre certificados TLS.

E se eu tiver dúvidas ou problemas?

Se você tiver dúvidas ou problemas, entre em contato com o Suporte.

Como sei se estou usando o TLS para conectar com meu cluster do Amazon DocumentDB?

Para ver se o cluster está usando TLS, examine o parâmetro `tls` do seu grupo de parâmetros do cluster. Se o parâmetro `tls` estiver definido como `enabled`, você está usando o certificado TLS para se conectar ao cluster. Para obter mais informações, consulte Gerenciando grupos de parâmetros de cluster do Amazon DocumentDB.

Por que vocês estão atualizando os certificados da CA e do servidor?

Os certificados da CA e do servidor do Amazon DocumentDB estão sendo atualizados como parte das práticas recomendadas de manutenção e segurança padrão do Amazon DocumentDB.

O que acontecerá se eu não fizer nada até a data de vencimento?

Se você estiver usando TLS para se conectar ao cluster do Amazon DocumentDB com um certificado CA expirado, suas aplicações que se conectam via TLS não poderão mais se comunicar com o cluster do Amazon DocumentDB.

O Amazon DocumentDB não alternará seus certificados de banco de dados automaticamente antes da expiração. É necessário atualizar suas aplicações e clusters para usar os novos certificados CA antes ou depois da data de expiração.

Como sei quais das minhas instâncias do Amazon DocumentDB estão usando o certificado do old/new servidor?

Para identificar as instâncias do Amazon DocumentDB que ainda usam o certificado de servidor antigo, você pode usar o Amazon DocumentDB Console de gerenciamento da AWS ou o. AWS CLI

Usar o Console de gerenciamento da AWS

Como identificar as instâncias em seus clusters que estão usando o certificado mais antigo

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. Na lista de regiões no canto superior direito da tela, escolha Região da AWS em que suas instâncias residem.
3. No painel de navegação no lado esquerdo do console, em DAX, selecione Clusters.
4. A coluna Autoridade de certificação (próxima à extremidade direita da tabela) mostra quais instâncias ainda estão com o certificado antigo do servidor (`rds-ca-2019`) e com o novo certificado do servidor (`rds-ca-rsa2048-g1`).

Usar o AWS CLI

Para identificar as instâncias em seus clusters que estão usando o certificado de servidor mais antigo, use o comando `describe-db-clusters` com o seguinte.

```
aws docdb describe-db-instances \
    --filters Name=engine,Values=docdb \
    --query 'DBInstances[*].
{CertificateVersion:CACertificateIdentifier,InstanceID:DBInstanceIdentifier}'
```

Como modifico instâncias individuais no meu cluster do Amazon DocumentDB para atualizar o certificado do servidor?

Atualize os certificados do servidor para todas as instâncias em um determinado cluster ao mesmo tempo. Para modificar as instâncias em seu cluster, é possível usar o console ou a AWS CLI.

> **Note**
>
> Antes de atualizar o certificado de servidor, verifique se você concluiu a Etapa 1.

Usar o Console de gerenciamento da AWS

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. Na lista de regiões no canto superior direito da tela, escolha Região da AWS em que seus clusters residem.
3. No painel de navegação no lado esquerdo do console, em DAX, selecione Clusters.
4. A coluna Autoridade de certificação (próxima à extremidade direita da tabela) mostra quais instâncias ainda estão com o certificado antigo do servidor (`rds-ca-2019`) e com o novo certificado do servidor ().
5. Na tabela Clusters, em Identificador de cluster, selecione uma instância para modificar.
6. Escolha Ação e Modificar.
7. Em Autoridade de certificação, selecione o novo certificado do servidor (`rds-ca-rsa2048-g1`) para esta instância.
8. Você verá um resumo das alterações na próxima página. Há um alerta adicional para lembrar você de garantir que a aplicação esteja usando o pacote de certificados da CA mais recente antes de modificar a instância, para evitar interrupções na conectividade.
9. É possível optar por aplicar a modificação durante a próxima janela de manutenção ou imediatamente.
10. Escolha Modificar instância para concluir a atualização.

Usar o AWS CLI

Conclua as etapas a seguir para identificar e alternar o certificado de servidor antigo para suas instâncias existentes do Amazon DocumentDB usando a AWS CLI.

1. Para modificar as instâncias imediatamente, execute o seguinte comando para cada instância do cluster.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier> --
ca-certificate-identifier rds-ca-rsa2048-g1 --apply-immediately
```

2. Para modificar as instâncias nos clusters para usarem o novo certificado CA na próxima janela de manutenção do cluster, execute o seguinte comando para cada instância no cluster.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier> --
ca-certificate-identifier rds-ca-rsa2048-g1 --no-apply-immediately
```

O que acontecerá se eu adicionar uma nova instância a um cluster existente?

Todas as novas instâncias criadas usam o certificado de servidor antigo e exigem conexões TLS com o certificado CA antigo. Qualquer instância nova do Amazon DocumentDB criada após 25 de janeiro de 2024 usará, como padrão, o novo certificado rds-ca-rsa2048-g1.

O que acontecerá se houver uma substituição de instância ou um failover no meu cluster?

Se houver uma substituição de instância no cluster, a nova instância criada continuará usando o mesmo certificado do servidor que a outra instância estava usando anteriormente. Recomendamos atualizar os certificados do servidor para todas as instâncias ao mesmo tempo. Se um failover acontecer no cluster, o certificado do servidor no novo primário será usado.

Se eu não uso TLS para me conectar ao meu cluster, ainda preciso atualizar cada uma das minhas instâncias?

É altamente recomendável ativar o TLS. Caso você não habilite o TLS, ainda recomendamos alternar os certificados em suas instâncias do Amazon DocumentDB, caso você planeje usar o TLS para se conectar aos seus clusters no futuro. Se você não pretende usar nunca o TLS para se conectar aos seus clusters do Amazon DocumentDB, não precisa fazer nada.

Se eu não estiver usando TLS para a conexão com meu cluster, mas planejo usá-lo no futuro, o que devo fazer?

Se você criou um cluster antes de janeiro de 2024, siga as Etapa 1 e Etapa 2 da seção anterior para garantir que sua aplicação esteja usando o pacote de CA atualizado e que cada instância do Amazon DocumentDB esteja usando o certificado de servidor mais recente. Se você criar um cluster após 25 de janeiro de 2024, seu cluster já terá o certificado de servidor mais recente (rds-ca-rsa2048-g1). Para verificar se a aplicação está usando o pacote de certificados CA mais recente, consulte Se eu não uso TLS para me conectar ao meu cluster, ainda preciso atualizar cada uma das minhas instâncias?.

O prazo pode ser prorrogado para além de agosto de 2024?

Se suas aplicações estiverem se conectando via TLS, o prazo não poderá ser prorrogado.

Como garantir que estou usando o pacote da CA mais recente?

Para verificar se você tem o pacote mais recente, use o comando a seguir. Para executar este comando, você deve ter o java instalado e as ferramentas java precisam estar na variável PATH do seu shell. Para obter mais informações, consulte Usar o Java

macOS e Amazon Linux

```
keytool -printcert -v -file global-bundle.pem
```

Windows

```
keytool -printcert -v -file global-bundle.p7b
```

Por que vejo "RDS" no nome do pacote da CA?

Para alguns recursos de gerenciamento, como o gerenciamento de certificados, o Amazon DocumentDB usa a tecnologia operacional que é compartilhada com o Amazon Relational Database Service (Amazon RDS).

Quando o novo certificado expirará?

O novo certificado de servidor expirará (geralmente) da seguinte forma:

- rds-ca-rsa2048-g1— Expira em 2061
- rds-ca-rsa4096-g1— Expira em 2121
- rds-ca-ecc384-g1: expira em 2121

Que tipo de erros eu terei se eu não agir antes que o certificado expire?

As mensagens de erro variam dependendo do driver. Em geral, você verá erros de validação do certificado que contêm a string “o certificado expirou”.

Se eu apliquei o novo certificado de servidor, posso revertê-lo para o antigo certificado de servidor?

Se você precisar reverter uma instância para o certificado antigo do servidor, faça isso para todas as instâncias no cluster. Você pode reverter o certificado do servidor para cada instância em um cluster usando o Console de gerenciamento da AWS ou o. AWS CLI

Usar o Console de gerenciamento da AWS

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. Na lista de regiões no canto superior direito da tela, escolha Região da AWS em que seus clusters residem.
3. No painel de navegação no lado esquerdo do console, em DAX, selecione Clusters.
4. Na tabela Clusters, em Identificador de cluster, selecione uma instância para modificar. Escolha Ações e, em seguida, Modificar.
5. Em Autoridade de certificação, é possível selecionar o certificado de servidor antigo (`rds-ca-2019`).
6. Escolha Continue para exibir um resumo das modificações.
7. Nesta página resultante, é possível optar por programar suas modificações para serem aplicadas na próxima janela de manutenção ou aplicá-las imediatamente. Faça sua seleção e escolha Modificar instância.

> **Note**
>
> Se você optar por aplicar as alterações imediatamente, todas as alterações na fila de modificações pendentes também serão aplicadas. Se qualquer uma das alterações pendentes exigir tempo de inatividade, escolher essa opção poderá causar um tempo de inatividade inesperado.

Usar o AWS CLI

```
aws docdb modify-db-instance --db-instance-identifier <db_instance_name> ca-
certificate-identifier rds-ca-2019 <--apply-immediately | --no-apply-immediately>
```

Se você escolher `--no-apply-immediately`, a alteração será aplicada na próxima janela de manutenção do cluster.

Se eu restaurar de um snapshot ou uma restauração point-in-time, ele terá o novo certificado de servidor?

Se você restaurar um snapshot ou executar uma restauração point-in-time após agosto de 2024, o novo cluster criado usará o novo certificado CA.

E se eu estiver tendo problemas para conectar-me diretamente ao cluster do Amazon DocumentDB de qualquer Mac OS?

O Mac OS atualizou os requisitos para certificados confiáveis. Os certificados confiáveis agora devem ser válidos por 397 dias ou menos (consulte https://support.apple.com/en-us/HT211025).

> **Note**
>
> Esta restrição é observada nas versões mais recentes do Mac OS.

Os certificados de instância do Amazon DocumentDB são válidos por mais de quatro anos, mais do que o máximo do Mac OS. Para conectar-se diretamente a um cluster do Amazon DocumentDB em um computador que executa o Mac OS, você deve dar permissão a certificados inválidos ao criar a conexão TLS. Nesse caso, os certificados inválidos significam que o período de validade é superior a 397 dias. É necessário entender os riscos antes de permitir certificados inválidos ao conectar-se ao seu cluster do Amazon DocumentDB.

Para se conectar a um cluster Amazon DocumentDB a partir do Mac OS usando o AWS CLI, use o `tlsAllowInvalidCertificates` parâmetro.

```
mongo --tls --host <hostname> --username <username> --password <password> --port 27017
 --tlsAllowInvalidCertificates
```

Validação de conformidade no Amazon DocumentDB

Auditores terceirizados avaliam a segurança e a conformidade do Amazon DocumentDB como parte de vários programas de conformidade da AWS , incluindo o seguinte:

- Controles de Sistema e Organização (SOC) 1, 2 e 3. Para obter mais informações, consulte SOC.
- Federal Risk and Authorization Management Program (FedRAMP). Para obter mais informações, consulte Serviços da AWS no escopo por programa de conformidade.
- Padrão de Segurança de Dados do Setor de Cartões de Pagamento (PCI DSS). Para obter mais informações, consulte PCI DSS.
- ISO 9001, 27001, 27017 e 27018. Para obter mais informações, consulte Certificado ISO.
- Contrato de Parceria Comercial da Lei de Responsabilidade e Portabilidade de Seguro Saúde (HIPAA BAA). Para obter mais informações, consulte Conformidade com a HIPAA

AWS fornece uma lista frequentemente atualizada de AWS serviços no escopo de programas de conformidade específicos no AWS Services in Scope by Compliance Program.

Third-party relatórios de auditoria estão disponíveis para você baixar usando AWS Artifact. Para obter mais informações, consulte Download de relatórios no AWS Artifact.

Para obter mais informações sobre programas de AWS conformidade, consulte Programas de AWS conformidade.

Sua responsabilidade de conformidade ao usar o Amazon DocumentDB é determinada pela confidencialidade de seus dados, pelas metas de conformidade da sua organização, pelas regulamentações e leis aplicáveis. Se seu uso do Amazon DocumentDB estiver sujeito à conformidade com padrões como HIPAA ou PCI, AWS forneça recursos para ajudar a:

- AWS Recursos de conformidade — Uma coleção de pastas de trabalho e guias que podem se aplicar ao seu setor e local.
- Guias de início rápido de segurança e conformidade — Guias de implantação que discutem considerações arquitetônicas e fornecem etapas para a implantação de ambientes básicos com foco em segurança e conformidade em. AWS
- AWS Config — Um serviço que avalia o quão bem suas configurações de recursos estão em conformidade com as práticas internas, as diretrizes do setor e as regulamentações.
- AWS Security Hub — Uma visão abrangente do seu estado de segurança AWS que ajuda você a verificar sua conformidade com os padrões e as melhores práticas do setor de segurança.
- Whitepaper Arquitetando para Segurança e Conformidade com a HIPAA — Um whitepaper que descreve como as empresas podem usar para criar aplicativos. AWS HIPAA-compliant

Resiliência no Amazon DocumentDB

A infraestrutura AWS global é construída em torno Regiões da AWS de zonas de disponibilidade. Regiões da AWS fornecem várias zonas de disponibilidade fisicamente separadas e isoladas, conectadas com redes de baixa latência, alto rendimento e altamente redundantes. Com as zonas de disponibilidade, é possível projetar e operar aplicações e bancos de dados que executam o failover automaticamente entre as zonas de disponibilidade sem interrupção. As Zonas de Disponibilidade são mais altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de datacenter tradicionais.

Um cluster do Amazon DocumentDB só pode ser criado em uma Amazon VPC que possua pelo menos duas sub-redes em pelo menos duas Zonas de Disponibilidade. Ao distribuir suas instâncias de cluster por pelo menos duas Zonas de Disponibilidade, o Amazon DocumentDB ajuda a garantir instâncias disponíveis no cluster no caso improvável de falha na Zona de Disponibilidade. O volume de seu cluster Amazon DocumentDB sempre abrange três Zonas de Disponibilidade para fornecer um armazenamento durável com menor possibilidade de perda de dados.

Para obter mais informações sobre zonas de disponibilidade Regiões da AWS e zonas de disponibilidade, consulte Infraestrutura AWS global.

Além da infraestrutura AWS global, o Amazon DocumentDB oferece vários recursos para ajudar a suportar suas necessidades de resiliência de dados e backup.

Fault-tolerant e armazenamento com autorrecuperação

> Cada porção de 10 GB do seu volume de armazenamento é replicada de seis maneiras, por três Zonas de Disponibilidade. O Amazon DocumentDB usa armazenamento tolerante a falhas que processa de forma transparente a perda de até duas cópias de dados sem afetar a disponibilidade de gravação do banco de dados, e até três cópias sem afetar a disponibilidade de leitura. O armazenamento do Amazon DocumentDB também é autocorretivo: os blocos de dados e discos são continuamente escaneados em busca de erros e substituídos automaticamente.

Backups e restauração manuais

> O Amazon DocumentDB fornece a capacidade de criar backups completos do cluster para retenção e recuperação a longo prazo. Para obter mais informações, consulte Backup e restauração no Amazon DocumentDB.

Point-in-time recuperação

Point-in-time a recuperação ajuda a proteger seus clusters do Amazon DocumentDB contra operações acidentais de gravação ou exclusão. Com a recuperação pontual, você não precisa se preocupar com a criação, a manutenção ou a programação de backups sob demanda. Para obter mais informações, consulte Restauração para um ponto no tempo.

Segurança da infraestrutura no Amazon DocumentDB

Como um serviço gerenciado, o Amazon DocumentDB é protegido pela segurança de rede AWS global. Para obter informações sobre serviços AWS de segurança e como AWS protege a infraestrutura, consulte Segurança AWS na nuvem. Para projetar seu AWS ambiente usando as melhores práticas de segurança de infraestrutura, consulte Proteção de infraestrutura no pilar de segurança AWS Well-Architected Framework.

Você usa chamadas de API AWS publicadas para acessar o Amazon DocumentDB pela rede. Os clientes devem oferecer compatibilidade com:

- Transport Layer Security (TLS). Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Suítes de criptografia com sigilo direto perfeito (PFS), como DHE (Ephemeral) ou ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Diffie-Hellman A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

É possível chamar essas operações de API de qualquer local de rede. Também é possível usar políticas do Amazon DocumentDB para controlar o acesso a partir de endpoints da Amazon Virtual Private Cloud (Amazon VPC) ou de VPCs específicos. Efetivamente, isso isola o acesso à rede a um determinado recurso do Amazon DocumentDB somente da VPC específica dentro da rede. AWS

> **Note**
>
> O Amazon DocumentDB não oferece suporte a políticas de acesso.

API do Amazon DocumentDB e endpoints de VPC de interface (AWS PrivateLink)

> **Note**
>
> Os clusters elásticos do Amazon DocumentDB não oferecem suporte a endpoints AWS PrivateLink VPC.

É possível estabelecer uma conexão privada entre a VPC e os endpoints da API do Amazon DocumentDB criando um endpoint da VPC de interface. Os endpoints de interface são alimentados por AWS PrivateLink.

Embora os clusters baseados em instâncias do Amazon DocumentDB não exijam uma interface de conexão de ponto final VPC, AWS PrivateLink você pode acessar de forma privada as operações da API do Amazon DocumentDB sem um gateway de internet, dispositivo NAT, conexão VPN ou conexão. Direct Connect As instâncias do Amazon DocumentDB na VPC não precisam de endereços IP públicos para se comunicarem com endpoints de API do Amazon DocumentDB para iniciar, modificar ou encerrar instâncias e clusters de banco de dados. As instâncias do Amazon DocumentDB também não precisam de endereços IP públicos para usar qualquer uma das operações de API do Amazon DocumentDB disponíveis. O tráfego entre seu VPC e Amazon DocumentDB não sai da rede da Amazon.

Cada endpoint de interface é representado por uma ou mais interfaces de rede elástica nas sub-redes. Para obter mais informações, consulte Interfaces de rede elástica no Guia do usuário do Amazon EC2.

Para obter mais informações sobre VPC endpoints, consulte Acessar e AWS service (Serviço da AWS) usar uma interface VPC endpoint no Guia do usuário do Amazon Virtual Private Cloud ()AWS PrivateLink. Para obter mais informações sobre as operações do Amazon DocumentDB, consulte a Referência da API do Amazon DocumentDB.

Tópicos

- Considerações sobre endpoints da VPC do
- Disponibilidade de regiões
- Como criar um endpoint da VPC de interface para a API do Amazon DocumentDB
- Criação de uma política de endpoint de VPC para a API Amazon DocumentDB

Considerações sobre endpoints da VPC do

Antes de configurar um endpoint da VPC de interface para endpoints da API Amazon DocumentDB, revise os pré-requisitos do endpoint de interface no Guia do usuário da Amazon Virtual Private Cloud (AWS PrivateLink).

Todas as operações da API do Amazon DocumentDB relevantes ao gerenciamento de recursos do Amazon DocumentDB estão disponíveis na VPC usando o AWS PrivateLink.

As políticas de endpoint da VPC têm suporte para endpoints da API do Amazon DocumentDB. Por padrão, o acesso total às operações de API do Amazon DocumentDB é permitido pelo endpoint. Para obter mais informações, consulte Controlar o acesso aos endpoints da VPC usando políticas de endpoint no Guia do usuário do Amazon Virtual Private Cloud (AWS PrivateLink).

Disponibilidade de regiões

Atualmente, a API Amazon DocumentDB oferece suporte a endpoints VPC nos seguintes casos: Regiões da AWS

- Leste dos EUA (Ohio)
- Leste dos EUA (N. da Virgínia)
- Oeste dos EUA (Oregon)
- África (Cidade do Cabo)
- Ásia-Pacífico (Hong Kong)
- Asia Pacific (Mumbai)
- Ásia-Pacífico (Hyderabad)
- Ásia-Pacífico (Osaka)
- Ásia-Pacífico (Seul)
- Ásia-Pacífico (Singapura)
- Ásia-Pacífico (Sydney)
- Ásia-Pacífico (Tóquio)
- Canadá (Central)
- China (Pequim)
- China (Ningxia)
- Europa (Frankfurt)

- Europa (Irlanda)
- Europa (Londres)
- Europa (Paris)
- Europa (Espanha)
- Europa (Milão)
- Oriente Médio (Emirados Árabes Unidos)
- América do Sul (São Paulo)
- AWS GovCloud (US-East)
- AWS GovCloud (US-West)

Como criar um endpoint da VPC de interface para a API do Amazon DocumentDB

Você pode criar um VPC endpoint para a API Amazon DocumentDB usando o Amazon VPC Console ou o (). AWS Command Line Interface AWS CLI Para obter mais informações, consulte Acessar e AWS service (Serviço da AWS) usar uma interface VPC endpoint no Guia do usuário do Amazon Virtual Private Cloud (AWS PrivateLink).

Crie um endpoint da VPC para a API do Amazon DocumentDB usando o nome de serviço `com.amazonaws.region.rds`.

Excluindo Regiões da AWS na China, se você habilitar o DNS privado para o endpoint, poderá fazer solicitações de API para o Amazon DocumentDB com o endpoint VPC usando seu nome DNS padrão para o, por exemplo, rds.us-east-1.amazonaws.com. Regiões da AWS Para a China (Pequim) e a China (Ningxia) Regiões da AWS, você pode fazer solicitações de API com o endpoint VPC usando rds-api.cn-north-1.amazonaws.com .cn e rds-api.cn-northwest-1.amazonaws.com .cn, respectivamente.

Para obter mais informações, consulte Acessar e AWS service (Serviço da AWS) usar uma interface VPC endpoint no Guia do usuário do Amazon Virtual Private Cloud (AWS PrivateLink).

Criação de uma política de endpoint de VPC para a API Amazon DocumentDB

É possível anexar uma política de endpoint ao endpoint da VPC que controla o acesso à API do Amazon DocumentDB. Essa política especifica as seguintes informações:

- A entidade principal que pode realizar ações.
- As ações que podem ser realizadas.
- Os recursos aos quais as ações podem ser aplicadas.

Para obter mais informações, consulte Controlar o acesso aos endpoints da VPC usando políticas de endpoint no Guia do usuário do Amazon Virtual Private Cloud (AWS PrivateLink).

Exemplo: política de endpoint da VPC para ações da API do Amazon DocumentDB

Veja a seguir um exemplo de uma política de endpoint para a API Amazon DocumentDB. Quando anexada a um endpoint, essa política concede acesso às ações indicadas da API do Amazon DocumentDB para todos os principais em todos os recursos.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Principal": "*",
      "Effect": "Allow",
      "Action": [
        "rds:CreateDBInstance",
        "rds:ModifyDBInstance",
        "rds:CreateDBSnapshot"
      ],
      "Resource": "*"
    }
  ]
}
```

Exemplo: política de endpoint de VPC que nega todo o acesso de um determinado Conta da AWS

A seguinte política de VPC endpoint nega a Conta da AWS 123456789012 todo acesso aos recursos usando o endpoint. A política permite todas as ações de outros Contas da AWS.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Resource": "*",
```

```
      "Principal": "*"
    },
    {
      "Action": "*",
      "Effect": "Deny",
      "Resource": "*",
      "Principal": { "AWS": [ "123456789012" ] }
    }
  ]
}
```

Práticas recomendadas de segurança para o Amazon DocumentDB

O Amazon DocumentDB fornece recursos de segurança, incluindo isolamento do Amazon VPC, uso de criptografia em repouso AWS KMS, criptografia em trânsito usando TLS e controle de acesso refinado por meio do IAM. Use as melhores práticas a seguir para proteger seus clusters, instâncias e dados do Amazon DocumentDB.

Use funções AWS Identity and Access Management (IAM) para controlar o acesso às operações da API Amazon DocumentDB. Isso inclui operações que criam, modificam ou excluem recursos do Amazon DocumentDB, como clusters, grupos de segurança e grupos de parâmetros. Quando aplicativos ou outros AWS serviços precisarem acessar os recursos do Amazon DocumentDB, use uma função do IAM para fornecer credenciais temporárias em vez de armazenar credenciais de longo prazo em seu aplicativo. Ao criar funções do IAM, aplique o princípio do menor privilégio. Para obter mais informações, consulte Funções do IAM e Cenários comuns para funções no Guia do usuário do IAM.

- Aplique o menor privilégio com o controle de acesso baseado em função.
- Não use o usuário Conta da AWS root para gerenciar os recursos do Amazon DocumentDB. Em vez disso, use o AWS Identity and Access Management Identity Center para criar usuários humanos com credenciais temporárias. Para obter mais informações, consulte Usuários do IAM Identity Center no Guia do usuário do IAM.
- Conceda a cada identidade o conjunto mínimo de permissões necessárias para realizar suas tarefas.
- Use políticas do IAM para gerenciar permissões e seguir o princípio do menor privilégio. Para obter mais informações, consulte Práticas recomendadas de segurança no IAM no Guia do usuário do IAM.

- Se você usa usuários do IAM com credenciais de longo prazo, alterne essas credenciais regularmente.

  > **Warning**
  >
  > Este cenário precisa de usuários do IAM com acesso programático e credenciais de longo prazo, o que representa um risco de segurança. Para ajudar a reduzir esse risco, recomendamos que você forneça a esses usuários somente as permissões necessárias para realizar a tarefa e que você os remova quando não forem mais necessários. As chaves de acesso podem ser atualizadas, se necessário. Para obter mais informações, consulte Atualizar chaves de acesso no Guia do usuário do IAM.

- Configure o Secrets Manager para alternar automaticamente os segredos do Amazon DocumentDB. Para obter mais informações, consulte Rotação de seus segredos e Rotação de segredos para o Amazon DocumentDB no Guia do AWS Secrets Manager usuário.
- Use Transport Layer Security (TLS) e criptografia em repouso para criptografar seus dados.

Auditar eventos do Amazon DocumentDB

Com o Amazon DocumentDB (compatível com MongoDB), é possível auditar eventos que foram realizados em seu cluster. Exemplos de eventos registrados incluem tentativas de autenticação bem-sucedidas e com falha, eliminação de uma coleção em um banco de dados ou criação de um índice. Por padrão, a auditoria fica desabilitada no Amazon DocumentDB e requer que você opte por esse atributo.

Quando a auditoria está habilitada, o Amazon DocumentDB registra eventos de linguagem de definição de dados (DDL), linguagem de manipulação de dados (DML), autenticação, autorização e gerenciamento de usuários no Amazon Logs. CloudWatch Quando a auditoria está habilitada, o Amazon DocumentDB exporta os registros de auditoria do seu cluster (documentos JSON) para o Amazon Logs. CloudWatch Você pode usar o Amazon CloudWatch Logs para analisar, monitorar e arquivar seus eventos de auditoria do Amazon DocumentDB.

Embora o Amazon DocumentDB não cobre um custo adicional para permitir a auditoria, você paga taxas padrão pelo uso de CloudWatch registros. Para obter informações sobre CloudWatch os preços do Logs, consulte os CloudWatch preços da Amazon.

O recurso de auditoria do Amazon DocumentDB é nitidamente diferente do uso de recursos do serviço com o qual é monitorado. AWS CloudTrail CloudTrail registra operações que são executadas

com o AWS Command Line Interface (AWS CLI) ou Console de gerenciamento da AWS em recursos como clusters, instâncias, grupos de parâmetros e instantâneos. A auditoria de recursos com CloudTrail está ativada por padrão e não pode ser desativada. O atributo de auditoria do Amazon DocumentDB é um atributo opcional. Ele registra operações que ocorrem dentro do seu cluster em objetos, como bancos de dados, coleções, índices e usuários.

Tópicos

- Eventos com suporte
- Ativar a auditoria
- Desativar a auditoria
- Acessar seus eventos de auditoria
- Filtrar eventos de auditoria de DML

Eventos com suporte

A auditoria do Amazon DocumentDB oferece suporte às seguintes categorias de eventos:

- Linguagem de definição de dados (DDL) – inclui operações de gerenciamento de banco de dados, conexões, gerenciamento de usuários e autorização.
- Eventos de leitura da linguagem de manipulação de dados (leituras DML) – incluem `find()` e os vários operadores de agregação, operadores aritméticos, operadores boolianos e outros operadores de consulta de leitura.
- Eventos de gravação da linguagem de manipulação de dados (gravações em DML) – incluem operadores `insert()`, `update()`, `delete()`, e `bulkWrite()`

Os tipos de evento são os seguintes.

Tipo de evento	Categoria	Description
`authCheck`	Autorização	Código de resultado 0: Sucesso
		Código de resultado 13: Tentativas não autorizadas

Tipo de evento	Categoria	Description
		de executar uma operação.
`authenticate`	Conexão	Tentativas de autenticação bem-sucedidas ou com falha em uma nova conexão.
`auditConfigure`	DDL	Auditar configuração de filtro.
`createDatabase`	DDL	Criação de um novo banco de dados.
`createCollection`	DDL	Criação de uma nova coleção em um banco de dados.
`createIndex`	DDL	Criação de um novo índice em uma coleção.
`dropCollection`	DDL	Eliminação de uma coleção em um banco de dados.
`dropDatabase`	DDL	Eliminação de um banco de dados.
`dropIndex`	DDL	Eliminação de um índice em uma coleção.
`modifyChangeStreams`	DDL	O fluxo de alteração foi criado.

Tipo de evento	Categoria	Description
`renameCollection`	DDL	Como renomear uma coleção em um banco de dados.
`createRole`	Gerenciamento de funções	Como criar uma função.
`dropAllRolesFromDatabase`	Gerenciamento de funções	Eliminação de todos os usuários em um banco de dados.
`dropRole`	Gerenciamento de funções	Eliminação de uma função.
`grantPrivilegesToRole`	Gerenciamento de funções	Como conceder privilégios a uma função.
`grantRolesToRole`	Gerenciamento de funções	Como coonceder funções a um perfil definido pelo usuário.
`revokePrivilegesFromRole`	Gerenciamento de funções	Revogação de privilégios de uma função.
`revokeRolesFromRole`	Gerenciamento de funções	Como revogar funções de um perfil definido pelo usuário.
`updateRole`	Gerenciamento de funções	Como atualizar uma função.
`createUser`	Gerenciamento de usuários	Criação de um novo usuário.

Tipo de evento	Categoria	Description
`dropAllUsersFromDatabase`	Gerenciamento de usuários	Eliminação de todos os usuários em um banco de dados.
`dropUser`	Gerenciamento de usuários	Eliminação de um usuário existente.
`grantRolesToUser`	Gerenciamento de usuários	Como conceder funções a um usuário.
`revokeRolesFromUser`	Gerenciamento de usuários	Como revogar funções de um usuário.
`updateUser`	UserManagement	Atualização de um usuário existente.
`insert`	Gravação de DML	Insere um documento ou documentos em uma coleção.
`delete`	Gravação de DML	Exclui um documento ou documentos de uma coleção.
`update`	Gravação de DML	Modifica um documento ou documentos existente s em uma coleção.
`bulkWrite`	Gravação de DML	Executa várias operações de gravação com controles para ordem de execução.

Tipo de evento	Categoria	Description
`setAuditConfig`	Gravação de DML	Definir um novo filtro para auditoria de DML.
`count`	Leitura de DML	Retorna a contagem de documentos que corresponderiam a uma consulta find() para a coleção ou visualização.
`countDocuments`	Leitura de DML	Retorna a contagem de documentos que correspondem a uma consulta para a coleção ou visualiza ção.
`find`	Leitura de DML	Seleciona documento s em uma coleção ou exibição e retorna um cursor para os documentos seleciona dos.
`getAuditConfig`	Leitura de DML	Recuperar o filtro atual para auditoria de DML.
`findAndModify`	Leitura e gravação de DML	Modifica e retorna um único documento.

Tipo de evento	Categoria	Description
`findOneAndDelete`	Leitura e gravação de DML	Exclui um único documento com base nos critérios de filtragem e classific ação, retornando o documento excluído.
`findOneAndReplace`	Leitura e gravação de DML	Substitui um único documento com base no filtro especificado.
`findOneAndUpdate`	Leitura e gravação de DML	Atualiza um único documento com base nos critérios de filtragem e classific ação.
`aggregate`	Leitura e gravação de DML	Oferece suporte a APIs no pipeline de agregação.
`distinct`	Leitura de DML	Encontra os valores distintos de um campo especific ado em uma única coleção ou exibição e retorna os resultados em uma matriz.

> **Note**
>
> Os valores no campo de parâmetro do documento de evento de DML têm um limite de tamanho de 1 KB. O Amazon DocumentDB trunca o valor se ele exceder 1 KB.

> **Note**
>
> Os eventos de exclusão de TTL não são auditados neste momento.

Ativar a auditoria

A ativação da auditoria em um cluster é um processo de duas etapas. Certifique-se de que ambas as etapas sejam concluídas, ou os registros de auditoria não serão enviados para CloudWatch os registros.

Etapa 1. Habilitar o parâmetro de cluster audit_logs

Para habilitar a auditoria, você precisa modificar o parâmetro `audit_logs` no grupo de parâmetros. `audit_logs` é uma lista de eventos delimitada por vírgulas, os quais serão registrados. Os eventos devem ser especificados em letras maiúsculas, e não pode haver espaço em branco entre os elementos da lista.

É possível especificar os seguintes valores para o grupo de parâmetros:

Valor	Description	
`ddl`	Definir isso permitirá a auditoria de eventos DDL, como createDatabase, dropDatabase, createCollection, dropCollection, createIndex, AuthCheck, authenticate, createUser, dropUser, dropUser, grant, revoke, updateUser e drop RolesToUser RolesFromUser AllUsersFromDatabase	

Valor	Description	
`dml_read`	Essa configuração permitirá a auditoria de eventos de leitura de DML, como find, sort count, distinct, group, projecta, unwind, geoNear, geoInters ects, geoWithin e outros operadores de consulta de leitura do MongoDB.	
`dml_write`	Essa configura ção habilitará a auditoria de eventos de gravação de DML, como insert(), update(), delete() e bulkWrite()	
`all`	Essa configuração habilitará a auditoria de eventos do seu banco de dados, como consultas de leitura, consultas de gravação, ações de banco de dados e ações de administr ador.	
`none`	Essa configura ção desabilitará a auditoria	

Valor	Description	
`enabled` (legado)	Essa é uma configuração de parâmetro herdada que é equivalente a 'ddl'. Definir isso permitirá a auditoria de eventos DDL, como createDatabase, dropDatabase, createCollection, dropCollection, createIndex, AuthCheck, authenticate, createUser, dropUser, dropUser, grant, revoke, updateUser e drop. RolesToUser RolesFromUser AllUsersFromDatabase Não é recomendável usar essa configuração, pois ela é uma configuração herdada.	
`disabled` (legado)	Essa é uma configuração de parâmetro herdada que é equivalente a 'none'. Não é recomendável usar essa configuração, pois ela é uma configuração herdada.	

> **Note**
>
> O valor padrão do parâmetro de cluster audit_logs é none (legacy "disabled").

Você também pode usar os valores mencionados acima em combinações.

Valor	Description
`ddl, dml_read`	Essa configuração ativará a auditoria de eventos de DDL e eventos de leitura de DML.
`ddl, dml_write`	Essa configuração ativará a auditoria de eventos de DDL e gravação de DML
`dml_read, dml_write`	Essa configuração habilitará a auditoria para todos os eventos DML.

> **Note**
>
> Não é possível modificar um grupo de parâmetros padrão.

Para obter mais informações, consulte as informações a seguir:

- Criando grupos de parâmetros de cluster do Amazon DocumentDB

  Depois que criar um grupo de parâmetros, modifique-o alterando o valor do parâmetro `audit_logs` para `all`.
- Modificando grupos de parâmetros de cluster do Amazon DocumentDB

Etapa 2. Habilitar a exportação do Amazon CloudWatch Logs

Quando o valor do parâmetro do `audit_logs` cluster é `enabled`, ou `ddl` `dml_read` `dml_write`, você também deve habilitar o Amazon DocumentDB para exportar registros para a Amazon CloudWatch. Se você omitir uma dessas etapas, os registros de auditoria não serão enviados para o. CloudWatch

Ao criar um cluster, realizar uma restauração pontual ou restaurar um snapshot, você pode ativar CloudWatch os registros seguindo estas etapas.

Using the Console de gerenciamento da AWS

Para permitir que o Amazon DocumentDB exporte logs para o CloudWatch uso do console, consulte os seguintes tópicos:

- Ao criar um cluster : em Criação de um cluster e uma instância primária usando o Console de gerenciamento da AWS, consulte Criar um cluster: configurações adicionais, (etapa 5, Exportações de log)
- Ao modificar um cluster existente — Modificar um cluster do Amazon DocumentDB
- Ao executar uma restauração de snapshot de cluster — Restauração de um snapshot de cluster
- Ao executar uma restauração point-in-time — Restauração para um ponto no tempo

Using the AWS CLI

Como habilitar logs de auditoria ao criar um novo cluster

O código a seguir cria o cluster `sample-cluster` e ativa os registros CloudWatch de auditoria.

Example

Para Linux, macOS ou 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
```

Como habilitar logs de auditoria ao modificar um cluster existente

O código a seguir modifica o cluster `sample-cluster` e ativa os registros CloudWatch de auditoria.

Example

Para Linux, macOS ou 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"]}'
```

A saída dessas operações é semelhante ao seguinte (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"
    }
}
```

Desativar a auditoria

Você pode desativar a auditoria desativando a exportação de CloudWatch registros e desativando o parâmetro. `audit_logs`

Desativando a exportação de CloudWatch registros

Você pode desativar a exportação de registros de auditoria usando o Console de gerenciamento da AWS ou o. AWS CLI

Using the Console de gerenciamento da AWS

O procedimento a seguir usa o Console de gerenciamento da AWS para desativar a exportação de registros do Amazon DocumentDB para o. CloudWatch

Como desabilitar logs de auditoria

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Clusters. Depois disso, escolha o botão à esquerda do nome do cluster para o qual você deseja desabilitar a exportação de logs.
3. Escolha Ações e, em seguida, Modificar.
4. Role para baixo até a seção Exportações de log e escolha Desativado.
5. Escolha Continuar.
6. Analise as alterações e escolha quando você deseja que essa mudança seja aplicada ao seu cluster.
   - Aplicar durante a próxima janela de manutenção programada
   - Aplicar imediatamente
7. Escolha Modificar Cluster.

Using the AWS CLI

O código a seguir modifica o cluster `sample-cluster` e desativa os registros de CloudWatch auditoria.

Example

Para Linux, macOS ou 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"]}'
```

A saída dessa operação é semelhante ao seguinte (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"
            }
        ]
    }
}
```

Desativar o parâmetro audit_logs

Para desabilitar o parâmetro `audit_logs` para o cluster, é possível modificar o cluster para que ele use um grupo de parâmetros em que o valor do parâmetro `audit_logs` é `disabled`. Ou é possível modificar o valor do parâmetro `audit_logs` no grupo de parâmetros de cluster para que ele seja `disabled`.

Para saber mais, consulte os seguintes tópicos:

- Modificar um cluster do Amazon DocumentDB
- Modificando grupos de parâmetros de cluster do Amazon DocumentDB

Acessar seus eventos de auditoria

Use as etapas a seguir para acessar seus eventos de auditoria na Amazon CloudWatch.

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. Certifique-se de estar na mesma região que seu cluster do Amazon DocumentDB.
3. No painel de navegação, selecione Logs.
4. Para localizar os logs de auditoria para o seu cluster, na lista, localize e escolha **`/aws/docdb/`*`yourClusterName`*`/audit`**.

   Os eventos de auditoria para cada uma das suas instâncias estão disponíveis em cada um dos respectivos nomes de instância.

Filtrar eventos de auditoria de DML

Conceitos básicos da filtragem de auditoria de DML

Os eventos de auditoria da DML podem ser filtrados antes de serem gravados na Amazon. CloudWatch Para utilizar esse recurso, o registro de auditoria e o log de DML devem estar ativados. O Amazon DocumentDB oferece suporte à filtragem em `atype`, `command`, `user`, `namespace` e `auditAuthorizationSuccess`.

> **Note**
>
> Os eventos de DDL não são filtrados.

É possível ativar a filtragem de auditoria a qualquer momento especificando o filtro de auditoria usando os parâmetros `setAuditConfig`, `filter` e `auditAuthorizationSuccess` na operação `db.adminCommand( { command } )`:

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter:
         {
            //filter conditions
         },
      auditAuthorizationSuccess: true | false
   }
)
```

Você também pode recuperar as configurações do filtro de auditoria executando o seguinte comando:

```
db.admin.runCommand( { getAuditConfig: 1})
```

Requisitos de segurança

Somente o banco de dados users/roles com ação privilegiada `auditConfigure` pode executar os comandos acima `admindb` ao definir ou listar filtros de auditoria DML. É possível usar um dos perfis integrados de [`clusterAdmin`, `hostManager`, `root`] ou criar perfis personalizados com

privilégios `auditConfigure`. Veja a seguir um exemplo do uso de funções existentes com o privilégio `auditConfigure` e um exemplo com funções personalizadas.

Usuário com função integrada:

```
use admin
db.createUser(
  {
    user: "myClusterAdmin",
    pwd: "password123",
    roles: [ { role: "clusterAdmin", db: "admin" } ]
  }
)
```

Usuário com funções personalizadas:

```
use admin
db.createRole(
   {
     role: "myRole",
     privileges: [
       { resource: { cluster: true }, actions: [ "auditConfigure" ] }
     ],
     roles: []
   }
)
db.createUser(
  {
    user: "myUser",
    pwd: "myPassword",
    roles: [ { role: "myRole", db: "admin" } ]
  }
)
```

Filtrar casos de uso

Exemplo: filtrar eventos por comandos

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {
```

```
        "$and": [
         {
            "param.command":
               {
                  $in: [ "find","count", "insert", "delete", "update",
 "findandmodify" ]
               }
         }
         ]
      },
      auditAuthorizationSuccess: true
   }
)
```

Exemplo: filtrar eventos por nome de usuário

Neste exemplo, somente o usuário “myUser” será registrado:

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {
      "$and": [
         {
            "param.user":
               {
                  $in: [ "myUser" ]
               }
         }
         ]},
      auditAuthorizationSuccess: true})
```

Exemplo: filtrar por **`atype`**

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {atype: "authCheck"},
      auditAuthorizationSuccess: true
   })
```

> **Note**
>
> Todos os logs de DML têm `authCheck` como `atype`. Somente DDL tem um `atype` diferente. Se você colocar um valor diferente `authCheck` `defilter`, ele não produzirá um login CloudWatch DML.

Exemplo: filtrar usando vários 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**
>
> No nível superior, somente `$and`, `$or` e `$nor` têm suporte. Quaisquer outros operadores não têm suporte e causarão um erro.

Exemplo: filtragem por eventos por **`auditAuthorizationSuccess`**

Nesse filtro, todos os comandos aprovados com sucesso na autorização não serão registrados:

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {},
      auditAuthorizationSuccess: false
   }
)
```

Exemplo: filtrar com as condições **`$in`** e **`$nin`**

Ao usar tanto em `$in` quanto em `$nin`, o comando não será registrado, pois haverá um “e” implícito entre as condições. Neste exemplo, a regex bloqueará o comando `find` para que nada seja registrado:

```
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})
```

Exemplo: filtrar por **`namespace`**

```
db.admin.runCommand(
```

```
   {
      setAuditConfig: 1,
      filter: {
      "$and": [
         {
            "param.ns":
               {
                  $in: [ "test.foo" ]
               }
         }
         ]},
      auditAuthorizationSuccess: true})
```

Exemplo: redefinir para o filtro padrão

Redefinir para o valor padrão significa que cada evento de auditoria de DML será registrado. Para redefinir a filtragem para o valor padrão, execute o comando a seguir:

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {},
      auditAuthorizationSuccess: true
   }
)
```

Amazon VPC e o Amazon DocumentDB

O Amazon Virtual Private Cloud (Amazon VPC) possibilita que você inicie AWS recursos, como instâncias do Amazon DocumentDB, em uma nuvem privada virtual (VPC).

Ao usar uma VPC, você tem controle sobre o ambiente de rede virtual. É possível escolher seu próprio intervalo de endereços IP, criar sub-redes e configurar o roteamento e listas de controle de acesso. Não há custos adicionais para executar seu cluster em uma VPC.

As contas têm uma VPC padrão. Todos os novos clusters são criados na VPC padrão, a menos que você especifique o contrário.

Tópicos

- Clusters do DocumentDB em uma VPC

- Acesso a um cluster do Amazon DocumentDB em uma VPC
- Crie uma IPv4-only VPC para uso com um cluster DocumentDB
- Criação de uma VPC de pilha dupla para uso com um cluster do DocumentDB

Veja a seguir uma discussão sobre a funcionalidade da VPC relevante para clusters do Amazon DocumentDB. Para obter mais informações sobre o Amazon VPC, consulte o Guia do usuário da Amazon VPC.

Clusters do DocumentDB em uma VPC

Seu cluster do Amazon DocumentDB está em uma nuvem privada virtual (VPC). Uma VPC é uma rede virtual que está logicamente isolada de outras redes virtuais no. Nuvem AWS O Amazon VPC possibilita que você inicie AWS recursos, como um cluster do Amazon DocumentDB ou uma instância do Amazon EC2, em uma VPC. A VPC pode ser uma VPC padrão que vem com sua conta ou aquele que você criou. Todas as VPCs estão associadas à sua AWS conta.

Sua VPC padrão possui três sub-redes que você pode usar para isolar recursos dentro da VPC. A VPC padrão também possui um gateway da Internet que pode ser usado para fornecer acesso a recursos na VPC de fora da VPC.

Para cenários envolvendo clusters do Amazon DocumentDB em uma VPC e fora de uma VPC, consulte Acesso a um cluster do Amazon DocumentDB em uma VPC.

Tópicos

- Trabalho com clusters em uma VPC
- Trabalho com grupos de sub-redes
- Endereçamento IP do Amazon DocumentDB
- Criar um cluster em uma VPC

Trabalho com clusters em uma VPC

Veja a seguir algumas dicas sobre como trabalhar com um cluster em uma VPC:

- A VPC deve ter pelo menos duas sub-redes. Essas sub-redes devem estar em duas zonas de disponibilidade diferentes no local em Região da AWS que você deseja implantar seu cluster. Uma sub-rede é um segmento do intervalo de endereços IP de uma VPC que pode ser especificada e que permite agrupar clusters com base nas suas necessidades operacionais e de segurança.

- Se quiser que seu cluster na VPC seja publicamente acessível, certifique-se de ativar os atributos `DNS hostnames` e `DNS resolution` da VPC.
- Sua VPC deve ter um grupo de sub-redes criado por você. Você cria um grupo de sub-redes especificando as sub-redes criadas. O Amazon DocumentDB escolhe uma sub-rede e um endereço IP dentro desse grupo de sub-redes para associar à instância primária no seu cluster. A instância primária usa a zona de disponibilidade que contém a sub-rede.
- Sua VPC deve ter um grupo de segurança da VPC que permita o acesso ao cluster.

  Para obter mais informações, consulte Acesso a um cluster do Amazon DocumentDB em uma VPC.
- Os blocos CIDR em cada uma das suas sub-redes devem ser suficientemente grandes para acomodar endereços IP sobressalentes para o Amazon DocumentDB usar durante atividades de manutenção, incluindo failover e escalabilidade de computação. Por exemplo, um intervalo como 10.0.0. 0/24 e 10.0.1. 0/24 normalmente é grande o suficiente.
- Uma VPC pode ter um atributo `instance tenancy` de `default` ou `dedicated`. Todas as VPC padrão têm o atributo `instance tenancy` definido como padrão, e uma VPC padrão pode oferecer suporte a qualquer classe de instância de banco de dados.

  Se você optar por manter seu cluster em uma VPC dedicada onde o atributo `instance tenancy` esteja definido como `dedicated`, a classe do seu cluster deverá ser de um dos tipos aprovados de instância dedicada do Amazon EC2. Por exemplo, a instância dedicada `r5.large` do EC2 corresponde à classe de instância `db.r5.large`. Para obter informações sobre a locação de instâncias em uma VPC, consulte Instâncias do Amazon EC2 no Guia do usuário do Amazon Elastic Compute Cloud.

  Consulte mais informações sobre os tipos de instância que podem estar em uma instância dedicada em Instâncias dedicadas do Amazon EC2, na página de preços do Amazon EC2.

  > **Note**
  >
  > Quando você define o atributo `instance tenancy` como `dedicated` para um cluster, isso não garante que o cluster terá execução em um host dedicado.

Trabalho com grupos de sub-redes

Sub-redes são segmentos do intervalo de endereços IP de uma VPC que você designa para agrupar seus recursos com base em necessidades operacionais e de segurança. Um grupo de sub-redes é uma coleção de sub-redes (geralmente privadas) que você cria em uma VPC e designa aos seus clusters. Ao usar um grupo de sub-redes, você pode especificar uma VPC específica ao criar clusters usando a API AWS CLI ou o Amazon DocumentDB. Se você usar o console, poderá escolher somente a VPC e os grupos de sub-redes que deseja usar.

Cada grupo de sub-redes deve ter sub-redes em pelo menos duas zonas de disponibilidade em uma determinada Região da AWS. Ao criar um cluster em uma VPC, escolha um grupo de sub-redes para eles. Dentre o grupo de sub-redes, o Amazon DocumentDB escolhe uma sub-rede e um endereço IP dentro dessa sub-rede para associar à instância primária no seu cluster. O banco de dados usa a zona de disponibilidade que contém a sub-rede. O DocumentDB sempre atribui um endereço IP de uma sub-rede que tenha espaço de endereço IP livre.

As sub-redes em um grupo de sub-redes são públicas ou privadas, dependendo da configuração definida para suas listas de controle de acesso à rede (ACLs da rede) e tabelas de rotas. Para que um cluster seja acessível ao público, todas as sub-redes em seu grupo de sub-redes devem ser públicas. Se uma sub-rede associada a um cluster acessível ao público mudar de pública para privada, ela poderá afetar a disponibilidade do cluster.

Para criar um grupo de sub-redes que ofereça suporte ao modo de pilha dupla, verifique se cada sub-rede adicionada ao grupo de sub-redes tem um bloco CIDR do Internet Protocol versão 6 (IPv6) associado a ela. Para obter mais informações, consulte Endereçamento IP do Amazon DocumentDB e Suporte a IPv6 para sua VPC no Guia do usuário da Amazon Virtual Private Cloud.

Quando o Amazon DocumentDB cria um cluster em uma VPC, ele atribui uma interface de rede ao cluster usando um endereço IP selecionado dentre seu grupo de sub-redes. Sempre use o nome do Sistema de Nomes de Domínio (DNS) para se conectar ao seu cluster, pois o endereço IP subjacente muda durante o failover.

> **Note**
>
> Para cada cluster executado em uma VPC, certifique-se de reservar pelo menos um endereço em cada sub-rede no grupo de sub-redes para uso pelo Amazon DocumentDB para ações de recuperação.

Sub-redes compartilhadas

É possível criar um cluster em uma VPC compartilhada.

Algumas considerações que você deve ter em mente ao usar VPCs compartilhadas:

- É possível mover um cluster de uma sub-rede de VPC compartilhada para uma sub-rede de VPC não compartilhada e vice-versa.
- Os participantes de uma VPC compartilhada devem criar um grupo de segurança na VPC para permitir que criem um cluster.
- Proprietários e participantes de uma VPC compartilhada podem acessar o banco de dados usando consultas do DocumentDB. No entanto, somente o criador de um recurso pode fazer chamadas de API no recurso.

Endereçamento IP do Amazon DocumentDB

Os endereços IP habilitam recursos na sua VPC para se comunicar com outros e com recursos na Internet. O Amazon DocumentDB oferece suporte a protocolos de endereçamento IPv4 e IPv6. Por padrão, o Amazon DocumentDB e a Amazon VPC usam o protocolo de endereçamento IPv4. Você não pode desativar esse comportamento. Ao criar uma VPC, especifique um bloco CIDR IPv4 (um intervalo de endereços IPv4 privados). Você também pode atribuir um bloco CIDR IPv6 à VPC e às sub-redes, bem como endereços IPv6 desse bloco a clusters em sua sub-rede.

> **Note**
>
> Dual-stack O modo (endereçamento IPv6) só é suportado nas versões 4.0 e 5.0 do Amazon DocumentDB.

A compatibilidade com o protocolo IPv6 expande o número de endereços IP compatíveis. Ao usar o protocolo IPv6, você tem a garantia de que terá endereços disponíveis suficientes para o crescimento futuro da Internet. Os recursos novos e existentes do DocumentDB podem usar endereços IPv4 e IPv6 na VPC. Configurar, proteger e converter o tráfego de rede entre os dois protocolos usados em diferentes partes de uma aplicação pode causar sobrecarga operacional. É possível padronizar o protocolo IPv6 para recursos do Amazon DocumentDB a fim de simplificar sua configuração de rede.

Tópicos

- Endereços IPv4
- Endereços IPv6
- Dual-stack modo

Endereços IPv4

Quando você cria uma VPC, é necessário especificar um intervalo de endereços IPv4 para a VPC em forma de um bloco CIDR, como `10.0.0.0/16`. Um grupo de sub-redes define o intervalo de endereços IP nesse bloco CIDR que um cluster pode usar. Esses endereços IP podem ser públicos ou privados.

Um endereço IPv4 privado é um endereço IP que não é acessível pela internet. É possível usar endereços IPv4 privados para comunicação entre seu cluster e outros recursos, como instâncias do Amazon EC2, na mesma VPC. Cada cluster tem um endereço IP privado para comunicação na VPC.

Um endereço IP público é um endereço IPv4 acessível pela Internet. Endereços IP públicos não são permitidos para seu cluster do DocumentDB. Qualquer endereço IP público deve ser resolvido pelo gateway da Internet e pelo EC2 na sub-rede pública.

Para ver como criar uma VPC somente com endereços IPv4 privados que podem ser usados para um cenário comum do Amazon DocumentDB, consulte Crie uma IPv4-only VPC para uso com um cluster DocumentDB.

Endereços IPv6

Como opção, você pode associar um bloco CIDR IPv6 a sua VPC e sub-redes e atribuir endereços IPv6 desse bloco a recursos em sua VPC. Todo endereço IPv6 é globalmente exclusivo.

O bloco CIDR IPv6 da VPC é automaticamente atribuído do grupo de endereços IPv6 da Amazon. Você não pode escolher o intervalo.

Ao se conectar a um endereço IPv6, verifique se as seguintes condições são atendidas:

- O cliente está configurado para que o tráfego do cliente para o banco de dados via IPv6 seja permitido.
- Os grupos de segurança do DocumentDB usados pelo cluster estão configurados corretamente para que o tráfego do cliente para o banco de dados via IPv6 seja permitido.

- A pilha do sistema operacional do cliente permite tráfego no endereço IPv6, e os drivers e bibliotecas do sistema operacional estão configurados para escolher o endpoint de cluster padrão correto (IPv4 ou IPv6).

Para obter mais informações sobre IPv6, consulte Endereçamento IP no Guia do usuário da Amazon Virtual Private Cloud.

Dual-stack modo

Quando um cluster consegue se comunicar pelos protocolos de endereçamento IPv4 e IPv6, a execução ocorre em modo de pilha dupla. Portanto, os recursos podem se comunicar com o cluster por IPv4, IPv6 ou ambos. O DocumentDB desabilita o acesso ao gateway da Internet para endpoints de IPv6 de clusters de banco de dados privados no modo de pilha dupla. O DocumentDB faz isso para garantir que os endpoints de IPv6 sejam privados e possam ser acessados somente de dentro da VPC.

Tópicos

- Dual-stack grupos de modo e sub-rede
- Trabalho com clusters no modo de pilha dupla
- Modificação de IPv4-only clusters para usar o modo de pilha dupla
- Dual-stack modo Disponibilidade de região e versão
- Limitações para clusters de rede de pilha dupla

Dual-stack grupos de modo e sub-rede

Para usar o modo de pilha dupla, verifique se cada sub-rede no grupo de sub-redes que você associa ao cluster tem um bloco CIDR IPv6 associado a ela. É possível criar um grupo de sub-redes ou modificar um existente para atender a esse requisito. Depois que um cluster entra no modo de pilha dupla, os clientes podem se conectar normalmente. Os firewalls de segurança do cliente e os grupos de segurança de clusters do DocumentDB devem ser configurados com precisão para permitir tráfego por IPv6. Para se conectar, os clientes usam o endpoint do cluster. As aplicações cliente podem especificar qual protocolo é o preferencial ao se conectar a um banco de dados. No modo de pilha dupla, o cluster detecta o protocolo de rede de preferência do cliente, IPv4 ou IPv6, e o usa para a conexão.

Se um grupo de sub-redes deixar de ser compatível com o modo de pilha dupla devido à exclusão de sub-rede ou desassociação do CIDR, existe o risco de um estado de rede incompatível para clusters

associados ao grupo de sub-redes. Além disso, não é possível usar o grupo de sub-redes ao criar um novo cluster no modo de pilha dupla.

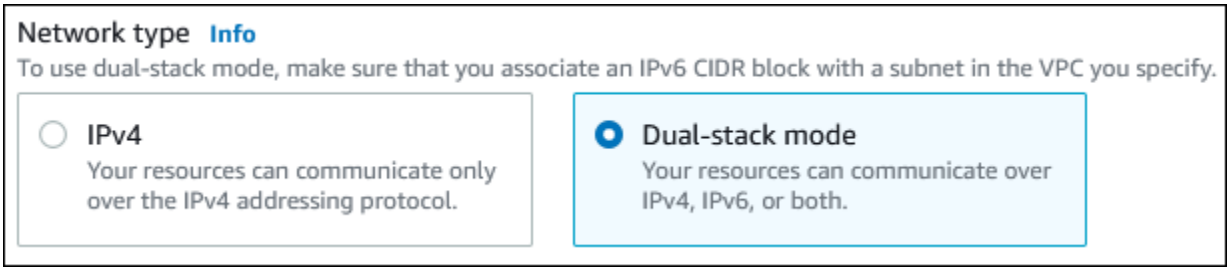
Para determinar se um grupo de sub-redes oferece suporte ao modo de pilha dupla usando o Console de gerenciamento da AWS, veja o Tipo de rede na página de detalhes do grupo de sub-redes. Para determinar se um grupo de sub-redes oferece suporte ao modo de pilha dupla usando o AWS CLI, execute o `describe-db-subnet-groups` comando e `SupportedNetworkTypes` visualize na saída.

As réplicas de leitura são tratadas como instâncias clusters independentes e podem ter um tipo de rede diferente do cluster primário. Se você alterar o tipo de rede do cluster primário de uma réplica de leitura, ela não será afetada. É possível restaurar um cluster para qualquer tipo de rede com suporte.

Trabalho com clusters no modo de pilha dupla

Ao criar ou modificar um cluster, é possível especificar o modo de pilha dupla para permitir que os recursos se comuniquem com o cluster por IPv4, IPv6 ou ambos.

Ao usar o Console de gerenciamento da AWS para criar ou modificar um cluster, você pode especificar o modo de pilha dupla na seção Tipo de rede. A imagem a seguir mostra a seção Tipo de rede no console:



Ao usar o AWS CLI para criar ou modificar um cluster, defina a `--network-type` opção como `DUAL` usar o modo de pilha dupla. Ao usar a API do DocumentDB para criar ou modificar um cluster, defina o parâmetro `NetworkType` como `DUAL` para usar o modo de pilha dupla. Se não houver suporte para o modo de pilha dupla na versão do mecanismo do DocumentDB especificado ou no grupo de sub-redes, o erro `NetworkTypeNotSupported` será retornado.

Para obter mais informações sobre como criar um cluster, consulte Criar um cluster do Amazon DocumentDB. Para obter mais informações sobre como modificar um cluster, consulteModificar um cluster do Amazon DocumentDB.

Para determinar se um cluster está no modo de pilha dupla usando o console, observe o Tipo de rede na guia Conectividade e segurança para o cluster.

Modificação de IPv4-only clusters para usar o modo de pilha dupla

Você pode modificar um IPv4-only cluster para usar o modo de pilha dupla. Para isso, altere o tipo de rede do cluster.

É recomendável que você altere o tipo de rede do seu cluster do Amazon DocumentDB durante uma janela de manutenção. É possível definir o tipo de rede manualmente usando o comando `modify-db-cluster`.

Antes de modificar um cluster para usar o modo de pilha dupla, certifique-se de que seu grupo de sub-redes ofereça suporte ao modo de pilha dupla. Se o grupo de sub-redes associado ao cluster não oferece suporte ao modo de pilha dupla, especifique um grupo de sub-redes com suporte adequado ao modificar o cluster. Modificação do grupo de sub-redes de um cluster pode ocasionar tempo de inatividade.

Se você modificar o grupo de sub-redes de um cluster antes de alterar o cluster para usar o modo de pilha dupla, certifique-se de que o grupo de sub-redes seja válido para o cluster antes e depois da alteração.

Execute a `ModifyDBCluster` chamada somente com o `NetworkType` parâmetro definido para alterar `DUAL` a rede para o modo de pilha dupla. O acréscimo de outros parâmetros junto com `NetworkType` na mesma chamada de API pode ocasionar tempo de inatividade. Para modificar vários parâmetros, primeiro é preciso concluir com êxito a modificação do tipo de rede para só então enviar outra solicitação `ModifyDBCluster` com outros parâmetros.

Se você não conseguir se conectar ao cluster após a alteração, verifique se os firewalls de segurança do cliente e do banco de dados e as tabelas de rotas estão configurados com precisão para permitir tráfego para o banco de dados na rede selecionada (IPv4 ou IPv6). Você também pode precisar modificar parâmetros, bibliotecas ou drivers do sistema operacional para se conectar por meio de um endereço IPv6.

Para modificar um IPv4-only cluster para usar o modo de pilha dupla

1. Modifique um grupo de sub-redes para oferecer suporte ao modo de pilha dupla ou crie um grupo de sub-redes que ofereça suporte a esse modo:

    a. Associe um bloco CIDR IPv6 à VPC.

Para obter instruções, consulte Adicionar ou remover um bloco CIDR na sua VPC no Guia do usuário da Amazon VPC.

b. Anexe o bloco CIDR IPv6 a todas as sub-redes do grupo de sub-redes.

Para obter instruções, consulte Adição ou remoção de um bloco CIDR IPv6 na sua sub-rede no Guia do usuário da Amazon Virtual Private Cloud.

c. Verifique se o grupo de sub-redes oferece suporte ao modo de pilha dupla.

Se você estiver usando o Console de gerenciamento da AWS, selecione o grupo de sub-redes e certifique-se de que o valor dos tipos de rede suportados seja Duplo.

Se você estiver usando o AWS CLI, execute o `describe-db-subnet-groups` comando e certifique-se de que o `SupportedNetworkType` valor do cluster seja`Dual`.

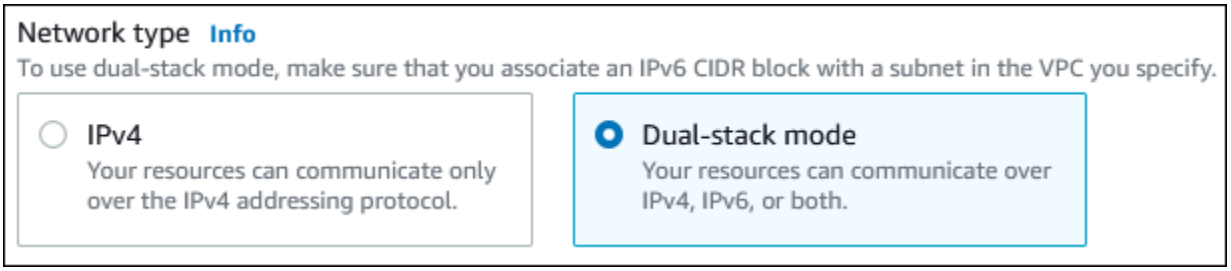
2. Modifique o grupo de segurança associado ao cluster para permitir conexões IPv6 com o banco de dados ou crie um grupo de segurança que permita conexões IPv6.

Para obter instruções, consulte Regras dos grupos de segurança no Guia do usuário da Amazon Virtual Private Cloud.

3. Modifique o cluster para oferecer suporte ao modo de pilha dupla. Para fazer isso, defina o Tipo de rede como Dual-stack modo.

Se você estiver usando o console, verifique se as seguintes configurações estão corretas:

- Tipo de rede — Dual-stack modo



- Grupo de sub-redes: o grupo de sub-redes que você configurou em uma etapa anterior.
- Grupo de segurança: o grupo de segurança que você configurou em uma etapa anterior.

Se você estiver usando o AWS CLI, verifique se as seguintes configurações estão corretas:

- `--network-type` — `dual`

- `--db-subnet-group-name`: o grupo de sub-redes que você configurou em uma etapa anterior
- `--vpc-security-group-ids`: o grupo de segurança da VPC que você configurou em uma etapa anterior.

Por exemplo:

```
aws docdb modify-db-cluster --db-cluster-identifier <cluster-name> --network-type
 "DUAL"
```

4. Verifique se o cluster oferece suporte ao modo de pilha dupla.

   Se você estiver usando o console, escolha a guia Conectividade e segurança do cluster e verifique se o valor do tipo de rede é Dual-stack modo.

   Se você estiver usando o AWS CLI, execute o `describe-db-cluster` comando e certifique-se de que o `NetworkType` valor do cluster sejadual.

   Execute o comando `dig` no endpoint do cluster para identificar o endereço IPv6 associado a ele.

   ```
   dig <db-cluster-endpoint> AAAA
   ```

   Use o endpoint do cluster, não o endereço IPv6, para se conectar ao cluster.

Dual-stack modo Disponibilidade de região e versão

A disponibilidade e a compatibilidade do recurso varia entre as Regiões da AWS.

Suporte de região

Atualmente, o modo de pilha dupla para o Amazon DocumentDB está disponível em todos os comerciais. Regiões da AWS

Suporte de versão

Dual-stack o modo é compatível com as versões 4.0, 5.0 e 8.0 do Amazon DocumentDB. Se você não conseguir acessar o modo de pilha dupla em nenhuma dessas versões, verifique se está executando a versão mais recente do patch de mecanismo em seu cluster.

Limitações para clusters de rede de pilha dupla

As limitações a seguir se aplicam a clusters de rede de pilha dupla:

- Os clusters não podem usar o protocolo IPv6 exclusivamente. É possível usar IPv4 exclusivamente ou o protocolo IPv4 e IPv6 (modo de pilha dupla).
- O Amazon DocumentDB não oferece suporte a sub-redes IPv6 nativas.
- Os clusters que usam o modo de pilha dupla devem ser privados. Eles não podem ser acessíveis ao público.

Criar um cluster em uma VPC

Os procedimentos a seguir ajudam você a criar um cluster em uma VPC. Para usar a VPC padrão, é possível começar na etapa 2 e usar a VPC e o grupo de sub-redes que já foram criados para você. Você também pode criar VPCs adicionais, caso seja necessário.

> **Note**
>
> Se você deseja que seu cluster na VPC seja acessível ao público geral, atualize as informações de DNS da VPC, habilitando os atributos `DNS hostnames` e `DNS resolution` da VPC. Para obter informações sobre como atualizar o DNS para uma instância de VPC, consulte Visualização e atualização de atributos do DNS para sua VPC no Guia do usuário da Amazon Virtual Private Cloud.

Siga estas etapas para criar um cluster em uma VPC:

- Etapa 1: criar uma VPC
- Etapa 2: criar um grupo de sub-redes
- Etapa 3: Criar um grupo de segurança da VPC
- Etapa 4: criação de um cluster na VPC

Etapa 1: criar uma VPC

Crie uma VPC com duas sub-redes em pelo menos duas zonas de disponibilidade. Você usará essas sub-redes ao criar um grupo de sub-redes. Se você tiver uma VPC padrão, uma sub-rede será criada automaticamente para você em cada zona de disponibilidade na Região da AWS.

Para obter mais informações, consulte Crie uma IPv4-only VPC para uso com um cluster DocumentDB, ou consulte Criação de uma VPC no Guia do usuário da Amazon Virtual Private Cloud.

Etapa 2: criar um grupo de sub-redes

Um grupo de sub-redes é uma coleção de sub-redes (geralmente privadas) que você cria para uma VPC e designa aos seus clusters. Um grupo de sub-redes permite que você especifique uma VPC específica ao criar clusters usando a API AWS CLI ou DocumentDB. Se você usar o Console de gerenciamento da AWS, basta escolher a VPC e as sub-redes que deseja usar. Cada grupo de sub-redes deve ter pelo menos uma sub-rede em pelo menos duas zonas de disponibilidade na Região da AWS. Como prática recomendada, cada grupo de sub-redes deve ter no mínimo uma sub-rede para cada zona de disponibilidade na Região da AWS.

Para que um cluster seja acessível publicamente, as sub-redes no grupo de sub-redes devem ter um gateway da Internet. Para obter mais informações sobre gateways da Internet para sub-redes, consulte Habilitação de acesso da Internet a uma VPC usando um gateway da Internet no Guia do usuário da Amazon Virtual Private Cloud.

> **Note**
>
> O grupo de sub-redes para uma zona local pode ter somente uma sub-rede.

Ao criar um cluster em uma VPC, é possível escolher um grupo de sub-redes. O Amazon DocumentDB escolhe uma sub-rede e um endereço IP (dentro dessa sub-rede) para associar ao seu cluster. Se não houver nenhum grupo de sub-redes, o Amazon DocumentDB criará um grupo de sub-redes padrão quando você criar o cluster. O DocumentDB cria e associa uma interface de rede elástica ao seu cluster com esse endereço IP. O cluster usa a zona de disponibilidade que contém a sub-rede.

Nesta etapa, você criará um grupo de sub-redes e adicionará as sub-redes criadas à sua VPC.

Para criar um grupo de sub-redes

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Grupos de sub-redes.
3. Escolha Criar.

4. Em Nome, insira o nome do grupo de sub-redes.
5. Em Descrição, insira uma descrição para o grupo de sub-redes.
6. Na seção Adicionar sub-redes, em VPC, escolha a VPC padrão ou a VPC que você criou. Em seguida, escolha as zonas de disponibilidade que incluem as sub-redes de Zonas de disponibilidade e escolha as sub-redes de Sub-redes.
7. Escolha Criar.

   Seu novo grupo de sub-redes será exibido na lista Grupos de sub-redes, no console do DocumentDB. É possível selecionar o grupo de sub-redes para obter detalhes, incluindo todas as sub-redes associadas a esse grupo, no painel de detalhes, na parte inferior da janela.

Etapa 3: Criar um grupo de segurança da VPC

Antes de criar seu cluster, crie um grupo de segurança da VPC para associar a ele. Se você não criar um grupo de segurança da VPC, poderá usar o grupo de segurança padrão ao criar o cluster. Para obter instruções sobre como criar um grupo de segurança para seu clusterCrie uma IPv4-only VPC para uso com um cluster DocumentDB, consulte ou consulte Controlar o tráfego para seus AWS recursos usando grupos de segurança no Guia do usuário da Amazon Virtual Private Cloud.

Etapa 4: criação de um cluster na VPC

Nesta etapa, você cria um cluster e usa o nome da VPC, o grupo de sub-redes e o grupo de segurança de VPC que você criou nas etapas anteriores.

> **Note**
>
> Se você deseja que seu cluster na VPC seja acessível ao público, atualize os atributos `DNS hostnames` e `DNS resolution` da VPC. Para obter mais informações, consulte Visualização e atualização de atributos de DNS da sua VPC no Guia do usuário da Amazon Virtual Private Cloud.

Para obter detalhes sobre como criar um cluster, consulte Criar um cluster do Amazon DocumentDB.

Quando solicitado na seção Conectividade, insira o nome da VPC, o grupo de sub-redes e o grupo de segurança da VPC.

> **Note**
>
> Não há suporte no momento para a atualização de VPCs em clusters do DocumentDB.

Acesso a um cluster do Amazon DocumentDB em uma VPC

O Amazon DocumentDB oferece suporte aos cenários a seguir para acesso a um cluster em uma VPC:

Tópicos

- Um cluster em uma VPC acessado por uma instância do Amazon EC2 na mesma VPC.
- Um cluster em uma VPC acessada por uma instância do Amazon EC2 em uma VPC diferente.

Um cluster em uma VPC acessado por uma instância do Amazon EC2 na mesma VPC.

Um uso comum de um cluster em uma VPC é compartilhar dados com um servidor de aplicações que está sendo executado em uma instância do Amazon EC2 na mesma VPC.

A maneira mais simples de gerenciar o acesso entre instâncias do EC2 e clusters na mesma VPC é fazer o seguinte:

- Crie um grupo de segurança de VPC para seus clusters. Esse grupo de segurança pode ser usado para restringir o acesso aos clusters. Por exemplo, você pode criar uma regra personalizada para esse grupo de segurança. Isso pode permitir o acesso TCP usando a porta que você atribuiu ao cluster quando o criou, e um endereço IP para acessar o cluster para desenvolvimento ou outros fins.
- Crie um grupo de segurança da VPC em que as suas instâncias do EC2 (servidores Web e clientes) estarão. Esse grupo de segurança pode, se necessário, permitir o acesso à instância do EC2 pela Internet usando a tabela de roteamento da VPC. Por exemplo, você pode definir regras nesse grupo de segurança para permitir o acesso TCP à instância do EC2 pela porta 22.
- Crie regras personalizadas no grupo de segurança para seus clusters que permitam conexões do grupo de segurança que você criou para suas instâncias do EC2. Essas regras podem permitir que qualquer membro do grupo de segurança acesse os clusters.

Há uma sub-rede pública e privada adicional em uma zona de disponibilidade separada. Um grupo de sub-redes do DocumentDB exige uma sub-rede em, pelo menos, duas zonas de disponibilidade. A sub-rede adicional facilita a alternância para uma implantação de cluster multi-AZ no futuro.

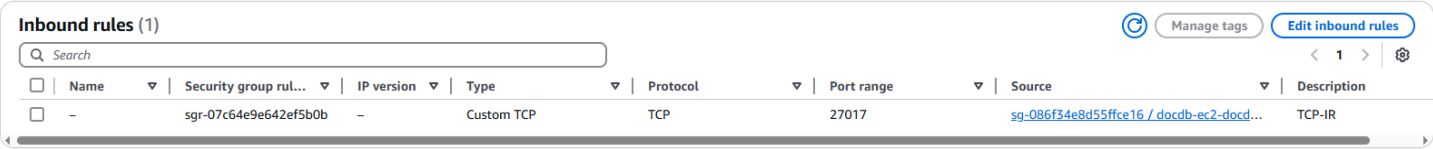
Para obter instruções sobre como criar uma VPC com sub-redes públicas e privadas para esse cenário, consulte Crie uma IPv4-only VPC para uso com um cluster DocumentDB.

> **Tip**
>
> É possível configurar a conectividade de rede entre uma instância do Amazon EC2 e um cluster do DocumentDB automaticamente ao criar o cluster. Para obter mais informações, consulte Conectar o Amazon EC2 automaticamente.

Para criar uma regra em um grupo de segurança da VPC que permita conexões de outro grupo de segurança, faça o seguinte:

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon VPC em https://console.aws.amazon.com/vpc.
2. No painel de navegação, localize e selecione Grupos de segurança.
3. Escolha ou crie um grupo de segurança ao qual você deseja conceder acesso para membros de outro grupo de segurança. Esse é o grupo de segurança que você usa para seus clusters. Vá para a guia Inbound Rules (Regras de entrada) e escolha Edit rules (Editar regras).
4. Na página Edit inbound rules (Editar regras de entrada), escolha Add Rule (Adicionar regra).
5. Em Tipo, escolha a entrada que corresponda à porta usada ao criar seu cluster, como TCP personalizado.
6. No campo Origem, comece a digitar o ID do grupo de segurança para listar os grupos de segurança correspondentes. Escolha o grupo de segurança com membros que você deseja que tenham acesso aos recursos protegidos por esse grupo de segurança. Esse é o grupo de segurança usado para sua instância do EC2 no cenário anterior.
7. Se necessário, repita as etapas para o protocolo TCP, criando uma regra com Todos os TCP como Tipo e seu grupo de segurança no campo Origem. Se você pretende usar o protocolo UDP, crie uma regra com All UDP (Todos os UDP) como Type (Tipo) e seu grupo de segurança em Source (Origem).
8. Selecione Salvar regras.

A tela a seguir mostra uma regra de entrada com um grupo de segurança como origem.



Para obter mais informações sobre como se conectar a um cluster a partir da sua instância do EC2, consulte Conectar o Amazon EC2 automaticamente.

Um cluster em uma VPC acessada por uma instância do Amazon EC2 em uma VPC diferente.

Quando seus clusters estão em uma VPC diferente da instância do EC2 que você está usando para acessá-los, é possível usar emparelhamento de VPC para acessar o cluster.

Uma conexão de emparelhamento da VPC é uma conexão de redes entre duas VPCs que permite direcionar o tráfego entre elas usando endereços IP privados. Recursos em qualquer VPC podem se comunicar uns com os outros como se estivessem na mesma rede. Você pode criar uma conexão de emparelhamento de VPC entre suas próprias VPCs, com uma VPC em outra AWS conta ou com uma VPC em outra. Região da AWS Para saber mais sobre o emparelhamento de VPCs, consulte Emparelhamento de VPCs, no Guia do usuário do Amazon Virtual Private Cloud.

Crie uma IPv4-only VPC para uso com um cluster DocumentDB

Um cenário comum inclui um cluster em uma nuvem privada virtual (VPC) com base no serviço Amazon VPC. Por exemplo, essa VPC poderia compartilhar dados com um serviço ou aplicação em execução na mesma VPC. Neste tópico, você criará a VPC para esse cenário.

Tópicos

- Etapa 1: criação de uma VPC com sub-redes públicas e privadas
- Etapa 2: criação de um grupo de segurança da VPC para uma aplicação pública
- Etapa 3: criação de um grupo de segurança da VPC para um cluster privado
- Etapa 4: criar um grupo de sub-redes
- Exclusão de uma VPC

Seu cluster de banco de dados só precisa estar disponível para sua aplicação, e não para a Internet pública. Assim, você cria uma VPC com sub-redes públicas e privadas. A aplicação está hospedada

na sub-rede pública, para que ela possa chegar à Internet pública. O cluster é hospedado em uma sub-rede privada. A aplicação pode se conectar ao cluster porque ela está hospedada na mesma VPC. Mas o cluster não está disponível para a Internet pública, o que oferece maior segurança.

O procedimento neste tópico configura uma sub-rede pública e privada adicional em uma zona de disponibilidade separada. Essas sub-redes não são usadas no procedimento. Um grupo de sub-redes do DocumentDB exige uma sub-rede em, pelo menos, duas zonas de disponibilidade. A sub-rede adicional facilita a configuração de mais de uma instância de DocumentDB.

Este tópico descreve como configurar uma VPC para os clusters do Amazon DocumentDB. Para obter mais informações sobre o Amazon VPC, consulte o Guia do usuário da Amazon VPC.

> **Tip**
>
> É possível configurar a conectividade de rede entre uma instância do Amazon EC2 e um cluster do DocumentDB automaticamente ao criar o cluster. A configuração da rede é semelhante à descrita neste cenário. Para obter mais informações, consulte Conectar o Amazon EC2 automaticamente.

Etapa 1: criação de uma VPC com sub-redes públicas e privadas

Use o seguinte procedimento para criar uma VPC com sub-redes públicas e privadas.

Para criar uma VPC e sub-redes

1. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc.
2. No canto superior direito do Console de gerenciamento da AWS, escolha a região na qual criar sua VPC. Este exemplo usa a região Oeste dos EUA (Oregon).
3. No canto superior esquerdo, escolha VPC dashboard (Painel da VPC). Para começar a criar uma VPC, escolha Create VPC (Criar VPC).
4. Para Resources to create (Recursos a serem criados) em VPC settings (Configurações da VPC), escolha VPC and more (VPC e mais).
5. Para VPC settings (Configurações da VPC), defina os seguintes valores:
   - Geração automática de etiquetas de nome — **`example`**
   - Bloco CIDR IPv4 — **`10.0.0.0/16`**

- Bloco CIDR IPv6: Nenhum bloco CIDR IPv6
- Locação: Padrão
- Número de zonas de disponibilidade (AZs) — 2
- Personalizar AZs: mantenha os valores padrão.
- Número de sub-redes públicas — 2
- Número de sub-redes privadas — 2
- Personalizar blocos CIDR de sub-redes: mantenha os valores padrão.
- Gateways de NAT ($): Nenhum
- Endpoints da VPC: Nenhum
- Opções de DNS: mantenha os valores padrão.

6. Escolha Criar VPC.

Etapa 2: criação de um grupo de segurança da VPC para uma aplicação pública

Em seguida, crie um grupo de segurança para acesso público. Para se conectar a instâncias públicas do EC2 em sua VPC, adicione regras de entrada ao seu grupo de segurança de VPC. Isso permite que o tráfego se conecte pela Internet.

Para criar um grupo de segurança da VPC

1. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc.
2. Escolha VPC Dashboard (Painel da VPC), Security Groups (Grupos de segurança) e depois Create grupo de segurança (Criar grupo de segurança).
3. Na página Create grupo de segurança (Criar grupo de segurança), defina esses valores:
   - Nome do grupo de segurança — **`example-securitygroup`**
   - Descrição — **`Application security group`**
   - VPC: escolha a VPC que você criou anteriormente, por exemplo: vpc-example.
4. Adicione regras de entrada ao grupo de segurança.
   a. Determine o endereço IP a ser usado para se conectar a instâncias do EC2 em sua VPC usando Secure Shell (SSH). Para determinar seu endereço IP público, em uma janela ou guia diferente do navegador, você pode usar o serviço em https://checkip.amazonaws.com. Um exemplo de endereço IP: `203.0.113.25/32`.

Em muitos casos, você pode se conectar por meio de um provedor de serviços de Internet (ISP) ou atrás de um firewall sem um endereço IP estático. Se sim, especifique o intervalo de endereços IP utilizado por computadores cliente.

> **Warning**
>
> Se usar `0.0.0.0/0` para acesso do SSH, você possibilitará que todos os endereços IP acessem suas instâncias públicas usando SSH. Essa abordagem é aceitável por um período curto em um ambiente de teste, mas não é seguro em ambientes de produção. Em produção, autorize somente um endereço IP específico ou um intervalo de endereços para acessar suas instâncias usando SSH.

b. Na seção Regras de entrada, escolha Adicionar regra.

c. Defina os valores a seguir para a sua nova regra de entrada, para possibilitar o acesso do SSH à sua instância do Amazon EC2. Depois de fazer isso, será possível se conectar à sua instância do EC2 para instalar a aplicação e outros utilitários. Você também se conecta à sua instância do EC2 para fazer upload de conteúdo para sua aplicação.

- Tipo — **SSH**
- Origem: o endereço IP ou o intervalo da Etapa a, por exemplo: **203.0.113.25/32**.

d. Escolha Adicionar regra.

e. Defina os valores a seguir para a sua nova regra de entrada a fim de permitir o acesso de HTTP à sua aplicação.

- Tipo — **HTTP**
- Fonte — **0.0.0.0/0**

5. Escolha Create grupo de segurança (Criar grupo de segurança) para criar o grupo de segurança.

   Anote o ID do grupo de segurança porque você precisará dele posteriormente em outro procedimento.

Etapa 3: criação de um grupo de segurança da VPC para um cluster privado

Para manter seu cluster privado, crie um segundo grupo de segurança para acesso privado. Para se conectar a clusters privados de banco de dados na sua VPC, você adiciona regras de entrada ao grupo de segurança da VPC que permitem o tráfego somente a partir da sua aplicação.

Para criar um grupo de segurança da VPC

1. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc.
2. Escolha VPC Dashboard (Painel da VPC), Security Groups (Grupos de segurança) e depois Create grupo de segurança (Criar grupo de segurança).
3. Na página Create grupo de segurança (Criar grupo de segurança), defina esses valores:
   - Nome do grupo de segurança — **`example-securitygroup`**
   - Descrição — **`Instance security group`**
   - VPC: escolha a VPC que você criou anteriormente, por exemplo: vpc-example
4. Adicione regras de entrada ao grupo de segurança.
   a. Na seção Regras de entrada, escolha Adicionar regra.
   b. Defina os valores a seguir para a sua nova regra de entrada, para permitir o tráfego do DocumentDB na porta 27017 de sua instância do Amazon EC2. Depois de fazer isso, será possível se conectar da sua aplicação ao seu cluster. Ao fazer isso, será possível armazenar e recuperar dados de sua aplicação para seu banco de dados.
      - Tipo — **`Custom TCP`**
      - Origem: o identificador do grupo de segurança da aplicação criado anteriormente neste tópico, por exemplo: sg-9edd5cfb.
   c. Escolha Adicionar regra.
   d. Defina os valores a seguir para a sua nova regra de entrada a fim de permitir o acesso de HTTP à sua aplicação.
      - Tipo — **`HTTP`**
      - Fonte — **`0.0.0.0/0`**
5. Escolha Create grupo de segurança (Criar grupo de segurança) para criar o grupo de segurança.

Etapa 4: criar um grupo de sub-redes

Um grupo de sub-redes é uma coleção de sub-redes que você cria em uma VPC e designa para seus clusters. Um grupo de sub-redes permite que você especifique uma VPC ao criar clusters.

Para criar um grupo de sub-redes

1. Identifique as sub-redes privadas do seu banco de dados na VPC.
   a. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc.
   b. Escolha VPC Dashboard (Painel da VPC), depois selecione Subnets (Sub-redes).
   c. Anote os IDs das sub-redes que você criou na Etapa 1 chamadas, por exemplo: example-subnet-private1-us-west-2a e example-subnet-private2-us-west-2b. Você precisará dos IDs de sub-rede ao criar seu grupo de sub-redes.
2. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.

   Conecte-se ao console do Amazon DocumentDB, não ao console da Amazon VPC.
3. No painel de navegação, escolha Grupos de sub-redes.
4. Escolha Criar.
5. Na página Criar grupo de sub-redes, defina esses valores na seção Detalhes do grupo de sub-redes:
   - Nome — **`example-db-subnet-group`**
   - Descrição — **`Instance security group`**
6. Na seção Adicionar sub-redes, defina estes valores:
   - VPC: escolha a VPC que você criou anteriormente, por exemplo: vpc-example
   - Zonas de disponibilidade: selecione as duas zonas de disponibilidade criadas na Etapa 1. Exemplo: us-west-2a e us-west-2b
   - Sub-redes: escolha as sub-redes privadas criadas na Etapa 1.
7. Escolha Criar.

Seu novo grupo de sub-redes será exibido na lista de grupos de sub-redes no console do DocumentDB. É possível selecionar o grupo de sub-redes para visualizar detalhes no painel de detalhes. Esses detalhes incluem todas as sub-redes associadas ao grupo.

> **Note**
>
> Se você criou essa VPC para associá-la a um cluster do DocumentDB, crie o cluster seguindo as instruções em Criar um cluster do Amazon DocumentDB.

Exclusão de uma VPC

É possível excluir uma VPC e os outros recursos usados nela não forem mais necessários.

> **Note**
>
> Se você incluiu recursos na VPC que criou neste tópico, talvez seja necessário excluí-los para poder excluir a VPC. Por exemplo, esses recursos podem incluir instâncias do Amazon EC2 ou clusters do DocumentDB. Para obter mais informações, consulte Como excluir a sua VPC no Guia do usuário da Amazon VPC.

Para excluir uma VPC e recursos relacionados

1. Exclua o grupo de sub-redes:
   a. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
   b. No painel de navegação, escolha Grupos de sub-redes.
   c. Selecione o grupo de sub-redes que você deseja excluir, como, por exemplo, example-db-subnet-group.
   d. Escolha Delete (Excluir) e, em seguida, Delete (Excluir) na janela de confirmação.
2. Anote o ID da VPC:
   a. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc.
   b. Escolha Painel da VPC e, em seguida, Suas VPCs.
   c. Na lista, identifique a VPC que você criou, como vpc-example.
   d. Anote o VPC ID (ID da VPC) da VPC que você criou. Você precisará do ID da VPC nas etapas posteriores.
3. Exclua os grupos de segurança:

a. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc.

b. Escolha VPC Dashboard (Painel da VPC) e, em seguida, Security Groups (Grupos de segurança).

c. Selecione o grupo de segurança para o cluster do Amazon DocumentDB, como example-securitygroup.

d. Em Ações, escolha Excluir grupos de segurança e, depois, Excluir na caixa de diálogo de confirmação.

e. Na página Grupos de segurança, selecione o grupo de segurança para a instância do Amazon EC2, como, por exemplo, example-securitygroup.

f. Em Ações, escolha Excluir grupos de segurança e, depois, Excluir na caixa de diálogo de confirmação.

4. Exclua a VPC:

a. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc.

b. Escolha Painel da VPC e, em seguida, Suas VPCs.

c. Selecione a VPC que deseja excluir, como a vpc-example.

d. Em Ações, escolha Excluir VPC.

A página de confirmação mostra outros recursos associados à VPC que também serão excluídos, incluindo as sub-redes associadas a ela.

e. Na caixa de diálogo de confirmação, insira **`delete`** e, em seguida, escolha Excluir.

Criação de uma VPC de pilha dupla para uso com um cluster do DocumentDB

Um cenário comum inclui um cluster em uma nuvem privada virtual (VPC) com base no serviço Amazon VPC. Essa VPC compartilha dados com uma instância pública do Amazon EC2 que está sendo executada na mesma VPC. Neste tópico, você criará a VPC para esse cenário.

Tópicos

- Etapa 1: criação de uma VPC com sub-redes públicas e privadas
- Etapa 2: criação de um grupo de segurança da VPC para uma instância pública do Amazon EC2
- Etapa 3: criação de um grupo de segurança da VPC para um cluster privado

- Etapa 4: criar um grupo de sub-redes
- Etapa 5: criação de uma instância do Amazon EC2 no modo de pilha dupla
- Etapa 6: criação de um cluster no modo de pilha dupla
- Etapa 7: conexão com sua instância do Amazon EC2 e cluster de banco de dados
- Exclusão da VPC

Neste procedimento, você cria a VPC para esse cenário que funciona com um banco de dados em execução no modo de pilha dupla. Dual-stack o modo permite a conexão pelo protocolo de endereçamento IPv6. Para obter mais informações sobre endereçamento IP, consulte Endereçamento IP do Amazon DocumentDB.

Dual-stack clusters de rede são suportados na maioria das regiões. Para obter mais informações, consulte Dual-stack modo Disponibilidade de região e versão. Para ver as limitações do modo de pilha dupla, consulte Limitações para clusters de rede de pilha dupla.

Esse tópico e o IPv4-only tópico criam as sub-redes públicas e privadas na mesma VPC. Para obter informações sobre como criar o cluster do Amazon DocumentDB em uma VPC e a instância do Amazon EC2 em uma VPC diferente, consulte Acesso a um cluster do Amazon DocumentDB em uma VPC.

Seu cluster do DocumentDB precisa estar disponível para sua instância do Amazon EC2, e não para a Internet pública. Assim, você cria uma VPC com sub-redes públicas e privadas. A instância do EC2 é hospedada na sub-rede pública, para que ela possa acessar a Internet pública. O cluster é hospedado em uma sub-rede privada. A instância do EC2 pode se conectar ao cluster de banco de dados porque ela está hospedada na mesma VPC. No entanto, o cluster não está disponível para a Internet pública, o que oferece maior segurança.

Os procedimentos neste tópico configuram uma sub-rede pública e privada adicional em uma zona de disponibilidade separada. Essas sub-redes não são usadas no procedimento. Um grupo de sub-redes do DocumentDB exige uma sub-rede em, pelo menos, duas zonas de disponibilidade. A sub-rede adicional facilita a configuração de mais de uma instância de DocumentDB.

Para criar um cluster que usa o modo de pilha dupla, especifique Dual-stack mod e para a configuração do tipo de rede. Você também pode modificar um cluster com a mesma configuração. Para obter mais informações sobre como criar um cluster, consulte Criar um cluster do Amazon DocumentDB. Para ter mais informações sobre como modificar um cluster de banco de dados, consulte Modificar um cluster do Amazon DocumentDB.

Este tópico descreve como configurar uma VPC para os clusters do Amazon DocumentDB. Para obter mais informações sobre o Amazon VPC, consulte o Guia do usuário da Amazon VPC.

Etapa 1: criação de uma VPC com sub-redes públicas e privadas

Use o seguinte procedimento para criar uma VPC com sub-redes públicas e privadas.

Para criar uma VPC e sub-redes

1. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc.
2. No canto superior direito do Console de gerenciamento da AWS, escolha a região na qual criar sua VPC. Este exemplo usa a região Oeste dos EUA (Oregon).
3. No canto superior esquerdo, escolha VPC dashboard (Painel da VPC). Para começar a criar uma VPC, escolha Create VPC (Criar VPC).
4. Para Resources to create (Recursos a serem criados) em VPC settings (Configurações da VPC), escolha VPC and more (VPC e mais).
5. Para VPC settings (Configurações da VPC), defina os seguintes valores:
   - Geração automática de etiquetas de nome — **`example-dual-stack`**
   - Bloco CIDR IPv4 — **`10.0.0.0/16`**
   - Bloco CIDR IPv6 — Amazon-provided bloco CIDR IPv6
   - Locação: Padrão
   - Número de zonas de disponibilidade (AZs) — 2
   - Personalizar AZs: mantenha os valores padrão.
   - Número de sub-redes públicas — 2
   - Número de sub-redes privadas — 2
   - Personalizar blocos CIDR de sub-redes: mantenha os valores padrão.
   - Gateways de NAT ($): Nenhum
   - Gateway da Internet somente de saída: Não
   - Endpoints da VPC: Nenhum
   - Opções de DNS: mantenha os valores padrão.
6. Escolha Criar VPC.

Etapa 2: criação de um grupo de segurança da VPC para uma instância pública do Amazon EC2

Em seguida, você criará um grupo de segurança para acesso público. Para se conectar a instâncias públicas do EC2 na sua VPC, adicione regras de entrada ao grupo de segurança de sua VPC que permitam que o tráfego se conecte da Internet.

Para criar um grupo de segurança da VPC

1. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc.
2. Escolha VPC Dashboard (Painel da VPC), Security Groups (Grupos de segurança) e depois Create grupo de segurança (Criar grupo de segurança).
3. Na página Create grupo de segurança (Criar grupo de segurança), defina esses valores:
   - Nome do grupo de segurança — **`example-dual-stack-securitygroup`**
   - Descrição — **`Dual-stack security group`**
   - VPC: escolha a VPC que você criou anteriormente, por exemplo: vpc-example-dual-stack.
4. Adicione regras de entrada ao grupo de segurança.
   a. Determine o endereço IP a ser usado para se conectar a instâncias do EC2 em sua VPC usando Secure Shell (SSH). Para determinar seu endereço IP público, em uma janela ou guia diferente do navegador, você pode usar o serviço em https://checkip.amazonaws.com.

      Um exemplo de um intervalo de endereços do Protocolo de Internet versão 4 (IPv4) é. `203.0.113.25/32` Um exemplo de intervalo de endereços de Protocolo de Internet versão 6 (IPv6) é `2001:db8:1234:1a00::/64`.

      Em muitos casos, você pode se conectar por meio de um provedor de serviços de Internet (ISP) ou atrás de um firewall sem um endereço IP estático. Se sim, especifique o intervalo de endereços IP utilizado por computadores cliente.

      > ⚠ Warning
      >
      > Se usar `0.0.0.0/0` para IPv4 ou `::0` para IPv6, você possibilitará que todos os endereços IP acessem suas instâncias públicas usando SSH. Essa abordagem é aceitável por um período curto em um ambiente de teste, mas não é seguro em

ambientes de produção. No ambiente de produção, autorize apenas um endereço IP específico ou um intervalo de endereços a acessar as instâncias.

b. Na seção Regras de entrada, escolha Adicionar regra.

c. Defina os valores a seguir para a sua nova regra de entrada, para possibilitar o acesso do SSH à sua instância do Amazon EC2. Depois de fazer isso, será possível se conectar à sua instância do EC2 para instalar aplicações e outros utilitários. Especifique um endereço IP para que você possa acessar sua instância do EC2:

- Tipo — **SSH**
- Origem: o endereço IP ou o intervalo que você criou na Etapa a. Um exemplo de intervalo de endereços IPv4 é **203.0.113.25/3**2. Um exemplo de intervalo de endereços IPv6 é **2001:DB8::/32**.

d. Escolha Adicionar regra.

5. Escolha Create grupo de segurança (Criar grupo de segurança) para criar o grupo de segurança.

   Anote o ID do grupo de segurança porque você precisará dele posteriormente em outro procedimento.

Etapa 3: criação de um grupo de segurança da VPC para um cluster privado

Para manter seu cluster privado, crie um segundo grupo de segurança para acesso privado. Para se conectar a clusters privados em sua VPC, adicione regras de entrada ao seu grupo de segurança de VPC. Eles permitem o tráfego somente de sua instância do Amazon EC2.

Para criar um grupo de segurança da VPC

1. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc.
2. Escolha VPC Dashboard (Painel da VPC), Security Groups (Grupos de segurança) e depois Create grupo de segurança (Criar grupo de segurança).
3. Na página Create grupo de segurança (Criar grupo de segurança), defina esses valores:

- Nome do grupo de segurança — **example-dual-stack-cluster-securitygroup**
- Descrição — **Dual-stack cluster security group**
- VPC: escolha a VPC que você criou anteriormente, por exemplo: vpc-example-dual-stack

4. Adicione regras de entrada ao grupo de segurança.

   a. Na seção Regras de entrada, escolha Adicionar regra.

   b. Defina os valores a seguir para a sua nova regra de entrada, para permitir o tráfego do DocumentDB na porta 27017 de sua instância do Amazon EC2. Depois de fazer isso, será possível se conectar da sua instância do EC2 ao seu cluster. Ao fazer isso, será possível enviar dados da instância do EC2 para o seu banco de dados.

      - Tipo — **`Custom TCP`**
      - Origem: o identificador do grupo de segurança do EC2 criado anteriormente neste tópico, por exemplo: sg-9edd5cfb.

5. Escolha Create grupo de segurança (Criar grupo de segurança) para criar o grupo de segurança.

Etapa 4: criar um grupo de sub-redes

Um grupo de sub-redes é uma coleção de sub-redes que você cria em uma VPC e designa para seus clusters. Ao usar um grupo de sub-redes, é possível especificar uma VPC específica ao criar clusters. Para criar um grupo de sub-redes que seja compatível com `DUAL`, todas as sub-redes devem ser compatíveis com `DUAL`. Para ser compatível com `DUAL`, uma sub-rede deve ter um CIDR IPv6 associado a ela.

Para criar um grupo de sub-redes

1. Identifique as sub-redes privadas do seu banco de dados na VPC.

   a. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc.

   b. Escolha VPC Dashboard (Painel da VPC), depois selecione Subnets (Sub-redes).

   c. Anote os IDs das sub-redes que você criou na Etapa 1 chamadas, por exemplo: example-dual-stack-subnet-private1-us-west-2a e example-dual-stack-subnet-private2-us-west-2b. Você precisará dos IDs de sub-rede ao criar seu grupo de sub-redes.

2. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.

   Conecte-se ao console do Amazon DocumentDB, não ao console da Amazon VPC.

3. No painel de navegação, escolha Grupos de sub-redes.

4. Escolha Criar.

5. Na página Criar grupo de sub-redes, defina esses valores na seção Detalhes do grupo de sub-redes:
   - Nome — **`example-dual-stack-cluster-subnet-group`**
   - Descrição — **`Dual-stack cluster subnet group`**
6. Na seção Adicionar sub-redes, defina estes valores:
   - VPC: escolha a VPC que você criou anteriormente, por exemplo: vpc-example-dual-stack
   - Zonas de disponibilidade: selecione as duas zonas de disponibilidade criadas na Etapa 1. Exemplo: us-west-2a e us-west-2b
   - Sub-redes: escolha as sub-redes privadas criadas na Etapa 1.
7. Escolha Criar.

Seu novo grupo de sub-redes será exibido na lista de grupos de sub-redes no console do DocumentDB. É possível selecionar o grupo de sub-redes para visualizar detalhes no painel de detalhes. Esses detalhes incluem todas as sub-redes associadas ao grupo.

Etapa 5: criação de uma instância do Amazon EC2 no modo de pilha dupla

Para criar uma instância do Amazon EC2, siga as instruções em Execução de uma instância do EC2 usando o assistente de inicialização de instâncias no console, no Guia do usuário do Amazon Elastic Compute Cloud.

Na página Configure Instance Details (Configurar os detalhes da instância), defina esses valores e mantenha os outros valores como padrão:

- Rede: escolha uma VPC existente com sub-redes públicas e privadas, como a vpc-example-dual-stack-vpc (vpc-identifier), criada em Etapa 1: criação de uma VPC com sub-redes públicas e privadas.
- Sub-rede: escolha uma sub-rede pública existente, como subnet-identifier | example-dual-stack-subnet-public1-us-east-2a | us-east-2a, criada em Etapa 2: criação de um grupo de segurança da VPC para uma instância pública do Amazon EC2.
- Auto-assign IP público — Escolha Ativar.
- Auto-assign IP IPv6 — Escolha Ativar.
- Firewall (grupos de segurança): escolha Selecionar um grupo de segurança existente.

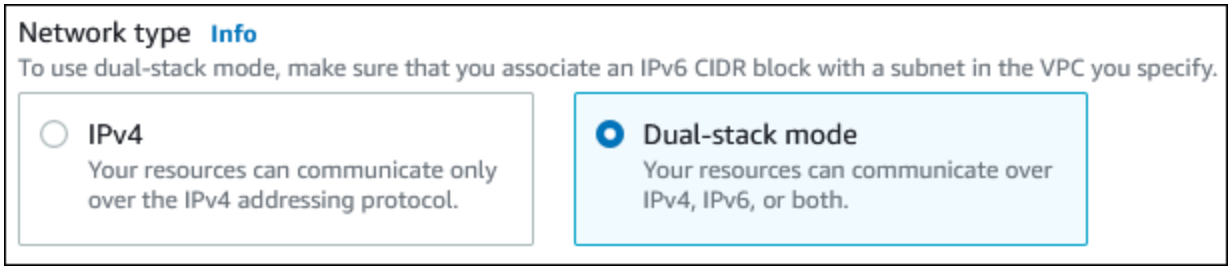
- Grupos de segurança comuns: escolha um grupo de segurança existente, como o example-dual-stack-securitygroup criado em Etapa 2: criação de um grupo de segurança da VPC para uma instância pública do Amazon EC2. Verifique se o grupo de segurança escolhido inclui regras de entrada para Secure Shell (SSH) e acesso HTTP.

Etapa 6: criação de um cluster no modo de pilha dupla

Nesta etapa, você criará um cluster de banco de dados para execução no modo de pilha dupla. !!! Observação: esta seção precisará ser editada depois que as atualizações do IPv6 forem feitas no console !!!

Para criar um cluster no modo de pilha dupla

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No canto superior direito do console, escolha Região da AWS onde você deseja criar o cluster DocumentDB. Este exemplo usa a região Leste dos EUA (Ohio).
3. No painel de navegação, escolha Clusters.
4. Na página de lista de Clusters, escolha Criar.
5. Na página Create Amazon DocumentDB cluster, certifique-se de que a opção Instance-based cluster esteja escolhida.
6. Na seção Conectividade, em Tipo de rede, escolha o Dual-stack modo.



7. Na parte inferior da página, ative Mostrar configurações avançadas.
8. Na seção Configurações de rede, defina estes valores:

   - Nuvem privada virtual (VPC): escolha uma VPC existente com sub-redes públicas e privadas, como a vpc-example-dual-stack (vpc-identifier) criada em Etapa 1: criação de uma VPC com sub-redes públicas e privadas.

     A VPC deve ter sub-redes em zonas de disponibilidade diferentes.

- Grupo de sub-redes: escolha um grupo de sub-redes para a VPC, como example-dual-stack-cluster-subnet-group criado em Etapa 4: criar um grupo de sub-redes.
- Acesso público: escolha Não.
- Grupo de segurança da VPC (firewall): selecione Escolher existente.
- Grupos de segurança da VPC existentes: escolha um grupo de segurança da VPC existente configurado para acesso privado, como o example-dual-stack-cluster-securitygroup criado em Etapa 3: criação de um grupo de segurança da VPC para um cluster privado.

  Remova outros grupos de segurança, como o grupo de segurança padrão, escolhendo o X associado.
- Zona de disponibilidade: selecione a zona de disponibilidade criada na Etapa 1. Exemplo: us-west-2a.

  Para evitar o tráfego entre zonas, certifique-se de que o cluster e a instância do EC2 estejam na mesma zona de disponibilidade.

9. Nas seções restantes, especifique suas configurações de cluster. Para obter informações sobre cada configuração, consulte Criar um cluster do Amazon DocumentDB.

Etapa 7: conexão com sua instância do Amazon EC2 e cluster de banco de dados

Depois de criar sua instância do Amazon EC2 e o cluster do DocumentDB no modo de pilha dupla, será possível se conectar a cada um usando o protocolo IPv6. Para se conectar a uma instância do EC2 usando o protocolo IPv6, siga as instruções em Conexão com sua instância do EC2 no Guia do usuário do Amazon Elastic Compute Cloud.

Para se conectar ao seu cluster do DocumentDB a partir da instância do EC2, siga as instruções em Etapa 5: instalação do shell do MongoDB, no tópico Conexão manual ao Amazon EC2 (e prossiga com as etapas 6 e 7 subsequentes no mesmo procedimento).

Exclusão da VPC

É possível excluir uma VPC e os outros recursos usados nela não forem mais necessários.

> **Note**
>
> Se você incluiu recursos na VPC que criou neste tópico, talvez seja necessário excluí-los para poder excluir a VPC. Por exemplo, esses recursos podem incluir instâncias do Amazon

EC2 ou clusters do DocumentDB. Para obter mais informações, consulte Como excluir a sua VPC no Guia do usuário da Amazon VPC.

Para excluir uma VPC e recursos relacionados

1. Exclua o grupo de sub-redes:
   a. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
   b. No painel de navegação, escolha Grupos de sub-redes.
   c. Selecione o grupo de sub-redes que deseja excluir, como, por exemplo, example-dual-stack-cluster-subnet-group.
   d. Escolha Delete (Excluir) e, em seguida, Delete (Excluir) na janela de confirmação.
2. Anote o ID da VPC:
   a. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc.
   b. Escolha Painel da VPC e, em seguida, Suas VPCs.
   c. Na lista, identifique a VPC que você criou, como vpc-example-dual-stack.
   d. Anote o VPC ID (ID da VPC) da VPC que você criou. Você precisará do ID da VPC nas etapas posteriores.
3. Exclua os grupos de segurança:
   a. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc.
   b. Escolha VPC Dashboard (Painel da VPC) e, em seguida, Security Groups (Grupos de segurança).
   c. Selecione o grupo de segurança para o cluster do Amazon DocumentDB, como example-dual-stack-securitygroup.
   d. Em Ações, escolha Excluir grupos de segurança e, depois, Excluir na caixa de diálogo de confirmação.
   e. Na página Grupos de segurança, selecione o grupo de segurança para a instância do Amazon EC2, como, por exemplo, example-securitygroup.
   f. Em Ações, escolha Excluir grupos de segurança e, depois, Excluir na caixa de diálogo de confirmação.
4. Exclua o gateway NAT:

a. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc.
b. Escolha VPC Dashboard (Painel da VPC) e, em seguida, Security Groups (Grupos de segurança).
c. Selecione o gateway NAT da VPC que você criou. Use o ID da VPC para identificar o gateway NAT correto.
d. Em Actions (Ações), escolha Delete NAT gateway (Excluir gateway NAT).
e. Na caixa de diálogo de confirmação, insira **`delete`** e, em seguida, escolha Excluir.

5. Exclua a VPC:

a. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc.
b. Escolha Painel da VPC e, em seguida, Suas VPCs.
c. Selecione a VPC que deseja excluir, como a vpc-example-dual-stack.
d. Em Ações, escolha Excluir VPC.

   A página de confirmação mostra outros recursos associados à VPC que também serão excluídos, incluindo as sub-redes associadas a ela.
e. Na caixa de diálogo de confirmação, insira **`delete`** e, em seguida, escolha Excluir.

6. Libere os endereços IP elásticos.

a. Abra o console do EC2 em https://console.aws.amazon.com/ec2.
b. Escolha Painel do EC2 e, em seguida, IPs elásticos.
c. Selecione o endereço de IP elástico que deseja liberar.
d. Em Actions (Ações), escolha Release Elastic IP addresses (Liberar endereços de IP elásticos).
e. Na caixa de diálogo de confirmação, escolha Liberar.

Backup e restauração no Amazon DocumentDB

O Amazon DocumentDB (compatível com MongoDB) O faz backup contínuo de dados no Amazon Simple Storage Service (Amazon S3) por 1 a 35 dias para que você possa restaurar rapidamente de qualquer ponto dentro do período de retenção de backup. O Amazon DocumentDB também tira snapshots automáticos dos seus dados como parte desse processo de backup contínuo.

> **Note**
>
> Esses são buckets do Amazon S3 gerenciados por serviços e você não terá acesso aos arquivos de backup. Para controlar seus próprios backups, consulteDespejo, restauração, importação e exportação de dados.

Você também pode reter os dados de backup além do período de retenção de backup, criando um snapshot manual dos dados do cluster. O processo de backup não afeta o desempenho do cluster.

Esta seção aborda os casos de uso para os recursos de backup no Amazon DocumentDB e mostra como gerenciar backups para os clusters do Amazon DocumentDB.

Tópicos

- Backup e restauração: conceitos
- Noções básicas do uso do armazenamento de backup do
- Despejo, restauração, importação e exportação de dados
- Considerações sobre snapshot de cluster
- Comparação dos snapshots automáticos e manuais
- Criação de um snapshot manual de cluster
- Cópia de snapshots de cluster do Amazon DocumentDB
- Compartilhamento de snapshots de cluster do Amazon DocumentDB
- Restauração de um snapshot de cluster
- Restauração para um ponto no tempo
- Exclusão de um snapshot de cluster

Backup e restauração: conceitos

Substantivo	Description	APIs (verbos)

Período de retenção de backup	Um período entre 1 e 35 dias no qual é possível executar uma restauração point-in-time.	`create-db-cluster`
		`modify-db-cluster`
		`restore-db-cluster-to-point-in-time`

Volume de armazenamento do Amazon DocumentDB	O volume de armazenamento resiliente e de alta disponibilidade que replica dados de seis maneiras em três zonas de disponibilidade. Um cluster do Amazon DocumentDB é	`create-db-cluster` `delete-db-cluster`

Substantivo	Description	APIs (verbos)
	resiliente, independentemente do número de instâncias no cluster.	

Janela de backup	Período no dia em que os snapshots automáticos são obtidos.	`create-db-cluster`
		`describe-db-cluster`
		`modify-db-cluster`

Snapshot automático	Snapshots diários que são backups completos do cluster e são criados automaticamente pelo processo de backup contínuo no Amazon DocumentDB.	`restore-db-cluster-from-snapshot`
		`describe-db-cluster-snapshot-attributes`
		`describe-db-cluster-snapshots`

Substantivo	Description	APIs (verbos)

Snapshot manual	Os snapshots que você cria manualmente para reter backups completos de um cluster além do período de backup.	`create-db-cluster-snapshot`
		`copy-db-cluster-snapshot`
		`delete-db-cluster-snapshot`
		`describe-db-cluster-snapshot-attributes`
		`describe-db-cluster-snapshots`
		`modify-db-cluster-snapshot-attribute`

Noções básicas do uso do armazenamento de backup do

O armazenamento de backup do Amazon DocumentDB consiste em backups contínuos dentro do período de retenção de backup e snapshots manuais fora do período de retenção. Para controlar o uso de armazenamento de backup, reduza o intervalo de retenção de backup, remova snapshots manuais antigos quando eles não forem mais necessários, ou ambos. Para obter informações gerais sobre backups do Amazon DocumentDB, consulte Backup e restauração no Amazon DocumentDB. Para obter informações sobre a definição de preço do armazenamento de backup do Amazon DocumentDB, consulte a página da web Definição de preço do Amazon DocumentDB.

Para controlar os custos, monitore a quantidade de armazenamento consumido por backups contínuos e snapshots manuais que persistem além do período de retenção. Em seguida, é possível reduzir o intervalo de retenção de backup e remover snapshots manuais quando eles não forem mais necessários.

Você pode usar as CloudWatch métricas `TotalBackupStorageBilled` da Amazon e revisar e monitorar `BackupRetentionPeriodStorageUsed` a quantidade de armazenamento usada pelos seus backups do Amazon DocumentDB, da seguinte forma: `SnapshotStorageUsed`

- `BackupRetentionPeriodStorageUsed` representa a quantidade de armazenamento de backup usado para armazenar backups contínuos na hora atual. Esse valor da métrica depende do tamanho do volume do cluster e do número de alterações feitas durante o período de retenção. No entanto, para fins de faturamento a métrica não excede o tamanho cumulativo do volume do cluster durante o período de retenção. Por exemplo, se o tamanho do cluster for 100 GiB e o período de retenção for de dois dias, o valor máximo de `BackRetentionPeriodStorageUsed` será 200 GiB (100 GiB + 100 GiB).

- `SnapshotStorageUsed` representa a quantidade de armazenamento de backup usado para armazenar snapshots manuais além do período de retenção de backup. Os snapshots manuais tirados dentro do período de retenção não são contados no armazenamento de backup. Da mesma forma, snapshots automáticos não são contados no armazenamento de backup. O tamanho de cada snapshot é o tamanho do volume do cluster no momento em que você faz o snapshot. O valor `SnapshotStorageUsed` depende do número de snapshots que você mantém e do tamanho de cada snapshot. Por exemplo, suponha que você tenha um snapshot fora do período de retenção e o tamanho do volume de cluster tenha sido 100 GiB quando esse snapshot foi feito. A quantidade de `SnapshotStorageUsed` será 100 GiB.

- `TotalBackupStorageBilled` representa a soma de `BackupRetentionPeriodStorageUsed` e `SnapshotStorageUsed`, menos uma quantidade de armazenamento de backup grátis igual ao tamanho do volume do cluster para um dia. Por exemplo, se o tamanho do cluster for 100 GiB, e você tiver um dia de retenção e um snapshot fora do período de retenção, o `TotalBackupStorageBilled` será 100 GiB (100 GiB + 100 GiB - 100 GiB).

- Essas métricas são calculadas de maneira independente para cada cluster do Amazon DocumentDB.

Você pode monitorar seus clusters do Amazon DocumentDB e criar relatórios usando CloudWatch métricas por meio do CloudWatch console. Para obter mais informações sobre como usar CloudWatch métricas, consulteMonitoramento e registro no Amazon DocumentDB.

Despejo, restauração, importação e exportação de dados

É possível usar os utilitários `mongodump`, `mongorestore`, `mongoexport` e `mongoimport` para mover dados para dentro e para fora do cluster do Amazon DocumentDB. Esta seção discute o propósito de cada uma dessas ferramentas e configurações para ajudar você a obter um melhor desempenho.

> **Requisito de versão do MongoDB Database Tools**
>
> Use as ferramentas de banco de dados do MongoDB até a versão 100.11.0, inclusive, para o Amazon DocumentDB. Para fazer o download, consulte os lançamentos do MongoDB Database Tools no site do MongoDB.

Tópicos

- mongodump
- mongorestore
- mongoexport
- importação de mongo
- Tutorial

`mongodump`

O utilitário `mongodump` cria um backup binário (BSON) de um banco de dados MongoDB. A ferramenta `mongodump` é o método preferencial de despejo de dados da implantação MongoDB de origem visando restaurá-los no cluster do Amazon DocumentDB devido às eficiências de tamanho obtidas armazenando os dados em um formato binário.

Dependendo dos recursos disponíveis na instância ou máquina que você está usando para executar o comando, é possível acelerar seu `mongodump` aumentando o número de conexões paralelas despejadas do padrão 1 usando a opção `--numParallelCollections`. Uma boa regra geral é começar com um operador por vCPU na instância principal do cluster do Amazon DocumentDB.

Exemplo de uso

Veja a seguir um exemplo de uso do utilitário `mongodump` no cluster do 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

O utilitário `mongorestore` permite restaurar um backup binário (BSON) de um banco de dados criado com o utilitário `mongodump`. É possível melhorar o desempenho da restauração aumentando o número de operadores para cada coleção durante a restauração com a opção `--numInsertionWorkersPerCollection` (o padrão é 1). Uma boa regra geral é começar com um operador por vCPU na instância principal do cluster do Amazon DocumentDB.

Exemplo de uso

Veja a seguir um exemplo de uso do utilitário `mongorestore` no cluster do 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

A ferramenta `mongoexport` exporta dados no Amazon DocumentDB para os formatos de arquivo JSON, CSV ou TSV. A ferramenta `mongoexport` é o método preferencial para exportar dados que precisam ser legíveis por humanos ou por máquina.

> **Note**
>
> `mongoexport` não oferece suporte diretamente a exportações paralelas. No entanto, é possível aumentar o desempenho executando vários trabalhos `mongoexport` simultaneamente para coleções diferentes.

Exemplo de uso

Veja a seguir um exemplo de uso da ferramenta `mongoexport` no cluster do 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
```

importação de mongo

A ferramenta `mongoimport` importa o conteúdo de arquivos JSON, CSV ou TSV para um cluster do Amazon DocumentDB. É possível usar o parâmetro `--numInsertionWorkers` para paralelizar e acelerar a importação (o padrão é 1).

Exemplo de uso

Veja a seguir um exemplo de uso da ferramenta `mongoimport` no cluster do 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

O tutorial a seguir descreve como usar os utilitários `mongodump`, `mongorestore`, `mongoexport` e `mongoimport` para mover dados para dentro e para fora de um cluster do Amazon DocumentDB.

1. Pré-requisitos: antes de começar, verifique se o cluster do Amazon DocumentDB está provisionado e se você tem acesso a uma instância do Amazon EC2 na mesma VPC que o cluster. Para obter mais informações, consulte Conecte usando o Amazon EC2.

   Para poder usar as ferramentas do utilitário mongo, é necessário ter o pacote mongodb-org-tools instalado na instância do EC2, conforme indicado a seguir.

   ```
   sudo yum install mongodb-org-tools-4.0.18
   ```

   Como o Amazon DocumentDB usa a criptografia Transport Layer Security (TLS) por padrão, também é necessário fazer download do arquivo de autoridade de certificação (CA) do Amazon RDS a fim de usar o shell mongo para se conectar, conforme indicado a seguir.

   ```
   wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
   ```

2. Download de exemplos de dados: para este tutorial, baixe alguns exemplos de dados com informações sobre restaurantes.

   ```
   wget https://raw.githubusercontent.com/ozlerhakan/mongodb-json-files/master/
   datasets/restaurant.json
   ```

3. Importar os exemplos de dados para o Amazon DocumentDB: como os dados estão em um formato JSON lógico, use o utilitário `mongoimport` a fim de importar os dados para o cluster do 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. Despejar os dados com **`mongodump`**: agora que você tem dados no cluster do Amazon DocumentDB, é possível fazer um despejo binário desses dados usando o utilitário `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. Descartar a coleção **`restaurants`**: antes de restaurar a coleção `restaurants` no banco de dados `business`, descarte a coleção que já existe nesse banco de dados, conforme indicado a seguir.

```
use business
```

```
db.restaurants.drop()
```

6. Restaurar dados com **`mongorestore`**: com o despejo binário dos dados da Etapa 3, agora é possível usar o utilitário `mongorestore` para restaurar seus dados para o cluster do 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. Exportar dados usando **`mongoexport`**: para concluir o tutorial, exporte os dados do cluster no formato de um arquivo JSON, da mesma maneira que o arquivo importado na Etapa 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. Validação: é possível validar que a saída da Etapa 5 produz o mesmo resultado que a Etapa 1 com os comandos a seguir.

```
wc -l restaurant.json
```

Saída desse comando:

```
2548 restaurant.json
```

```
wc -l restaurant2.json
```

Saída desse comando:

```
2548 restaurant2.json
```

Considerações sobre snapshot de cluster

O Amazon DocumentDB cria snapshots automáticos diários do seu cluster durante a janela de backup do seu cluster. O Amazon DocumentDB salva os snapshots automáticos da instância do seu cluster de acordo com o período de retenção de retenção especificado. Se necessário, é possível recuperar seu cluster para qualquer momento determinado durante o período de retenção de backup. Os snapshots automáticos não são obtidos enquanto uma operação de cópia está sendo executada na mesma região para o mesmo cluster.

Tópicos

- Armazenamento de backup
- Janela de backup
- Período de retenção de backup
- Copiar criptografia de snapshot de cluster

Além de snapshots automáticos de cluster, você também pode criar manualmente um snapshot de cluster. É possível copiar os snapshots automáticos e manuais. Para obter mais informações, consulte Criação de um snapshot manual de cluster e Cópia de snapshots de cluster do Amazon DocumentDB.

> **Note**
>
> O cluster deve estar no estado disponível para que um snapshot automático seja obtido. Não é possível compartilhar um snapshot de cluster automatizado do Amazon DocumentDB. Como alternativa, crie um snapshot manual copiando o snapshot automatizado e compartilhe essa cópia. Para obter mais informações sobre como copiar um snapshot, consulte Cópia de snapshots de cluster do Amazon DocumentDB. Para obter mais informações sobre a restauração de um cluster com base em um snapshot, consulte Restauração de um snapshot de cluster.

Armazenamento de backup

Seu armazenamento de backup do Amazon DocumentDB para cada um Região da AWS é composto pelo armazenamento de backup necessário para o período de retenção de backup, que inclui snapshots de cluster automáticos e manuais nessa região. O período de retenção de backup padrão é de 1 dia. Para obter mais informações sobre o preço do armazenamento de backup, consulte Definição de preço do Amazon DocumentDB.

Quando você exclui um cluster, todos os seus snapshots automáticos são excluídos e não podem ser recuperados. No entanto, os snapshots manuais não são excluídos quando você exclui um cluster. Se preferir que o Amazon DocumentDB crie um snapshot final (snapshot manual) antes do cluster ser excluído, poderá usar o snapshot final para recuperar o cluster.

Para obter mais informações sobre snapshots e armazenamento, consulte Noções básicas do uso do armazenamento de backup do.

Janela de backup

Os snapshots automáticos são obtidos diariamente durante a janela de backup escolhida. Se o snapshot exigir mais tempo do que o atribuído à janela de backup, o processo de backup continuará até que seja concluído, mesmo que a janela de backup tenha terminado. A janela de backup não pode se sobrepor à janela de manutenção semanal do cluster.

Se você não especificar uma janela de backup preferencial ao criar o cluster, o Amazon DocumentDB atribuirá uma janela de backup padrão de 30 minutos. Essa janela é escolhida aleatoriamente de um bloco de tempo de 8 horas associado à região do cluster. É possível alterar a janela de backup preferencial modificando o cluster. Para obter mais informações, consulte Modificar um cluster do Amazon DocumentDB.

Nome da região	Região	Bloco de tempo UTC
Leste dos EUA (Ohio)	us-east-2	03:00-11:00
Leste dos EUA (Norte da Virgínia)	us-east-1	03:00-11:00
Oeste dos EUA (Oregon)	us-west-2	06:00-14:00
África (Cidade do Cabo)	af-south-1	03:00-11:00
Ásia-Pacífico (Hong Kong)	ap-east-1	06:00-14:00
Ásia-Pacífico (Hyderabad)	ap-south-2	06:30–14:30
Ásia-Pacífico (Malásia)	ap-southeast-5	13:00-21:00
Ásia-Pacífico (Mumbai)	ap-south-1	06:00-14:00
Ásia-Pacífico (Osaka)	ap-northeast-3	12:00-20:00
Ásia-Pacífico (Seul)	ap-northeast-2	13:00-21:00
Ásia-Pacífico (Singapura)	ap-southeast-1	14:00-22:00
Ásia-Pacífico (Sydney)	ap-southeast-2	12:00-20:00
Ásia-Pacífico (Jacarta)	ap-southeast-3	08:00-16:00
Ásia-Pacífico (Melbourne)	ap-southeast-4	11:00-19:00
Ásia-Pacífico (Tailândia)	ap-southeast-7	15:00-23:00
Ásia-Pacífico (Tóquio)	ap-northeast-1	13:00-21:00

Nome da região	Região	Bloco de tempo UTC
Canadá (Central)	ca-central-1	03:00-11:00
Oeste do Canadá (Calgary)	ca-west-1	18:00-02:00
China (Pequim)	cn-north-1	06:00-14:00
China (Ningxia)	cn-northwest-1	06:00-14:00
Europa (Frankfurt)	eu-central-1	21:00-05:00
Europa (Zurique)	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 (Milão)	eu-south-1	02:00-10:00
Europa (Paris)	eu-west-3	23:59-07:29
Europa (Espanha)	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
Oriente Médio (Emirados Árabes Unidos)	me-central-1	05:00-13:00
América do Sul (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

Período de retenção de backup

O período de retenção de backup é o número de dias que um backup automático é retido antes de ser excluído automaticamente. O Amazon DocumentDB é compatível com um período de retenção de backup de 1 a 35 dias.

É possível definir o período de retenção de backup ao criar um cluster. Se você não definir explicitamente o período de retenção de backup, o período de retenção de backup padrão de 1 dia será atribuído ao cluster. Depois de criar um cluster, você pode modificar o período de retenção de backup modificando o cluster usando o Console de gerenciamento da AWS ou o. AWS CLI Para obter mais informações, consulte Modificar um cluster do Amazon DocumentDB.

Copiar criptografia de snapshot de cluster

A criptografia de cluster e snapshot é baseada em uma chave de criptografia do KMS. O ID da chave do KMS é o nome do recurso da Amazon (ARN), o identificador da chave do KMS ou o alias da chave do KMS para a chave de criptografia do KMS.

As seguintes diretrizes e limitações são aplicáveis:

- A criptografia é inferida do cluster ao criar um snapshot. Se o cluster for criptografado, o snapshot do cluster será criptografado com a mesma chave KMS. Se o cluster não for criptografado, o snapshot não será criptografado.
- Se você copiar um snapshot do cluster criptografado da conta Amazon Web Services, é possível especificar um valor para `KmsKeyId` a fim de criptografar a cópia com uma nova chave de criptografia do KMS. Se você não especificar um valor para `KmsKeyId` , a cópia do snapshot do cluster será criptografada com a mesma chave KMS que o snapshot do cluster de origem.
- Se você copiar um snapshot de cluster criptografado compartilhado de outra conta Amazon Web Services, deverá especificar um valor para `KmsKeyId`.
- Para copiar um snapshot de cluster criptografado para outra região Amazon Web Services, configure `KmsKeyId` para o ID da chave KMS que você deseja usar para criptografar a cópia do snapshot de cluster na região de destino. Chaves de criptografia do KMS são específicas da região da Amazon Web Services em que são criadas, e não é possível usar chaves de criptografia de uma região da Amazon Web Services em outra.
- Se você tentar copiar um snapshot de cluster não criptografado e especificar um valor para o parâmetro `KmsKeyId`, um erro será retornado.

Comparação dos snapshots automáticos e manuais

A seguir, estão os principais atributos dos snapshots automáticos e manuais do Amazon DocumentDB (compatível com MongoDB).

Os snapshots automáticos do Amazon DocumentDB têm os recursos principais a seguir:

- Nomeação de snapshots automáticos: nomes de snapshots automáticos seguem o padrão `rds:<cluster-name>-yyyy-mm-dd-hh-mm`, com `yyyy-mm-dd-hh-mm` representando a data e a hora em que o snapshot foi criado.
- Criado automaticamente em uma programação: ao criar ou modificar um cluster, é possível definir o período de retenção de backup para um valor inteiro de 1 a 35 dias. Por padrão, os novos clusters têm um período de retenção de backup de 1 dia. O período de retenção de backup define o número de dias que os snapshots automáticos são mantidos antes de serem excluídos automaticamente. Não é possível desabilitar backups automáticos em clusters do Amazon DocumentDB.

  Além de definir o período de retenção de backup, você também define a janela de backup, a hora do dia durante a qual os snapshots automáticos são criados.
- Exclusão de snapshots automáticos: snapshots automáticos são excluídos quando você exclui o cluster do snapshot automático. Não é possível excluir manualmente um snapshot automático.
- Incremental: durante o período de retenção de backup, as atualizações do banco de dados são registradas para que haja um registro incremental de alterações.
- Restauração de um snapshot automático: é possível restaurar de um snapshot automático usando o Console de gerenciamento da AWS ou a AWS CLI. Ao restaurar a partir de um snapshot usando o AWS CLI, você deve adicionar instâncias separadamente depois que o cluster estiver disponível.
- Compartilhamento: não é possível compartilhar um snapshot de cluster automatizado do Amazon DocumentDB. Como alternativa, crie um snapshot manual copiando o snapshot automatizado e compartilhe essa cópia. Para obter mais informações sobre como copiar um snapshot, consulte Cópia de snapshots de cluster do Amazon DocumentDB. Para obter mais informações sobre a restauração de um cluster com base em um snapshot, consulte Restauração de um snapshot de cluster.
- É possível restaurar a partir de qualquer ponto dentro do período de retenção de backup: como as atualizações de banco de dados são gravadas de forma incremental, é possível restaurar o cluster para qualquer momento determinado dentro do período de retenção de backup.

Ao restaurar a partir de um snapshot automático ou de uma restauração pontual usando o AWS CLI, você deve adicionar instâncias separadamente depois que o cluster estiver disponível.

Os snapshots manuais do Amazon DocumentDB têm os principais atributos a seguir:

- Criados sob demanda — Os snapshots manuais do Amazon DocumentDB são criados sob demanda usando o Amazon DocumentDB Management Console ou. AWS CLI
- Exclusão de um snapshot manual: um snapshot manual é excluído somente quando você o exclui explicitamente usando o console ou a AWS CLI do Amazon DocumentDB. Um snapshot manual não é excluído quando você exclui o cluster.
- Backups completos: quando um snapshot manual é obtido, um backup completo dos dados do cluster é criado e armazenado.
- Nomeação manual do snapshot: você especifica o nome do instantâneo manual. O Amazon DocumentDB não adiciona um selo `datetime` ao nome, então você deve adicionar essas informações se quiser incluí-las no nome.
- Restauração de um snapshot manual: é possível restaurar de um snapshot manual usando o console ou a AWS CLI. Ao restaurar a partir de um snapshot usando o AWS CLI, você deve adicionar instâncias separadamente depois que o cluster estiver disponível.
- Cotas de serviço — Você está limitado a um máximo de 100 instantâneos manuais por. Região da AWS
- Compartilhamento: é possível compartilhar snapshots manuais de cluster, que podem ser copiados por Contas da AWS autorizadas. É possível compartilhar snapshots manuais criptografados ou não criptografados. Para obter mais informações sobre como copiar um snapshot, consulte Cópia de snapshots de cluster do Amazon DocumentDB.
- Você restaura para o momento em que o snapshot manual foi obtido: ao restaurar de um snapshot manual, você restaura para o momento em que o snapshot manual foi obtido.

  Ao restaurar a partir de um snapshot usando o AWS CLI, você deve adicionar instâncias separadamente depois que o cluster estiver disponível.

Criação de um snapshot manual de cluster

Você pode criar um instantâneo manual usando o Console de gerenciamento da AWS ou AWS CLI. O tempo necessário para criar um snapshot varia com o tamanho dos bancos de dados. Ao criar um snapshot, faça o seguinte:

1. Identifique o cluster para fazer backup.
2. Atribua um nome ao snapshot. Isso permite que faça restaurações a partir dele mais tarde.

Using the Console de gerenciamento da AWS

Para criar um instantâneo manual usando o Console de gerenciamento da AWS, você pode seguir qualquer um dos métodos abaixo.

1. Método 1:

    1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
    2. No painel de navegação, escolha Snapshots.

        > **Tip**
        >
        > Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

    3. Na página Snapshots, selecione Create (Criar).
    4. Na página Criar snapshot do cluster:
        a. Identificador de cluster: na lista suspensa de clusters, escolha o cluster do qual deseja criar um snapshot.
        b. Identificador de instantâneo: insira um nome para seu snapshot.

            Restrições de nomenclatura do snapshot:

            - O comprimento é de [1 a 255] letras, números ou hífens.
            - O primeiro caractere deve ser uma letra.
            - Não pode terminar com um hífen ou conter dois hifens consecutivos.

- Deve ser exclusivo para todos os clusters (no Amazon RDS, Amazon Neptune e Amazon DocumentDB) por conta AWS , por região.

c. Escolha Criar.

2. Método 2:

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Clusters.

> **Tip**
>
> Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

3. Na página Clusters, escolha o botão à esquerda do cluster que pretende gerar snapshots.
4. No menu Ações, escolha Tirar snapshot.
5. Na página Create cluster snapshot (Criar snapshot de cluster):

a. Identificador de instantâneo: insira um nome para seu snapshot.

Restrições de nomenclatura do snapshot:

- O comprimento é de [1 a 63] letras, números ou hífens.
- O primeiro caractere deve ser uma letra.
- Não pode terminar com um hífen ou conter dois hifens consecutivos.
- Deve ser exclusivo para todos os clusters (no Amazon RDS, Amazon Neptune e Amazon DocumentDB) por conta AWS , por região.

b. Escolha Criar.

Using the AWS CLI

Para criar um snapshot de cluster usando o AWS CLI, use a `create-db-cluster-snapshot` operação com os parâmetros a seguir.

Parâmetros

- **`--db-cluster-identifier`** — Obrigatório. O nome do cluster do qual você está tirando um snapshot. Esse cluster deve existir e estar disponível.
- **`--db-cluster-snapshot-identifier`** — Obrigatório. O nome do snapshot manual que você está criando.

O exemplo a seguir cria um snapshot chamado `sample-cluster-snapshot` para um cluster chamado `sample-cluster`.

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante à seguinte.

```
{
    "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"
    }
}
```

Cópia de snapshots de cluster do Amazon DocumentDB

No Amazon DocumentDB, você pode copiar snapshots no mesmo Região da AWS ou em um diferente. Região da AWS Você também pode copiar instantâneos compartilhados para sua conta da mesma forma Região da AWS ou de outra Região da AWS. Para obter mais informações sobre o compartilhamento de snapshots, consulte Compartilhamento de snapshots de cluster do Amazon DocumentDB.

> **Note**
>
> O Amazon DocumentDB cobra com base na quantidade de dados de backup e snapshot do e no período em que você os mantém. Para obter informações sobre o armazenamento associado a backups e snapshots do Amazon DocumentDB, consulte Noções básicas do uso do armazenamento de backup do. Para obter informações sobre a definição de preço do armazenamento do Amazon DocumentDB, consulte a página da web Definição de preço do Amazon DocumentDB.

Tópicos

- Copiar snapshots compartilhados
- Copiando instantâneos em Regiões da AWS
- Limitações
- Lidar com a criptografia
- Considerações de grupos de parâmetros
- Cópia de um snapshot de cluster

Copiar snapshots compartilhados

Você pode copiar instantâneos compartilhados com você por outras AWS contas. Se você estiver copiando um instantâneo criptografado que foi compartilhado de outra AWS conta, deverá ter acesso à chave de AWS KMS criptografia que foi usada para criptografar o instantâneo. Para obter mais informações, consulte Lidar com a criptografia.

Copiando instantâneos em Regiões da AWS

Quando você copia um instantâneo para um Região da AWS que é diferente do instantâneo de origem Região da AWS, cada cópia é um instantâneo completo. Uma cópia completa de snapshot contém todos os dados e metadados necessários para restaurar o cluster do Amazon DocumentDB.

Limitações

Algumas limitações ao copiar snapshots:

- Se você excluir um snapshot de origem antes que o snapshot de destino fique disponível, a cópia do snapshot poderá falhar. Verifique se o snapshot de destino possui um status `AVAILABLE` antes de excluir um snapshot de origem.
- É possível ter até cinco solicitações de cópia de snapshot em andamento para uma única região de destino por conta.
- Dependendo do Regiões da AWS envolvido e da quantidade de dados a serem copiados, uma cópia instantânea entre regiões pode levar horas para ser concluída. Em alguns casos, pode haver um grande número de solicitações de cópia de snapshot entre regiões de determinada Região da AWS de origem. Nesses casos, o Amazon DocumentDB pode colocar novas solicitações de cópia entre regiões dessa fonte Região da AWS em uma fila até que algumas cópias em andamento sejam concluídas. Nenhuma informação de andamento é exibida sobre solicitações de cópia enquanto elas estão na fila. As informações sobre o andamento são exibidas quando a cópia é iniciada.

Lidar com a criptografia

É possível copiar um snapshot que foi criptografado usando uma chave de criptografia do AWS KMS . Se você copiar um snapshot criptografado, a cópia desse snapshot também deverá ser criptografada. Se você copiar um instantâneo criptografado dentro do mesmo Região da AWS, poderá criptografar a cópia com a mesma chave de AWS KMS criptografia do instantâneo original

ou especificar uma chave de criptografia diferente. AWS KMS Se você copiar um instantâneo criptografado entre regiões, não poderá usar a mesma chave de AWS KMS criptografia para a cópia usada para o instantâneo de origem, porque AWS KMS as chaves são. Region-specific Em vez disso, você deve especificar uma AWS KMS chave válida no destino Região da AWS n.

O snapshot de origem permanece criptografado ao longo do processo de cópia. Para obter mais informações, consulte Proteção de dados no Amazon DocumentDB.

> **Note**
>
> Para snapshots de cluster do Amazon DocumentDB, você não pode criptografar um snapshot de cluster não criptografado quando você copia o snapshot.

Considerações de grupos de parâmetros

Ao copiar um snapshot entre regiões, a cópia não inclui o grupo de parâmetros usado pelo cluster original do Amazon DocumentDB. Quando você restaura um snapshot para criar um novo cluster, esse cluster obtém o grupo de parâmetros padrão para o qual Região da AWS foi criado. Para dar ao novo cluster os mesmos parâmetros que o original, você deve fazer o seguinte:

1. No destino Região da AWS, crie um grupo de parâmetros do cluster Amazon DocumentDB com as mesmas configurações do cluster original. Se um já existir no novo Região da AWS, você pode usá-lo.
2. Depois de restaurar o snapshot no destino Região da AWS, modifique o novo cluster Amazon DocumentDB e adicione o grupo de parâmetros novo ou existente da etapa anterior. Para obter mais informações, consulte Modificar um cluster do Amazon DocumentDB.

Cópia de um snapshot de cluster

Você pode copiar um cluster do Amazon DocumentDB usando o Console de gerenciamento da AWS ou o AWS CLI, da seguinte forma.

Using the Console de gerenciamento da AWS

Para fazer uma cópia de um snapshot de cluster usando o Console de gerenciamento da AWS, conclua as etapas a seguir. Esse procedimento funciona para copiar snapshots de cluster criptografados ou não criptografados, na mesma região Região da AWS ou em várias regiões.

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Snapshots e, em seguida, escolha o botão à esquerda do snapshot que você deseja copiar.

   > **Tip**
   >
   > Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

3. No menu Actions, escolha Copy.
4. Na página Fazer cópia do snapshot do cluster resultante, preencha a seção Configurações.

   a. Região de destino: opcional. Para copiar o snapshot do cluster para outro Região da AWS, escolha essa opção Região da AWS para Região de destino.

   b. Identificador do novo snapshot: digite um nome para o novo snapshot.

      Restrições de nomenclatura do snapshot de destino:

      - Não pode ser o nome de um snapshot existente.
      - O comprimento é de [1 a 63] letras, números ou hífens.
      - O primeiro caractere deve ser uma letra.
      - Não pode terminar com um hífen ou conter dois hifens consecutivos.
      - Deve ser exclusivo para todos os clusters no Amazon RDS, Neptune e Amazon DocumentDB por Conta da AWS região.

   c. Copiar tags: para copiar as tags existentes no snapshot de origem para a sua cópia do snapshot, escolha Copiar tags.

5. Complete a Encryption-at-rest seção.

   a. Criptografia em repouso: se o snapshot não estiver criptografado, essas opções não estarão disponíveis para você, pois não é possível criar uma cópia criptografada a partir de um snapshot não criptografado. Se seu snapshot estiver criptografado, você poderá alterar o AWS KMS key usado durante a criptografia em repouso.

      Para obter mais informações sobre criptografia de cópias de snapshot, consulte Copiar criptografia de snapshot de cluster.

Para obter mais informações sobre criptografia em repouso, consulte Criptografar dados em repouso do Amazon DocumentDB.

b. AWS KMS Chave — Na lista suspensa, escolha uma das seguintes opções:

- (padrão) aws/rds — O número da conta e o ID da AWS KMS chave são listados após essa opção.
- <some-key-name>: se você tiver criado uma chave, ela estará listada e disponível para você escolher.
- Inserir o ARN da chave: na caixa ARN, insira nome do recurso da Amazon (ARN) para a chave do AWS KMS . O formato do ARN é `arn:aws:kms:<region>:<accountID>:key/<key-id>` .

6. Para criar uma cópia do snapshot selecionado, escolha Copiar snapshot. Como alternativa, é possível escolher Cancelar para não criar uma cópia do snapshot.

Using the AWS CLI

Para fazer uma cópia de um snapshot não criptografado de cluster usando o AWS CLI, use a operação `copy-db-cluster-snapshot` com os parâmetros a seguir. Se você estiver copiando o instantâneo para outro Região da AWS, execute o comando no qual Região da AWS o instantâneo será copiado.

- **`--source-db-cluster-snapshot-identifier`** — Obrigatório. O identificador do snapshot de cluster a ser copiado. O snapshot do cluster deve existir e estar no estado disponível. Se você estiver copiando o snapshot para outro Região da AWS ou se estiver copiando um snapshot de cluster compartilhado, esse identificador deverá estar no formato ARN do snapshot do cluster de origem. Esse parâmetro não diferencia maiúsculas de minúsculas.
- **`--target-db-cluster-snapshot-identifier`** — Obrigatório. O identificador do novo snapshot de cluster a ser criado a partir do snapshot de cluster de origem. Esse parâmetro não diferencia maiúsculas de minúsculas.

  Restrições de nomenclatura do snapshot de destino:

  - Não pode ser o nome de um snapshot existente.
  - O comprimento é de [1 a 63] letras, números ou hífens.
  - O primeiro caractere deve ser uma letra.

  - Não pode terminar com um hífen ou conter dois hifens consecutivos.
  - Deve ser exclusivo para todos os clusters no Amazon RDS, Neptune e Amazon DocumentDB por Conta da AWS região.
- **`--source-region`**— Se você estiver copiando o snapshot para outro Região da AWS, especifique de onde Região da AWS o snapshot criptografado do cluster será copiado.

  Se estiver copiando o snapshot para outra Região da AWS e não especificar `--source-region`, deverá especificar o parâmetro `pre-signed-url` em vez disso. O `pre-signed-url` valor deve ser uma URL que contenha uma solicitação assinada da Signature Versão 4 para que a `CopyDBClusterSnapshot` ação seja chamada na origem da Região da AWS qual o snapshot do cluster é copiado. Para saber mais sobre o`pre-signed-url`, consulte CopyDBClusterSnapshot.
- **`--kms-key-id`**: o identificador da chave do KMS da chave a ser usada para criptografar a cópia do snapshot do cluster.

  Se você estiver copiando um snapshot de cluster criptografado para outro Região da AWS, esse parâmetro será necessário. Você deve especificar uma chave KMS para o destino Região da AWS.

  Se você estiver copiando um snapshot de cluster criptografado no mesmo Região da AWS, o parâmetro AWS KMS key é opcional. A cópia do snapshot do cluster é criptografada com a mesma AWS KMS chave do snapshot do cluster de origem. Se quiser especificar uma nova chave de AWS KMS criptografia a ser usada para criptografar a cópia, você pode fazer isso usando esse parâmetro.
- **`--copy-tags`**: opcional. As tags e os valores a serem copiados.

Para cancelar uma operação de cópia quando ela estiver em andamento, é possível excluir o snapshot do cluster de destino identificado por `--target-db-cluster-snapshot-identifier` ou `TargetDBClusterSnapshotIdentifier` enquanto ele estiver no status copiando.

Example

Exemplo 1: copiar um snapshot não criptografado para a mesma região

O AWS CLI exemplo a seguir cria uma cópia de `sample-cluster-snapshot` named `sample-cluster-snapshot-copy` da Região da AWS mesma forma que o instantâneo de origem.

Quando a cópia é feita, todas as tags do snapshot original são copiadas para a cópia do snapshot.

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante à seguinte.

```
{
    "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

Exemplo 2: Copiar um instantâneo não criptografado em Regiões da AWS

O AWS CLI exemplo a seguir cria uma cópia de`sample-cluster-snapshot`, que tem o ARN. `arn:aws:rds:us-east-1:123456789012:cluster-snapshot:sample-cluster-snapshot` Essa cópia é nomeada `sample-cluster-snapshot-copy` e está no local Região da AWS em que o comando é executado.

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante à seguinte.

```
{
    "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

Exemplo 3: Copiar um instantâneo criptografado em Regiões da AWS

O AWS CLI exemplo a seguir cria uma cópia `sample-cluster-snapshot` da região us-west-2 para a região us-east-1. Este comando é chamado na região us-east-1.

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante à seguinte.

```
{
    "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

Exemplo 4: Copiar um instantâneo compartilhado não criptografado em Regiões da AWS

O AWS CLI exemplo a seguir, account -123456789012, cria uma cópia de um snapshot de cluster não criptografado `sample-cluster-snapshot` compartilhado por account -999999999999 da região us-east-1 para a região us-west-2. Esse comando é chamado na região us-east-2. Para obter mais informações sobre o compartilhamento de snapshots, consulte Compartilhar um snapshot.

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante à seguinte.

```
{
 "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

Exemplo 5: Copiar um instantâneo compartilhado criptografado em Regiões da AWS

O AWS CLI exemplo a seguir, account - 123456789012 cria uma cópia de um snapshot de cluster criptografado `sample-cluster-snapshot` compartilhado pela conta -

999999999999 da região us-east-1 para a região us-west-2. O snapshot de destino é criptografado com uma chave do KMS gerenciada pelo cliente - `arn:aws:kms:us-west-2:123456789012:key/6c1f3264-1797-472b-ba37-03011e682d28`. Esse comando é chamado na região us-east-2. Para obter mais informações sobre o compartilhamento de snapshots, consulte Compartilhar um snapshot.

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante à seguinte.

```
{
 "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 obter mais informações sobre criptografia de cópias de snapshot, consulte Copiar criptografia de snapshot de cluster.
> Para obter mais informações sobre criptografia em repouso, consulte Criptografar dados em repouso do Amazon DocumentDB.

Compartilhamento de snapshots de cluster do Amazon DocumentDB

Com o Amazon DocumentDB, é possível compartilhar um snapshot manual do cluster das formas a seguir:

- O compartilhamento manual de um instantâneo de cluster, criptografado ou não, permite que AWS contas autorizadas copiem o instantâneo.
- O compartilhamento manual de um snapshot de cluster, criptografado ou não, permite que AWS contas autorizadas restaurem diretamente um cluster a partir do snapshot, em vez de fazer uma cópia dele e restaurá-lo a partir dele.

> **Note**
>
> Para compartilhar um snapshot automatizado de cluster, crie um snapshot manual do cluster copiando o snapshot automatizado e, em seguida, compartilhe essa cópia. Esse processo também se aplica aos recursos gerados pelo AWS backup.

Você pode compartilhar um instantâneo manual com até 20 outros Contas da AWS. Também é possível compartilhar um snapshot manual não criptografado como público, disponibilizando-o para todas as contas da . Ao compartilhar um snapshot como público, verifique se as informações privadas não estão incluídas nos snapshots públicos.

Ao compartilhar snapshots manuais com outros Contas da AWS e você restaurar um cluster a partir de um snapshot compartilhado usando a API AWS CLI ou a API Amazon DocumentDB, você deve especificar o Amazon Resource Name (ARN) do snapshot compartilhado como o identificador do snapshot.

Compartilhar um snapshot criptografado

As seguintes restrições se aplicam ao compartilhamento de snapshots criptografados:

- Não é possível compartilhar snapshots criptografados como públicos.
- Você não pode compartilhar um instantâneo que tenha sido criptografado usando a chave de AWS KMS criptografia padrão da conta que compartilhou o instantâneo.

Siga estas etapas para compartilhar snapshots criptografados.

1. Compartilhe a chave de criptografia AWS Key Management Service (AWS KMS) que foi usada para criptografar o instantâneo com qualquer conta que você queira que possa acessar o instantâneo.

   Você pode compartilhar chaves de AWS KMS criptografia com outras AWS contas adicionando as outras contas à política de AWS KMS chaves. Para obter detalhes sobre a atualização de uma política de chaves, consulte Usando políticas de chaves no AWS KMS no Guia do AWS Key Management Service desenvolvedor. Para ver um exemplo de como criar uma política de chaves, consulte Criar uma política do IAM para permitir a cópia do snapshot criptografado, mais adiante neste tópico.
2. Use o AWS CLI, conforme mostrado abaixo, para compartilhar o instantâneo criptografado com as outras contas.

Permitindo acesso a um AWS KMS chave de criptografia

Para Conta da AWS que outra pessoa copie um instantâneo criptografado compartilhado de sua conta, a conta com a qual você compartilha seu instantâneo deve ter acesso à AWS KMS chave que

criptografou o instantâneo. Para permitir que outra conta acesse uma AWS KMS chave, atualize a política de AWS KMS chaves da chave com o ARN da conta com a qual você está compartilhando como principal na política de AWS KMS chaves. Então, permita a ação `kms:CreateGrant`.

Depois de conceder a uma conta acesso à sua chave de AWS KMS criptografia, para copiar seu instantâneo criptografado, essa conta deve criar um usuário AWS Identity and Access Management (IAM), caso ainda não tenha um. Além disso, essa conta também deve anexar uma política do IAM ao usuário do IAM que permita que ele copie um instantâneo criptografado usando sua AWS KMS chave. A conta deve ser um usuário do IAM e não pode ser uma Conta da AWS identidade raiz devido a restrições AWS KMS de segurança.

No exemplo de política de chaves a seguir, o usuário 123451234512 é o proprietário da chave de criptografia. AWS KMS O usuário 123456789012 é a conta com a qual a chave está sendo compartilhada. Essa política de chaves atualizada dá à conta acesso à AWS KMS chave. Isso é feito incluindo o ARN da Conta da AWS identidade raiz do usuário 123456789012 como principal da política e permitindo a ação. `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}}
        }
    ]
}
```

Criar uma política do IAM para permitir a cópia do snapshot criptografado

Quando o externo Conta da AWS tem acesso à sua AWS KMS chave, o proprietário dessa conta pode criar uma política para permitir que um usuário do IAM criado para a conta copie um instantâneo criptografado criptografado com essa AWS KMS chave.

O exemplo a seguir mostra uma política que pode ser anexada a um usuário do IAM para Conta da AWS 123456789012. A política permite que o usuário do IAM copie um instantâneo compartilhado da conta 123451234512 que foi criptografado com a chave na região 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 obter mais detalhes sobre a atualização de uma política de chaves, consulte Políticas de chave no AWS KMS no Guia do desenvolvedor do AWS Key Management Service .

Compartilhar um snapshot

Você pode compartilhar um snapshot manual de cluster do Amazon DocumentDB (ou uma cópia de um snapshot automatizado) usando o ou o Console de gerenciamento da AWS : AWS CLI

Using the Console de gerenciamento da AWS

Para compartilhar um instantâneo usando o Console de gerenciamento da AWS, conclua as seguintes etapas:

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Snapshots.

3. Selecione o snapshot manual que você deseja compartilhar.
4. No menu suspenso Ações, escolha Compartilhar.
5. Escolha uma das opções a seguir para Visibilidade do snapshot de banco de dados.

   - Se a origem não estiver criptografada, escolha Pública para permitir que todas as AWS contas restaurem um cluster a partir do seu snapshot manual. Ou escolha Privado para permitir que somente AWS as contas especificadas por você restaurem um cluster a partir do seu snapshot manual.

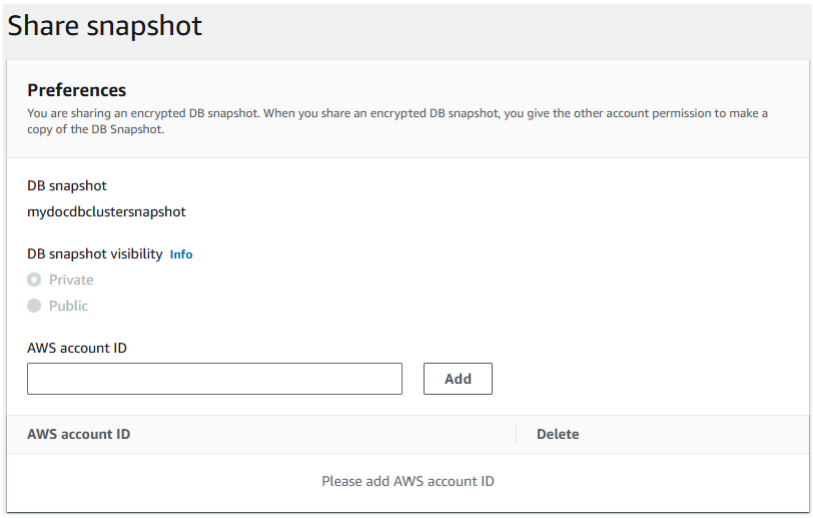
     > **Warning**
     >
     > Se você definir a visibilidade do DB snapshot como Pública, todas as AWS contas poderão restaurar um cluster a partir do seu snapshot manual e ter acesso aos seus dados. Não compartilhe nenhum snapshot manual de cluster que contenha informações privadas como Público.

   - Se a origem estiver criptografada, DB snapshot visibility (Visibilidade do snapshot de banco de dados) será definida como Private (Privada) porque os snapshots criptografados não podem ser compartilhados como públicos.

     > **Note**
     >
     > Os instantâneos que foram criptografados com o padrão não AWS KMS key podem ser compartilhados.

6. Em ID da AWS conta, insira o AWS identificador de conta de uma conta que você deseja permitir para restaurar um cluster a partir do seu snapshot manual e escolha Adicionar. Repita para incluir identificadores de AWS conta adicionais, até 20 AWS contas.

   Se você cometer um erro ao adicionar um identificador de AWS conta à lista de contas permitidas, poderá excluí-lo da lista escolhendo Excluir à direita do identificador de AWS conta incorreto.



7. Depois de adicionar identificadores para todas as AWS contas que você deseja permitir para restaurar o instantâneo manual, escolha Salvar para salvar suas alterações.

Using the AWS CLI

Para compartilhar um snapshot usando o AWS CLI, use a operação Amazon DocumentDB`modify-db-snapshot-attribute`. Use o `--values-to-add` parâmetro para adicionar uma lista das IDs dos Contas da AWS que estão autorizados a restaurar o instantâneo manual.

O exemplo a seguir permite que dois Conta da AWS identificadores, 123451234512 e 123456789012, restaurem o instantâneo chamado. `manual-snapshot1` Ele também remove o valor de atributo `all` para marcar o snapshot como privado.

Para Linux, macOS ou 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"]'
```

A saída dessa operação é semelhante à seguinte.

```
{
    "DBClusterSnapshotAttributesResult": {
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot",
        "DBClusterSnapshotAttributes": [
            {
                "AttributeName": "restore",
                "AttributeValues": [
                    "123451234512",
                    "123456789012"
                ]
            }
        ]
    }
}
```

Para remover um Conta da AWS identificador da lista, use o `--values-to-remove` parâmetro. O exemplo a seguir impede que o Conta da AWS ID 123456789012 restaure o instantâneo.

Para Linux, macOS ou 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"]'
```

A saída dessa operação é semelhante à seguinte.

```
{
    "DBClusterSnapshotAttributesResult": {
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot",
        "DBClusterSnapshotAttributes": [
            {
```

```
                "AttributeName": "restore",
                "AttributeValues": [
                    "123451234512"
                ]
            }
        ]
    }
}
```

Restauração de um snapshot de cluster

O Amazon DocumentDB (compatível com MongoDB) cria um snapshot de cluster do seu volume de armazenamento. É possível criar um novo cluster com a restauração de um snapshot de cluster. Ao restaurar o cluster, você fornece o nome do snapshot do cluster do qual restaurar e um nome para o novo cluster criado pela restauração. Não é possível restaurar de um snapshot para um cluster existente, pois um novo cluster é criado quando você restaura.

Quando você estiver restaurando um cluster a partir de um snapshot de cluster:

- Essa ação restaura apenas o cluster, e não as instâncias desse cluster. É necessário invocar a ação `create-db-instance` para criar instâncias para o cluster restaurado, especificando o identificador do cluster restaurado em `--db-cluster-identifier`. É possível criar instâncias apenas depois que o cluster estiver disponível.
- Não é possível restaurar um snapshot criptografado para um cluster descriptografado. No entanto, você pode restaurar um snapshot não criptografado em um cluster criptografado especificando a chave. AWS KMS
- Para restaurar um cluster a partir de um snapshot criptografado, você deve ter acesso à AWS KMS chave.

> **Note**
>
> Não é possível restaurar um cluster 3.6 para um 4.0, mas pode migrar de uma versão de cluster para outra. Para obter mais informações, acesse Migração e atualização do Amazon DocumentDB.

Using the Console de gerenciamento da AWS

O procedimento a seguir mostra como restaurar um cluster do Amazon DocumentDB a partir de um snapshot de cluster usando o console de gerenciamento do Amazon DocumentDB.

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Snapshots e, em seguida, escolha o botão à esquerda do snapshot que você deseja usar para restaurar o cluster.

   > **Tip**
   >
   > Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

3. No menu Ações, selecione Restaurar.
4. Na página Restaurar snapshot, preencha a seção Configurações.

   a. Identificador de cluster: o nome do novo cluster. É possível aceitar o nome do Amazon DocumentDB fornecido ou digitar o nome que preferir. O nome fornecido pelo Amazon DocumentDB estará no `docdb-` e um timestamp em UTC, por exemplo, `docdb-`*`yyyy-mm-dd-hh-mm-ss`*.

   b. Classe de instância: a classe de instância do novo cluster. É possível aceitar a classe de instância padrão ou escolher uma classe de instância na lista suspensa.

   c. Número de instâncias: o número de instâncias criadas com esse cluster. Você pode aceitar o padrão de 3 instâncias (1 primária read/write e 2 réplicas somente para leitura) ou escolher o número de instâncias na lista suspensa.

5. Para Configuração de armazenamento do cluster, escolha uma opção de armazenamento.

   > **Note**
   >
   > A configuração I/O-Optimized de armazenamento do Amazon DocumentDB está disponível somente nas versões do mecanismo Amazon DocumentDB 5.0 e 8.0.

6. Se você estiver satisfeito com a configuração de cluster, escolha Restaurar cluster e aguarde enquanto o cluster é restaurado.

7. Se preferir alterar algumas configurações, como especificar uma VPC Amazon não padrão ou um grupo de segurança, escolha Mostrar configurações avançadas na parte inferior da página e continuar as etapas seguintes.

   a. Conclua a seção Configurações de rede.

      - Nuvem privada virtual (VPC): aceite a VPC atual ou escolha uma VPC na lista suspensa.
      - Grupo de sub-redes: aceite o grupo de sub-redes `default` ou escolha um na lista suspensa.
      - Grupos de segurança da VPC: aceite o grupo de segurança `default (VPC)` ou escolha um na lista.

   b. Conclua a seção Opções do cluster.

      - Porta do banco de dados: aceite a porta padrão, `27017`, ou use a seta para cima ou para baixo para definir a porta que você deseja usar para conexões de aplicação.

   c. Preencha a seção Criptografia.

      - Criptografia em repouso: se o seu snapshot é criptografado, essas opções não estão disponíveis para você. Se não estiver criptografado, é possível escolher uma das seguintes ações:
        - Para criptografar todos os dados do cluster, escolha Habilitar criptografia em repouso. Se escolher essa opção, você deve designar uma chave mestre.
        - Para não criptografar os dados do cluster, escolha Desabilitar criptografia em repouso. Se escolher essa opção, você concluiu a seção de criptografia.
      - AWS KMS Chave — Escolha uma das seguintes opções na lista suspensa:
        - (padrão) aws/rds — O número da conta e o ID da AWS KMS chave são listados após essa opção.
        - Customer-managed chave — Essa opção está disponível somente se você criou uma chave de criptografia do IAM no console AWS Identity and Access Management (IAM). Você pode escolher a chave para criptografar seu cluster.
        - Insira um ARN de chave — Na caixa ARN, insira o Nome de recurso da Amazon (ARN) para sua chave. AWS KMS O formato do ARN é `arn:aws:kms:<region>:<accountID>:key/<key-id>`.

   d. Complete a seção Exportações de log.

- Selecione os tipos de registro nos quais publicar CloudWatch — Escolha uma das seguintes opções:
  - Ativado — Permite que seu cluster exporte registros DDL para o Amazon CloudWatch Logs.
  - Desativado — Impede que seu cluster exporte registros DDL para o Amazon CloudWatch Logs. Desabilitado é o padrão.
- Perfil do IAM: na lista, escolha Perfil vinculado ao serviço RDS.

e. Complete a seção Tags.

- Adicionar tag: na caixa Chave, insira o nome da tag do cluster. Na caixa Valor, opcionalmente insira o valor da tag. As tags são usadas com políticas AWS Identity and Access Management (IAM) para gerenciar o acesso aos recursos do Amazon DocumentDB e controlar quais ações podem ser aplicadas aos recursos.

f. Complete a seção Proteção contra exclusão.

- Habilitar proteção contra exclusão: protege o cluster contra exclusão acidental. Quando essa opção estiver habilitada, não será possível excluir o cluster.

8. Escolha Restaurar cluster.

Using the AWS CLI

Para restaurar um cluster a partir de um snapshot usando o AWS CLI, use a `restore-db-cluster-from-snapshot` operação com os parâmetros a seguir. Para obter mais informações, consulte RestoreDBClusterFromSnapshot.

- **`--db-cluster-identifier`** — Obrigatório. O nome do cluster que é criado pela operação. Um cluster com este nome não pode existir antes dessa operação.

  Restrições de nomeação de cluster:
  - O comprimento é de [1 a 63] letras, números ou hífens.
  - O primeiro caractere deve ser uma letra.
  - Não pode terminar com um hífen ou conter dois hifens consecutivos.
  - Deve ser exclusivo para todos os clusters no Amazon RDS, Neptune e Amazon DocumentDB por Conta da AWS região.

- **`--snapshot-identifier`** — Obrigatório. O nome do snapshot usado para restauração. Um snapshot com esse nome deve existir e estar no estado disponível.
- **`--engine`** — Obrigatório. Deve ser `docdb`.
- **`--storage-type standard | iopt1`**: opcional. Padrão: `standard`.
- **`--kms-key-id`**: opcional. O ARN do identificador de AWS KMS chave a ser usado ao restaurar um instantâneo criptografado ou criptografar um cluster ao restaurar a partir de um instantâneo não criptografado. O fornecimento da ID da AWS KMS chave faz com que o cluster restaurado seja criptografado com a AWS KMS chave, independentemente de o snapshot ter sido criptografado ou não.

  O formato do `--kms-key-id` é `arn:aws:kms:<region>:<accountID>:key/<key-id>`. Se você não especificar um valor para o parâmetro `--kms-key-id`, ocorrerá o seguinte:

  - Se o snapshot `--snapshot-identifier` for criptografado, o cluster restaurado será criptografado usando a mesma AWS KMS chave usada para criptografar o snapshot.
  - Se o snapshot em `--snapshot-identifier` não estiver criptografado, o cluster restaurado não será criptografado.

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante à seguinte.

```
{
    "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
    }
}
```

Depois que o status do cluster estiver disponível, crie pelo menos uma instância para o cluster.

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante à seguinte.

```
{
    "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"
    }
}
```

Restauração para um ponto no tempo

Você pode restaurar um cluster em qualquer momento que esteja dentro do período de retenção de backup do cluster usando o Console de gerenciamento da AWS ou AWS Command Line Interface (AWS CLI).

> **Note**
>
> Não é possível restaurar um ponto no tempo de um cluster 3.6 para um 4.0, mas pode migrar de uma versão de cluster para outra. Para obter mais informações, acesse Migração e atualização do Amazon DocumentDB.

Lembre-se do seguinte ao restaurar um cluster para um momento determinado.

- O novo cluster é criado com a mesma configuração do cluster de origem, exceto pelo fato de que o novo cluster é criado com o grupo de parâmetros padrão. Para definir o grupo de parâmetros do novo cluster como o grupo de parâmetros do cluster de origem, modifique o cluster depois que ele estiver disponível. Para obter mais informações sobre como modificar um cluster, consulte Modificar um cluster do Amazon DocumentDB.

Using the Console de gerenciamento da AWS

É possível restaurar um cluster para um point-in-time que esteja dentro do período de retenção de backup concluindo o seguinte usando o Console de gerenciamento da AWS.

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Clusters. Na lista de clusters, escolha o botão à esquerda do cluster que você deseja restaurar.

   > **Tip**
   >
   > Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

3. No menu Ações, escolha Restaurar para um determinado momento.
4. Preencha a seção Hora de restauração, que especifica a data e a hora da restauração.

   a. Data de restauração: escolha ou insira uma data entre a Hora de restauração mais antiga e a Hora de restauração mais recente.

   b. Hora de restauração: escolha ou insira uma hora, minuto e segundos entre a Hora de restauração mais antiga e a Hora de restauração mais recente.

5. Preencha a seção Configuração.

   a. Identificador de cluster: aceite o identificador padrão ou insira um identificador que você preferir.

      Restrições de nomeação de cluster:

      - O comprimento é de [1 a 63] letras, números ou hífens.
      - O primeiro caractere deve ser uma letra.

- Não pode terminar com um hífen ou conter dois hifens consecutivos.
- Deve ser exclusivo para todos os clusters no Amazon RDS, Neptune e Amazon DocumentDB por Conta da AWS região.

b. Classe da instância: na lista suspensa, escolha a classe de instância que você deseja usar para as instâncias do cluster.

c. Número de instâncias: na lista suspensa, escolha o número de instâncias que você deseja criar quando o cluster é restaurado.

6. Para Configuração de armazenamento do cluster, escolha uma opção de armazenamento.

> **Note**
>
> A configuração I/O-Optimized de armazenamento do Amazon DocumentDB está disponível somente nas versões do mecanismo Amazon DocumentDB 5.0 e 8.0.

7. Opcional. Para definir as configurações de rede e as opções do cluster e ativar exportações de log, escolha Mostrar configurações avançadas e preencha as seções a seguir. Caso contrário, siga para a próxima etapa.

- Configurações de rede
  1. Nuvem privada virtual (VPC) na lista suspensa, escolha a VPC na qual você deseja usar esse cluster.
  2. Grupo de sub-redes: na lista suspensa, escolha o grupo de sub-redes para esse cluster.
  3. Grupos de segurança da VPC: na lista suspensa, escolha os grupos de segurança da VPC para esse cluster.

- Opções do cluster
  1. Porta: aceite a porta padrão (27017) ou use as setas para cima e para baixo para definir a porta de comunicação com esse cluster.

- Exportações de log
  1. Registros de auditoria — Selecione essa opção para permitir a exportação de registros de auditoria para o Amazon CloudWatch Logs. Se você selecionar essa opção, será necessário habilitar `audit_logs` no grupo de parâmetros personalizado do cluster. Para obter mais informações, consulte Auditar eventos do Amazon DocumentDB.

2. Registros do Profiler — Selecione essa opção para permitir a exportação dos logs do Profiler Operation Profiler para o Amazon Logs. CloudWatch Se você selecionar essa opção, também será necessário modificar os seguintes parâmetros no grupo de parâmetros personalizado do cluster:
   - `profiler`: defina como `enabled`.
   - `profiler_threshold_ms` defina como um valor `[0-INT_MAX]` para configurar o limite para operações de criação de perfil.
   - `profiler_sampling_rate`: defina como um valor `[0.0-1.0]` para configurar o percentual de operações lentas para perfilar.

   Para obter mais informações, consulte Definir o perfil das operações do Amazon DocumentDB.
3. Registros do perfilador — Exporte os registros do perfilador para a Amazon CloudWatch
4. Perfil do IAM: na lista suspensa, escolha Perfil vinculado ao serviço RDS.

- Tags
  1. Adicionar tag: na caixa Chave, insira o nome da tag do cluster. Na caixa Valor, opcionalmente insira o valor da tag. As tags são usadas com políticas do (IAM) AWS Identity and Access Management para gerenciar acesso aos recursos do Amazon DocumentDB e controlar quais ações podem ser aplicadas aos recursos.
- Proteção contra exclusão
  1. Habilitar proteção contra exclusão: protege o cluster contra exclusão acidental. Quando essa opção estiver habilitada, não será possível excluir o cluster.

8. Para restaurar o cluster, escolha Criar cluster. Como alternativa, escolha Cancelar para cancelar a operação.

Using the AWS CLI

Para restaurar um cluster para um momento determinado usando o período de retenção de backup do snapshot, use a operação `restore-db-cluster-to-point-in-time` com os parâmetros a seguir.

- **`--db-cluster-identifier`**: obrigatório. O nome do novo cluster a ser criado. Esse cluster não pode existir antes da operação. O valor do parâmetro deve atender às seguintes restrições.

Restrições de nomeação de cluster:

- O comprimento é de [1 a 63] letras, números ou hífens.
- O primeiro caractere deve ser uma letra.
- Não pode terminar com um hífen ou conter dois hifens consecutivos.
- Deve ser exclusivo para todos os clusters no Amazon RDS, Neptune e Amazon DocumentDB por Conta da AWS região.

- **`--restore-to-time`**: a data e a hora, em formato UTC, em que o cluster será restaurado. Por exemplo, `.2018-06-07T23:45:00Z`

  Restrições de tempo:

  - Devem ser anteriores ao último momento restaurável do cluster.
  - Devem ser especificadas se o parâmetro `--use-latest-restorable-time` não for especificado.
  - Não podem ser especificadas se o parâmetro `--use-latest-restorable-time` for `true`.
  - Não podem ser especificadas se o valor do parâmetro `--restore-type` for `copy-on-write`.
- **`--source-db-cluster-identifier`**: o nome do cluster de origem a partir do qual restaurar. Esse cluster deve existir e estar disponível.
- **`--use-latest-restorable-time`** ou **`--no-use-latest-restorable-time`**: se deseja restaurar para a última hora de backup restaurável. Não pode ser especificado se o parâmetro `--restore-to-time` não for especificado.
- **`--storage-type standard | iopt1`**: opcional. Padrão: `standard`.

A AWS CLI operação restaura `restore-db-cluster-to-point-in-time` somente o cluster, não as instâncias desse cluster. É necessário invocar a operação `create-db-instance` para criar instâncias para o cluster restaurado, especificando o identificador do cluster restaurado em `--db-cluster-identifier`. Você só pode criar instâncias após a conclusão da operação `restore-db-cluster-to-point-in-time` e com o cluster restaurado em estado disponível.

Example

O exemplo a seguir cria o `sample-cluster-restored` do snapshot `sample-cluster-snapshot` para o último momento restaurável.

Para Linux, macOS ou 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

O exemplo a seguir cria o `sample-cluster-restored` do snapshot `sample-cluster-snapshot` para 03:15 de 11 de dezembro de 2018 (UTC), que está dentro do período de retenção de backup de `sample-cluster`.

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante à seguinte.

```
{
    "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
    }
}
```

Exclusão de um snapshot de cluster

Um instantâneo manual é um backup completo que é excluído somente quando você o exclui manualmente usando o Console de gerenciamento da AWS ou AWS CLI. Não é possível excluir

manualmente um snapshot automático porque os snapshots automáticos são excluídos apenas quando o período de retenção do snapshot expira ou quando você exclui o cluster do snapshot.

Using the Console de gerenciamento da AWS

Para excluir um snapshot manual do cluster usando o Console de gerenciamento da AWS, conclua as etapas a seguir.

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Snapshots.

   > **Tip**
   >
   > Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

3. Na lista de snapshots, escolha o botão à esquerda do snapshot que você deseja excluir. O tipo do snapshot deve ser manual.

   1. É possível verificar se o tipo do snapshot é manual verificando se ele está listado como `manual` ou `automatic` na coluna Tipo.
4. No menu Ações, escolha Excluir. Se a opção Excluir estiver indisponível, você provavelmente escolheu um snapshot automático.
5. Na tela de confirmação de exclusão, para eliminar o snapshot, escolha Excluir. Para manter o snapshot, escolha Cancelar.

Using the AWS CLI

Um snapshot de cluster manual do Amazon DocumentDB é um backup completo que é possível excluir manualmente usando a AWS CLI. Não é possível excluir manualmente um snapshot automático.

Para excluir um instantâneo manual do cluster usando o AWS CLI, use a `delete-db-cluster-snapshot` operação com os parâmetros a seguir.

Parâmetros

- **`--db-cluster-snapshot-identifier`** — Obrigatório. O nome do snapshot manual a ser excluído.

O exemplo a seguir exclui o snapshot de cluster `sample-cluster-snapshot`.

Para Linux, macOS ou 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
```

A saída desta operação lista os detalhes do snapshot do cluster excluído.

Monitoramento e registro no Amazon DocumentDB

Monitorar seus AWS serviços é uma parte importante para manter seus sistemas saudáveis e funcionando de maneira ideal. Colete dados de monitoramento de todas as partes da sua AWS solução para que você possa depurar e corrigir falhas ou problemas de desempenho com mais facilidade. Antes de começar a monitorar suas AWS soluções, considere as seguintes questões:

- Quais são seus objetivos de monitoramento?
- Você fará o monitoramento de quais recursos?
- Com que frequência você fará o monitoramento desses recursos?
- Quais ferramentas de monitoramento você usará?
- Quem é responsável por fazer o monitoramento?
- Quem deve ser notificado e por quais meios se algo der errado?

Para entender os padrões de desempenho atuais, identificar anomalias de desempenho e elaborar métodos para a solução de problemas, você deverá estabelecer métricas de desempenho de linha de base para vários momentos e sob diferentes condições de carga. Ao monitorar sua AWS solução, armazene seus dados históricos de monitoramento para referência futura e para estabelecer suas linhas de base.

Em geral, os valores aceitáveis para as métricas de desempenho dependem do aspecto da linha de base e do que a aplicação está fazendo. Investigue variações consistentes ou tendenciais de sua linha de base. Veja a seguir uma instrução sobre os tipos específicos de métricas:

- Alto uso de CPU ou RAM: valores altos de uso de CPU ou RAM podem ser adequados, desde que estejam de acordo com seus objetivos em relação à sua aplicação (como throughput ou simultaneidade) e sejam esperados.
- Consumo de volume de armazenamento: investigue o consumo de armazenamento (`VolumeBytesUsed`) se o espaço utilizado for consistentemente igual ou superior a 85% do espaço total do volume de armazenamento. Determine se você pode excluir dados do volume de armazenamento ou dados de arquivamento em outro sistema para liberar mais espaço. Para obter mais informações, consulte Armazenamento do Amazon DocumentDB e Cotas do Amazon DocumentDB.

- Tráfego de rede - em relação ao tráfego de rede, fale com o administrador do sistema para entender qual throughput é esperado para sua rede de domínio e conexão com a internet. Inspecione o tráfego de rede caso o throughput seja consistentemente menor do que a esperada.
- Conexões do banco de dados - considere restringir as conexões do banco de dados caso perceba um alto número de conexões de usuários em conjunto com uma diminuição na performance da instância e no tempo de resposta. O melhor número de conexões de usuários para sua instância varia conforme a classe da instância e a complexidade das operações sendo executadas.
- Métricas de IOPS - os valores esperados para as métricas de IOPS dependem da especificação do disco e da configuração do servidor, por isso, use sua linha de base para saber os valores típicos. Inspecione caso os valores sejam consistentemente diferentes da sua linha de base. Para obter a melhor performance de IOPS, confira se o seu conjunto de trabalho típico se adéqua à memória para minimizar as operações de leitura e gravação.

O Amazon DocumentDB (com compatibilidade com o MongoDB) fornece uma variedade de CloudWatch métricas da Amazon que você pode monitorar para determinar a integridade e o desempenho de seus clusters e instâncias do Amazon DocumentDB. Você pode visualizar as métricas do Amazon DocumentDB usando várias ferramentas, incluindo o console do Amazon DocumentDB AWS CLI, a CloudWatch API e o Performance Insights.

Tópicos

- Monitoramento do status de um cluster do Amazon DocumentDB
- Monitoramento do status de uma instância do Amazon DocumentDB
- Visualização das recomendações do Amazon DocumentDB
- Usar assinaturas de eventos do Amazon DocumentDB
- Monitorando o Amazon DocumentDB com CloudWatch
- Registro de chamadas de API do Amazon DocumentDB com AWS CloudTrail
- Definir o perfil das operações do Amazon DocumentDB
- Monitoramento com o Performance Insights

Monitoramento do status de um cluster do Amazon DocumentDB

O status de um cluster indica a integridade do cluster. Você pode visualizar o status de um cluster usando o console do Amazon DocumentDB ou o AWS CLI `describe-db-clusters` comando.

Tópicos

- Valores de status do cluster
- Monitorar o status de um cluster

Valores de status do cluster

A tabela a seguir lista os valores válidos para o status de um cluster.

Status do cluster	Description
`active`	O cluster está ativo. Esse status se aplica somente a clusters elásticos.
`available`	O cluster está íntegro e disponível. Esse status se aplica somente a clusters baseados em instâncias.
`backing-up`	O backup do cluster está sendo executado no momento.
`creating`	O cluster está sendo criado. Ele fica inacessível enquanto está sendo criado.
`deleting`	O cluster está sendo excluído. Ele fica inacessível enquanto está sendo excluído.
`failing-over`	Um failover da instância principal para uma réplica do Amazon DocumentDB está sendo executado.
`inaccessible-encryption-credentials`	A AWS KMS chave usada para criptografar ou descripto

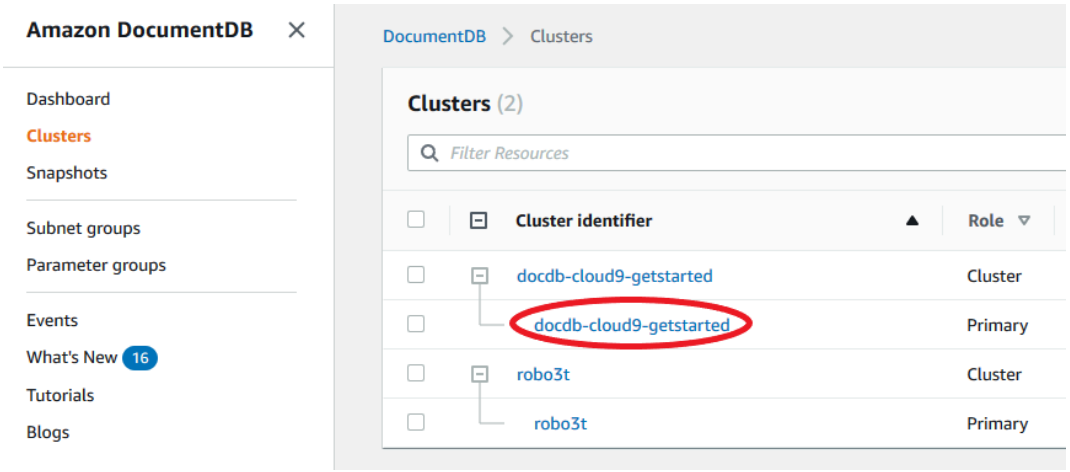
Status do cluster	Description
	grafar o cluster não pode ser acessada.
`maintenance`	Uma atualização de manutenção está sendo aplicada ao cluster. Esse status é usado para a manutenção em nível de cluster que o Amazon DocumentDB programa com antecedência.
`migrating`	Um snapshot do cluster está sendo restaurado em um cluster.
`migration-failed`	Ocorreu uma falha na migração.
`modifying`	O cluster está sendo modificad o devido a uma solicitação do cliente para modificá-lo.
`renaming`	O cluster está sendo renomeado devido a uma solicitação do cliente para renomeá-lo.
`resetting-master-credentials`	As credenciais principais do cluster estão sendo redefinid as devido a uma solicitação do cliente para redefini-las.
`upgrading`	A versão do mecanismo do cluster está sendo atualizada.

Monitorar o status de um cluster

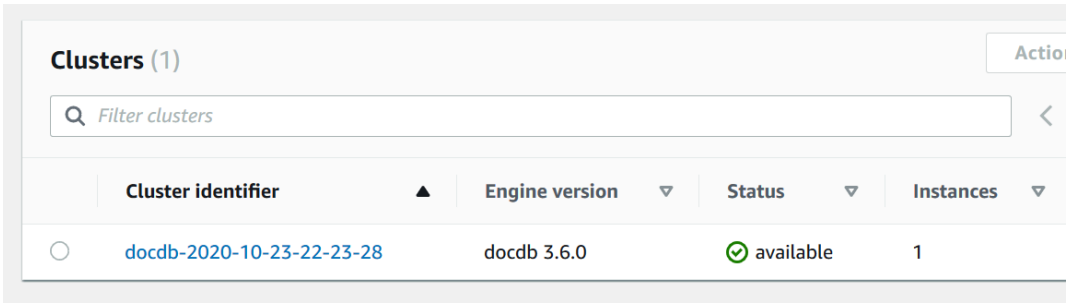
Using the Console de gerenciamento da AWS

Ao usar o Console de gerenciamento da AWS para determinar o status de um cluster, use o procedimento a seguir.

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Clusters.
3. Na caixa de navegação Clusters, você verá a coluna Identificador do cluster. Suas instâncias estão listadas em clusters, semelhante à captura de tela a seguir.



4. Na coluna Identificador de cluster, encontre o nome da instância desejada. Em seguida, para localizar o status da instância, leia essa linha para a coluna Status, como mostrado a seguir.



Using the AWS CLI

Ao usar o AWS CLI para determinar o status de um cluster, use a `describe-db-clusters` operação. O código a seguir descobre o status do cluster `sample-cluster`.

Para Linux, macOS ou 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]'
```

A saída dessa operação é semelhante à seguinte.

```
[
    [
        "sample-cluster",
        "available"
    ]
]
```

Monitoramento do status de uma instância do Amazon DocumentDB

O Amazon DocumentDB fornece informações sobre a condição atual de cada instância configurada no banco de dados.

Há três tipos de status que você pode visualizar para uma instância do Amazon DocumentDB:

- Status da instância: esse status é mostrado na coluna Status da tabela de clusters no Console de gerenciamento da AWS e mostra a condição atual do ciclo de vida da instância. Os valores mostrados na coluna Status são derivados do campo de `Status` da resposta da API `DescribeDBCluster`.
- Status de integridade da instância: Esse status é mostrado na coluna de integridade da instância da tabela de clusters no Console de gerenciamento da AWS e mostra se o mecanismo de banco de dados, o componente responsável pelo gerenciamento e recuperação de dados, está em execução. Os valores mostrados na coluna Instance health são baseados na métrica CloudWatch `EngineUptime` do sistema Amazon.

- Status de manutenção: Esse status é mostrado na coluna Manutenção da tabela de clusters em Console de gerenciamento da AWS e indica o status de qualquer evento de manutenção que precise ser aplicado a uma instância. O status de manutenção é independente do status da outra instância e derivado da API `PendingMaintenanceAction`. Para ter mais informações sobre o status de manutenção, consulte Manutenção do Amazon DocumentDB.

Tópicos

- Valores de status de instâncias
- Monitorando o status da instância usando o Console de gerenciamento da AWS or AWS CLI
- Status de integridade da instância
- Monitorando o status de integridade da instância usando o Console de gerenciamento da AWS

Valores de status de instâncias

A tabela a seguir lista os possíveis valores de status de instâncias e como cada status é cobrado. Ela mostra se você será cobrado pela instância e pelo armazenamento, somente pelo armazenamento ou se não será cobrado. Para todos os status de instância, você sempre será cobrado pelo uso de backup.

status da instância	Faturado	Description
`available`	Faturado	A instância está íntegra e disponível.
`backing-up`	Faturado	No momento, está sendo feito o backup da instância.
`configuring-log-exports`	Faturado	A publicação de arquivos de log no Amazon CloudWatch Logs está sendo ativada ou desativada nesta instância.
`creating`	Não faturado	A instância está sendo criada. A instância não fica inacessível enquanto está sendo criada.
`deleting`	Não faturado	A instância está sendo excluída.

status da instância	Faturado	Description
`failed`	Não faturado	Houve falha na instância e o Amazon DocumentDB não pôde recuperá-la. Para recuperar os dados, execute uma restauração point-in-time no último momento restaurável da instância.
`inaccessible-encryption-credentials`	Não faturado	A AWS KMS chave usada para criptografar ou descriptografar a instância não pôde ser acessada.
`incompatible-network`	Não faturado	O Amazon DocumentDB está tentando executar uma ação de recuperação em uma instância, mas não consegue fazer isso porque a VPC está em um estado que impede a conclusão da ação. Este status pode ocorrer se, por exemplo, todos os endereços IP disponíveis em uma sub-rede estiverem em uso e o Amazon DocumentDB for incapaz de obter um endereço IP para a instância.
`maintenance`	Faturado	O Amazon DocumentDB está aplicando uma atualização de manutenção na instância de banco de dados. Este status é usado para a manutenção de nível de instância que o Amazon DocumentDB agenda com antecedência. Estamos avaliando maneiras de expor ações adicionais de manutenção para clientes com este status.
`modifying`	Faturado	A instância está sendo alterada devido a uma solicitação.

status da instância	Faturado	Description
`rebooting`	Faturado	A instância está sendo reinicializada devido a uma solicitação ou a um processo do Amazon DocumentDB que exige a reinicialização da instância.
`renaming`	Faturado	A instância está sendo renomeada devido a uma solicitação.
`resetting-master-credentials`	Faturado	As credenciais principais da instância estão sendo redefinidas devido a uma solicitação.
`restore-error`	Faturado	A instância encontrou um erro ao tentar restaurar para um determinado point-in-time ou de um snapshot.
`starting`	Faturado para armazenamento	A instância está iniciando.
`stopped`	Faturado para armazenamento	A instância está interrompida.
`stopping`	Faturado para armazenamento	A instância está sendo interrompida.
`storage-full`	Faturado	A instância alcançou sua alocação de capacidade de armazenamento. Este é um status crítico e deve ser resolvido imediatamente. É necessário aumentar o armazenamento verticalmente modificando a instância. Defina CloudWatch os alarmes da Amazon para avisá-lo quando o espaço de armazenamento estiver ficando baixo, para que você não se depare com essa situação.

Monitorando o status da instância usando o Console de gerenciamento da AWS or AWS CLI

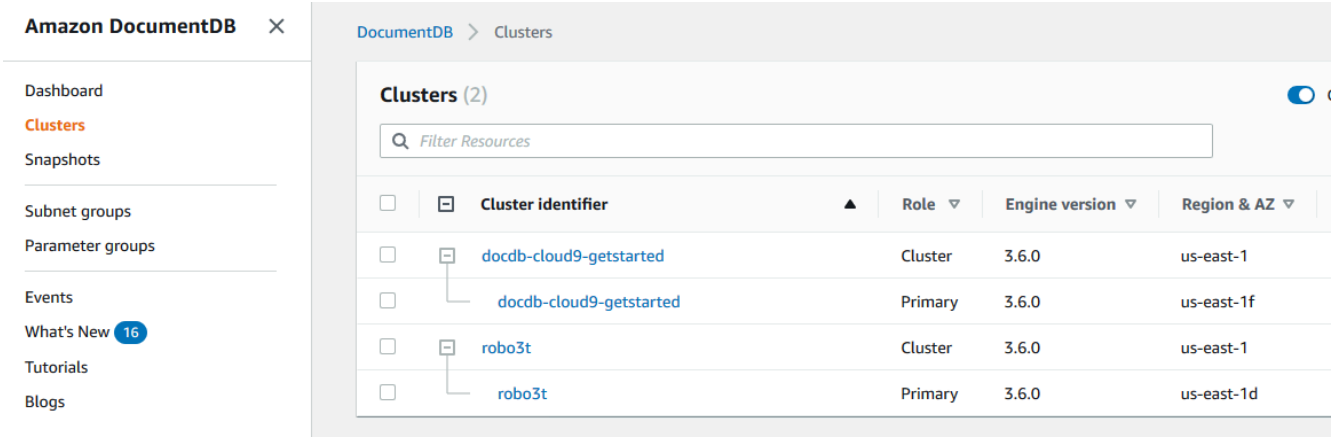
Use o Console de gerenciamento da AWS ou AWS CLI para monitorar o status da sua instância.

Using the Console de gerenciamento da AWS

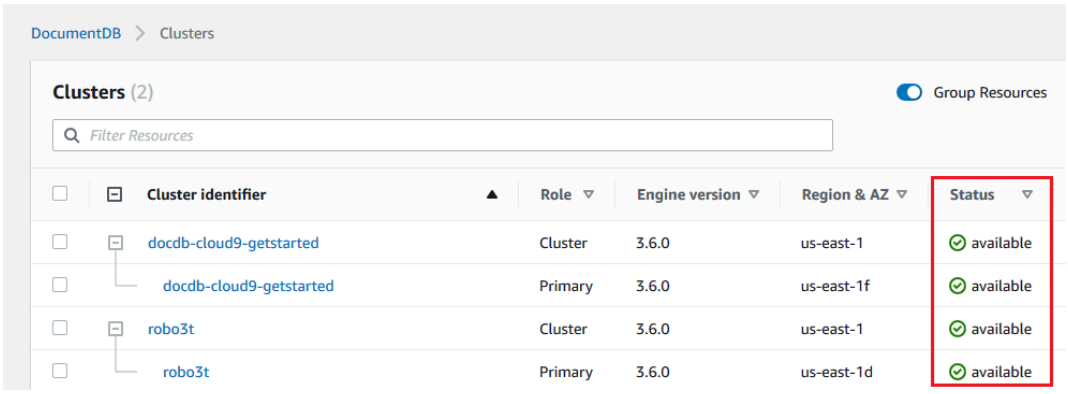
Ao usar o Console de gerenciamento da AWS para determinar o status de um cluster, use o procedimento a seguir.

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Clusters.

> **Note**
>
> Observe que, na caixa de navegação Clusters, a coluna Identificador do cluster mostra tanto os clusters quanto as instâncias. As instâncias são listadas abaixo dos clusters, de forma semelhante à imagem a seguir.



3. Encontre o nome da instância de seu interesse. Em seguida, para localizar o status da instância, leia essa linha para a coluna Status, como mostrado a seguir.



Using the AWS CLI

Ao usar o AWS CLI para determinar o status de um cluster, use a `describe-db-instances` operação. O código a seguir verifica o status da instância `sample-cluster-instance-01`.

Para Linux, macOS ou 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]'
```

A saída dessa operação é semelhante à seguinte.

```
[
          [
              "sample-cluster-instance-01",
              "available"
          ]
      ]
```

Status de integridade da instância

A tabela a seguir lista os valores possíveis do status de integridade das instâncias. A coluna de integridade da instância, localizada na tabela Clusters em Console de gerenciamento da AWS, mostra se o mecanismo do banco de dados, o componente responsável por armazenar, gerenciar e recuperar dados, está operando normalmente. Essa coluna também indica se a métrica `EngineUptime` do sistema, disponível em CloudWatch, está mostrando o status de integridade de cada instância.

Status de integridade da instância	Description
integridade	O mecanismo de banco de dados está sendo executado na instância do Amazon DocumentDB.
não íntegro	O mecanismo de banco de dados não está funcionando ou foi reiniciado há menos de um minuto.

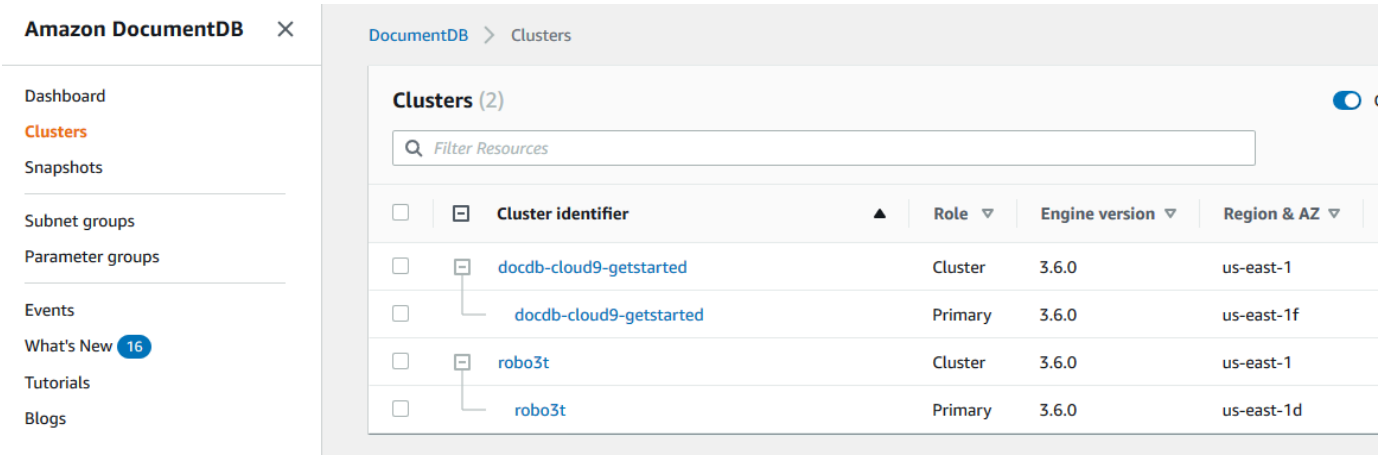
Monitorando o status de integridade da instância usando o Console de gerenciamento da AWS

Use o Console de gerenciamento da AWS para monitorar o status de integridade da sua instância.

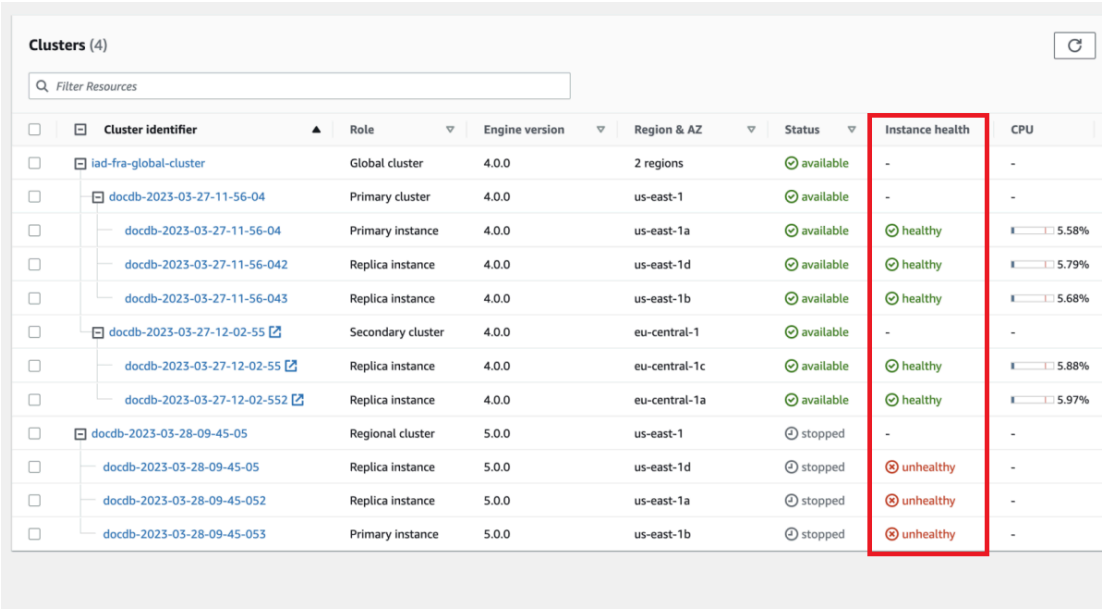
Ao usar Console de gerenciamento da AWS, use as etapas a seguir para entender o status de integridade da instância.

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Clusters.

> **Note**
>
> Na caixa de navegação Clusters, a coluna Identificador do cluster mostra tanto os clusters quanto as instâncias. As instâncias são listadas abaixo dos clusters, de forma semelhante à imagem a seguir.



3. Encontre o nome da instância de seu interesse. Em seguida, para localizar o status da instância, leia essa linha cruzando com a coluna Status, como mostrado a seguir:



> **Note**
>
> A pesquisa do status de integridade da instância ocorre a cada 60 segundos e é baseada na métrica do CloudWatch `EngineUptime` sistema. Os valores na coluna Integridade da instância são atualizados automaticamente.

Visualização das recomendações do Amazon DocumentDB

O Amazon DocumentDB fornece uma lista de recomendações automatizadas para recursos de banco de dados, como instâncias e clusters. Essas recomendações fornecem orientações de práticas recomendadas, analisando as configurações de seu cluster e instância.

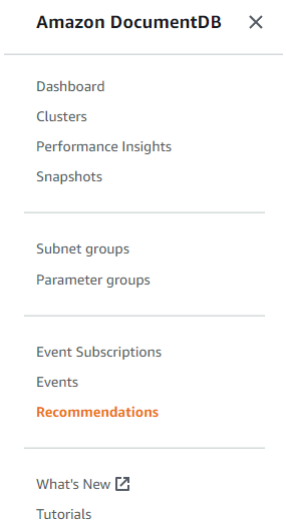
Veja abaixo alguns exemplos dessas recomendações:

Tipo	Descrição	Recomendação	Informações adicionais
Uma instância	O cluster contém apenas uma instância	Desempenho e disponibilidade: adicione outra instância com a mesma classe de instância em uma zona de disponibilidade diferente.	Alta disponibilidade e replicação do Amazon DocumentDB

O Amazon DocumentDB gera recomendações para um recurso quando esse recurso é criado ou modificado. O Amazon DocumentDB também verifica periodicamente seus recursos e gera recomendações.

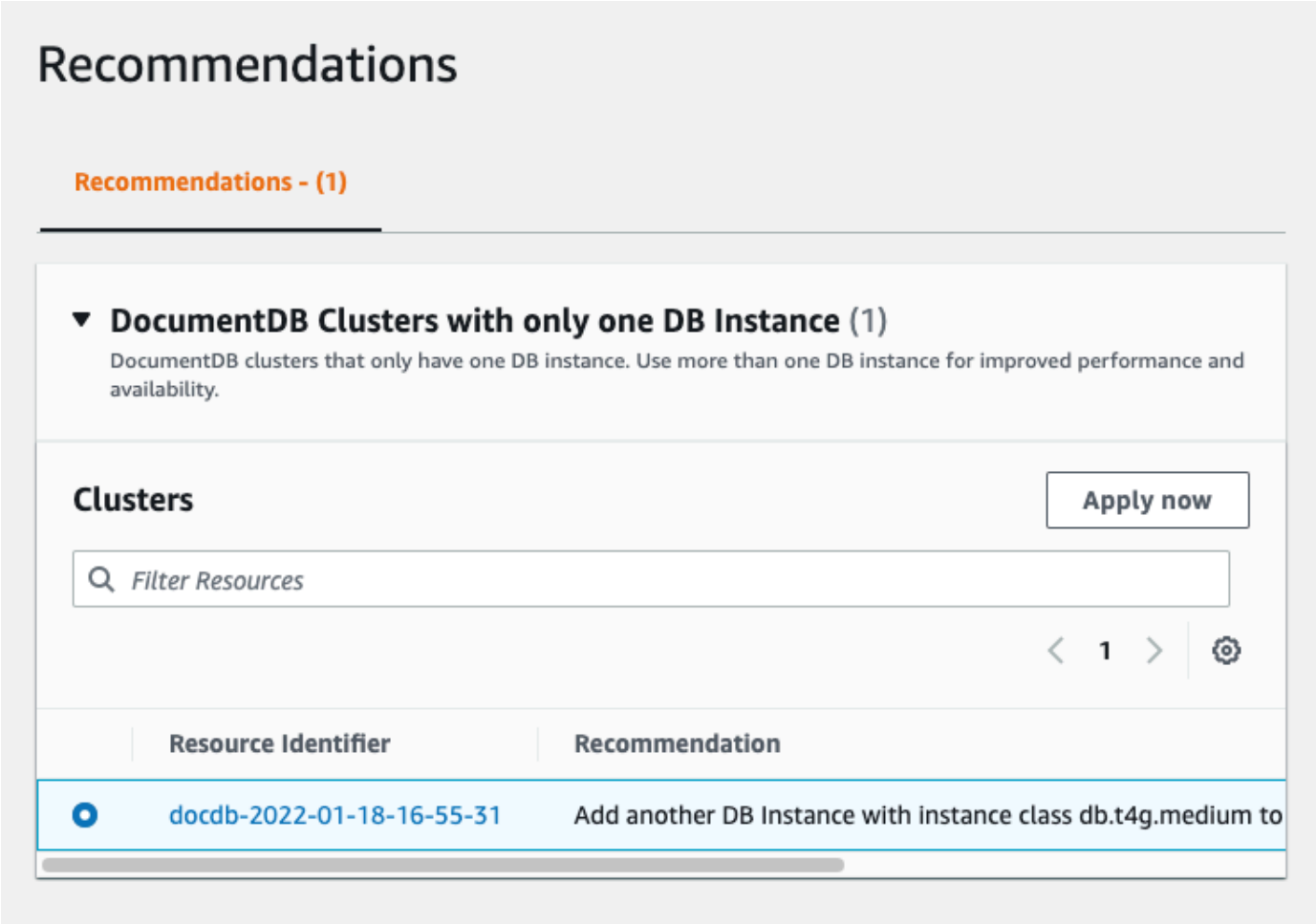
Para visualizar e seguir as recomendações do Amazon DocumentDB

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, selecione Recomendações:



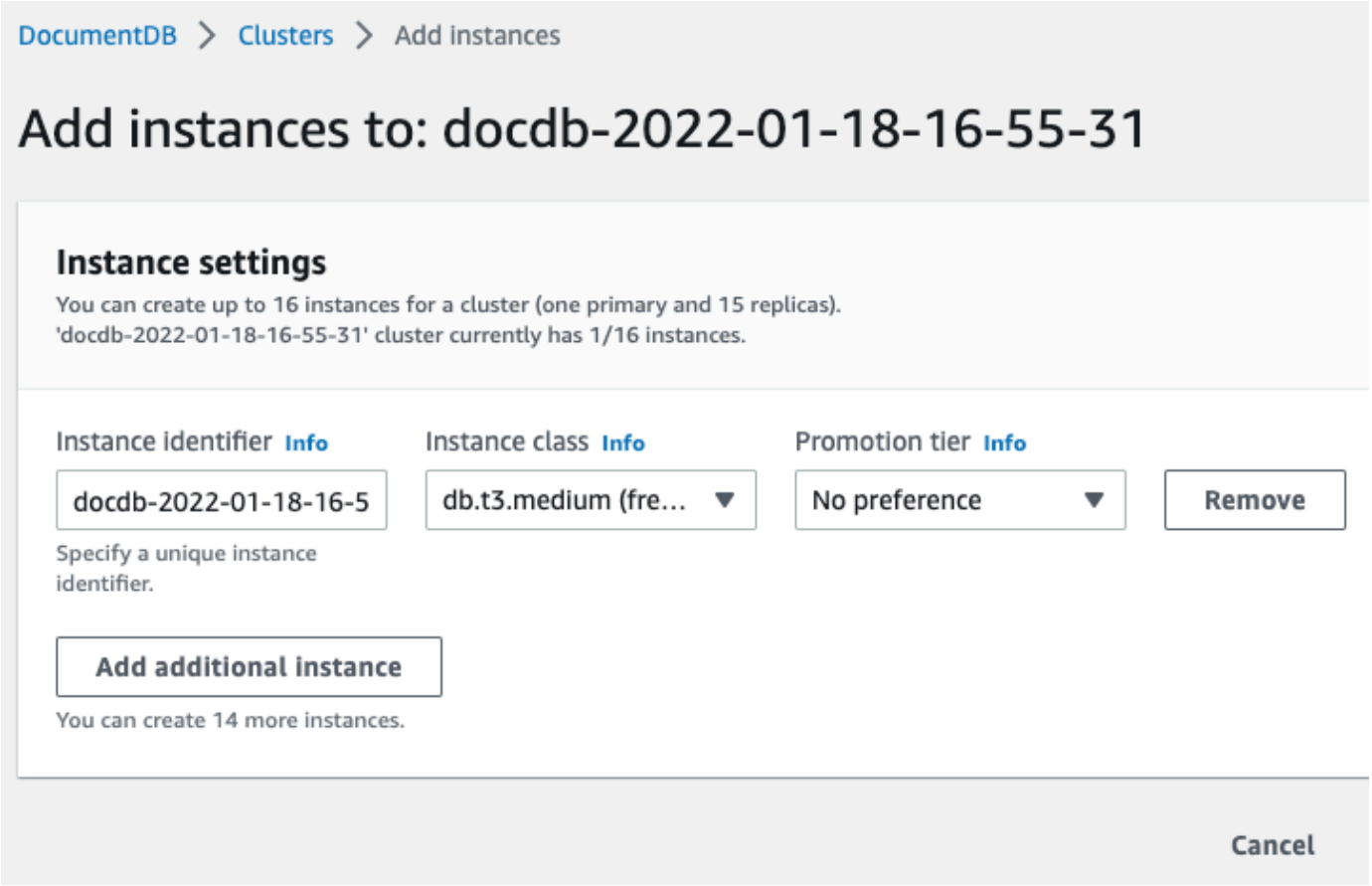
3. Na caixa de diálogo Recomendações, expanda a seção de interesse e selecione a tarefa recomendada.

   No exemplo a seguir, a tarefa recomendada se aplica a um cluster Amazon DocumentDB com apenas uma instância. A recomendação é adicionar outra instância para melhorar o desempenho e a disponibilidade.



4. Escolha Apply now.

   Neste exemplo, a caixa de diálogo Adicionar instâncias é exibida:



5. Modifique as configurações da sua nova instância e escolha Criar.

Usar assinaturas de eventos do Amazon DocumentDB

O Amazon DocumentDB usa o Amazon Simple Notification Service (Amazon SNS) para fornecer uma notificação quando um evento do Amazon DocumentDB ocorre. Essas notificações podem estar em qualquer formato compatível com o Amazon SNS para um Região da AWS, como um e-mail, uma mensagem de texto ou uma chamada para um endpoint HTTP.

O Amazon DocumentDB agrupa esses eventos em categorias nas quais é possível inscrever-se para receber notificações quando ocorre um evento dessa categoria. É possível assinar uma categoria de evento para uma instância, cluster, o snapshot, snapshot do cluster, ou para uma grupo de parâmetros. Por exemplo, se você se inscrever na categoria Backup para uma determinada instância, será notificado sempre que ocorrer um evento relacionado ao backup que afete a instância. Você também recebe uma notificação quando uma assinatura de evento é alterada.

Ocorrem eventos tanto no nível do cluster, quanto no nível da instância, portanto, é possível receber eventos assinar em um cluster ou uma instância.

Assinaturas de eventos são enviadas aos endereços que você fornece ao criar a assinatura. Pode ser interessante criar várias assinaturas diferentes, como uma assinatura para receber todas as notificações de eventos e outra que inclua somente eventos críticos para as suas instâncias de produção. É possível facilmente desativar a notificação sem excluir uma assinatura. Para fazer isso, defina o botão Habilitado como Não no console do Amazon DocumentDB.

> **Important**
>
> O Amazon DocumentDB não garante a ordem dos eventos enviados em um fluxo de eventos. A ordem do evento está sujeita a alterações.

O Amazon DocumentDB usa o nome do recurso da Amazon (ARN) de um tópico do Amazon SNS para identificar cada assinatura. O console do Amazon DocumentDB cria o ARN para você quando você cria a assinatura.

O faturamento das assinaturas de eventos do Amazon DocumentDB é feito por meio do Amazon SNS. As taxas do Amazon SNS se aplicam durante o uso da notificação de eventos. Para obter mais informações, consulte Amazon Simple Notification Service Pricing. Além das cobranças do Amazon SNS, o Amazon DocumentDB não cobra por assinaturas de eventos.

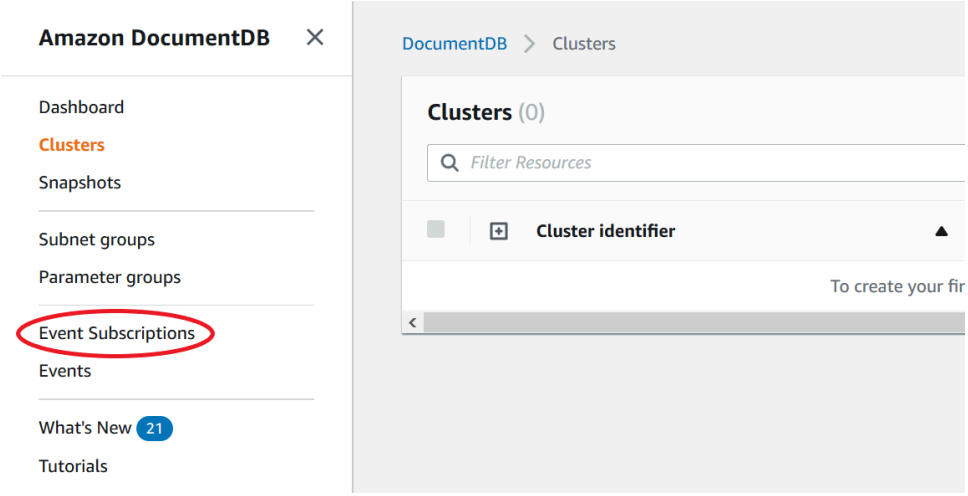
Tópicos

- Tornar-se assinante de eventos do Amazon DocumentDB
- Gerenciar assinaturas de notificação de eventos do Amazon DocumentDB
- Categorias e mensagens de eventos do Amazon DocumentDB

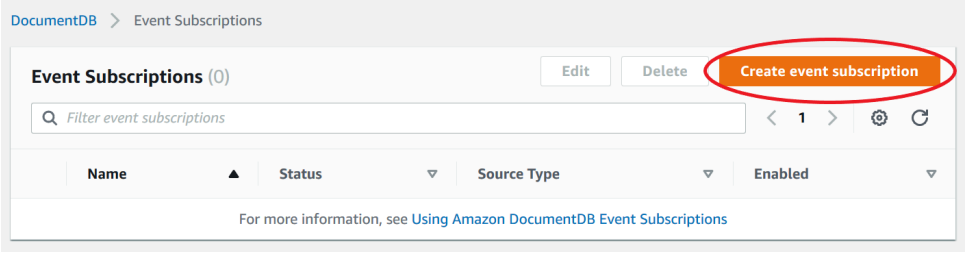
Tornar-se assinante de eventos do Amazon DocumentDB

É possível usar o console do Amazon DocumentDB para se inscrever em assinaturas de eventos, da seguinte forma:

1. Faça login no Console de gerenciamento da AWS athttps://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Assinaturas de eventos.

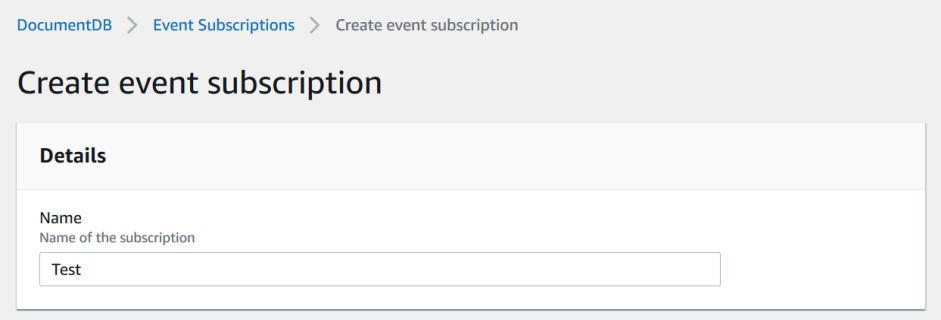


3. No painel Event subscriptions (Assinaturas de eventos), escolha Create event subscription (Criar assinatura de evento).

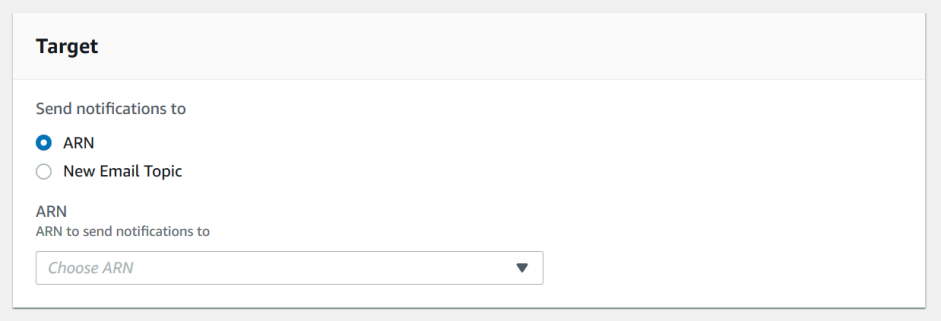


4. Na caixa de diálogo Criar assinatura de evento faça o seguinte:

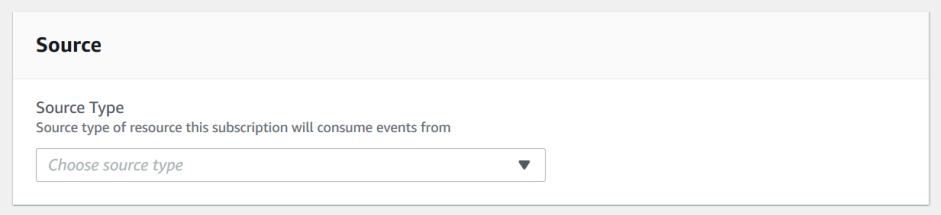
   - Em Name (Nome), insira um nome para a assinatura de notificação de eventos.



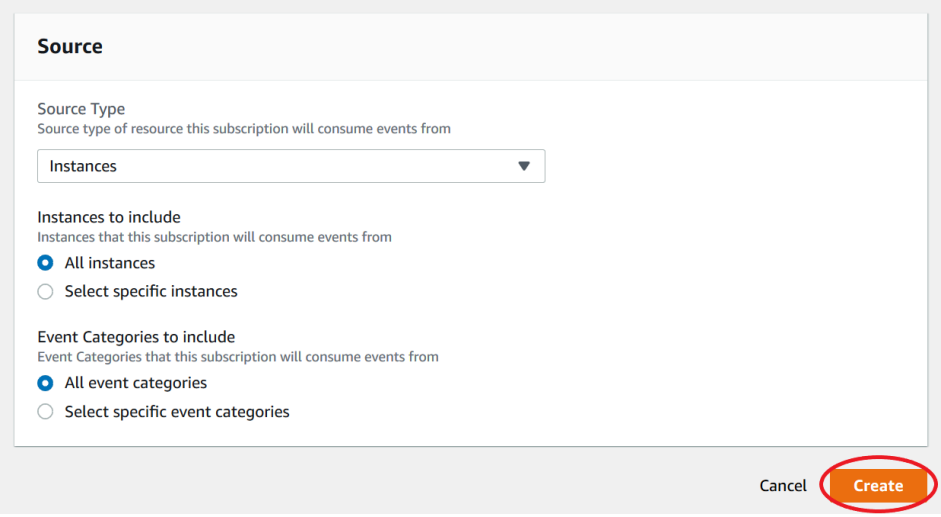
   - Para Destino, escolha para onde você deseja enviar notificações. É possível escolher um ARN existente ou escolher Novo tópico de e-mail para inserir o nome de um tópico e uma lista de destinatários.



- Em Origem, escolha um tipo de origem. Dependendo do tipo de origem que você selecionou, escolha as categorias e origens de eventos para as quais quer receber notificações de eventos.



- Escolha Criar.

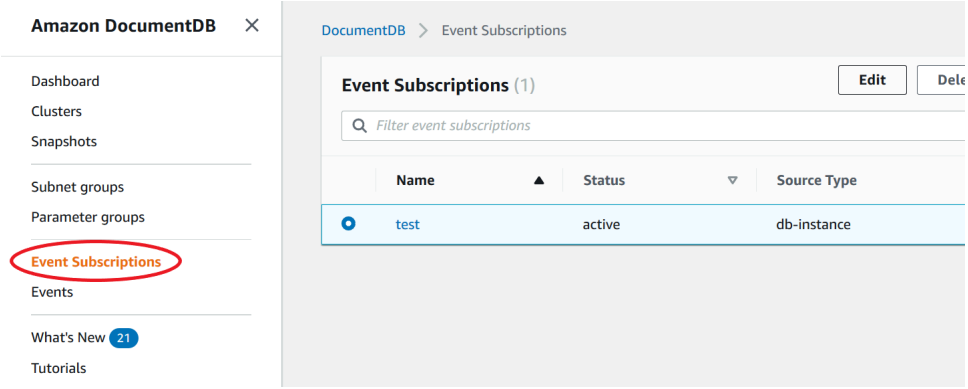


Gerenciar assinaturas de notificação de eventos do Amazon DocumentDB

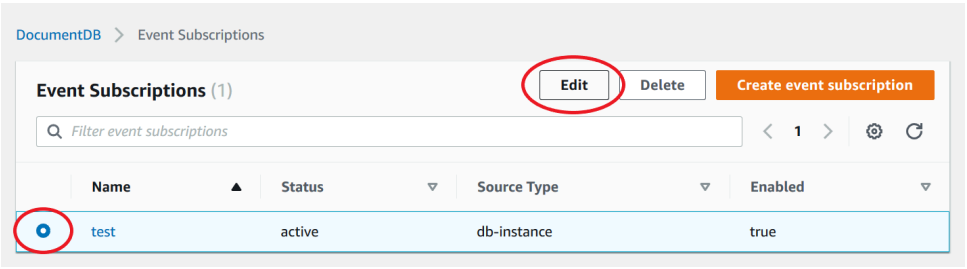
Se você escolher Assinaturas de eventos no painel de navegação do console do Amazon DocumentDB, é possível visualizar categorias de assinatura e uma lista das suas assinaturas atuais. Você também pode modificar ou excluir uma assinatura específica.

Para listar suas atuais assinaturas de notificações de eventos do Amazon DocumentDB

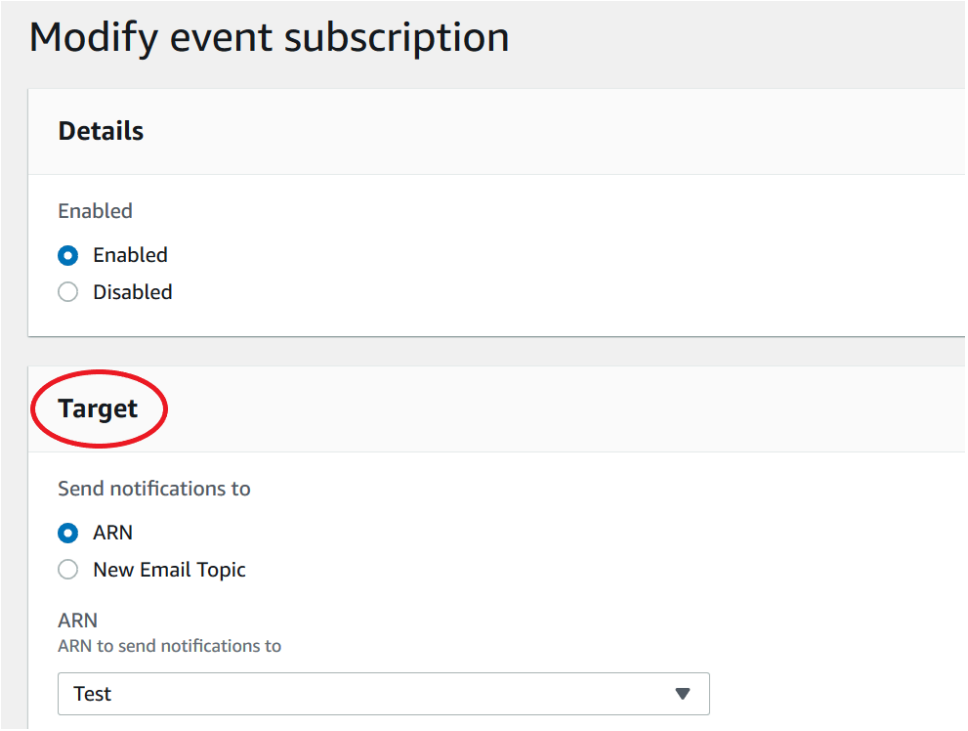
1. Faça login no Console de gerenciamento da AWS athttps://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Event subscriptions (Assinaturas de eventos). O painel Event subscriptions (Assinaturas de eventos) exibirá todas as suas assinaturas de notificação de eventos.



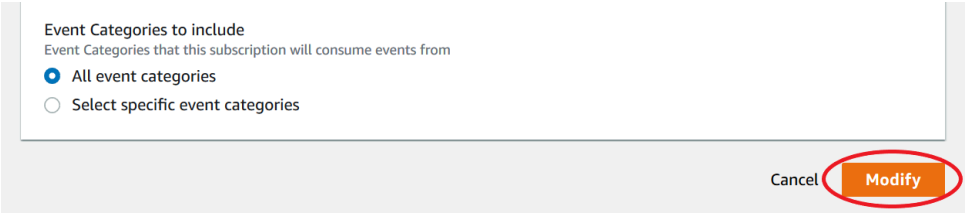
3. No painel Event subscriptions (Assinaturas de eventos), escolha a assinatura que deseja modificar e escolha Edit (Editar).



4. Faça as alterações na assinatura usando as seções Destino ou Origem. É possível adicionar ou remover identificadores de origem selecionando-os ou desmarcando-os na seção Fonte.



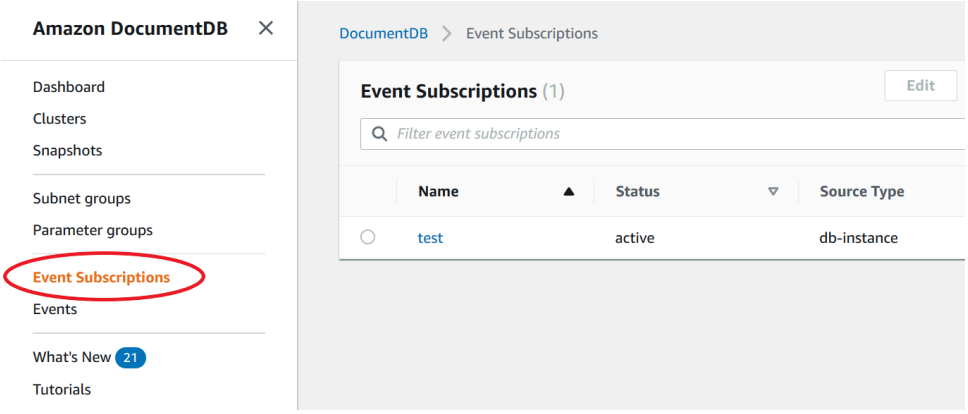
5. Escolha Modificar. O console do Amazon DocumentDB indica que a assinatura está sendo modificada.



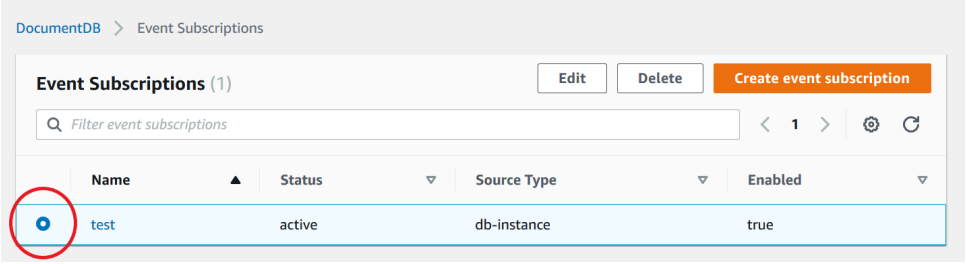
Como excluir uma assinatura de notificação de evento do Amazon DocumentDB

É possível excluir uma assinatura quando não precisar mais dela. Todos os assinantes do tópico não receberão mais notificações de evento especificadas pela assinatura.

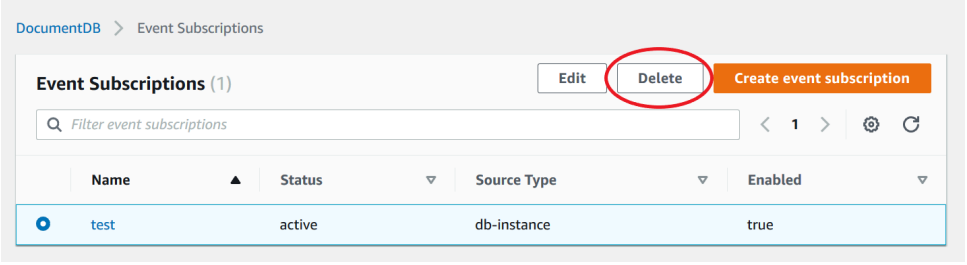
1. Faça login no Console de gerenciamento da AWS athttps://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Assinaturas de eventos.



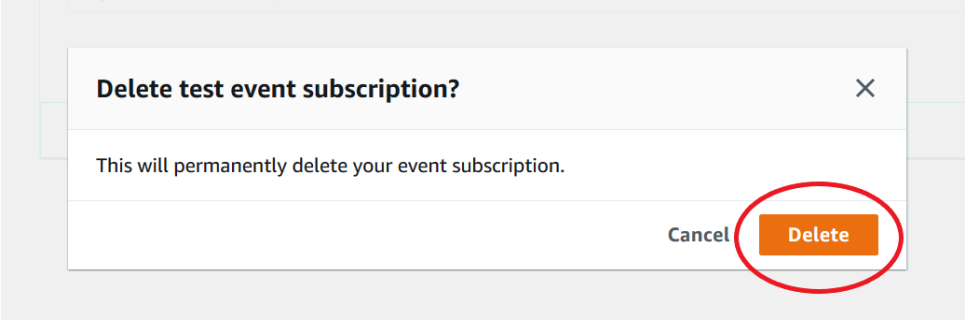
3. No painel Minhas assinaturas de eventos de banco de dados, escolha a assinatura que deseja excluir.



4. Escolha Excluir.



5. Uma janela pop-up aparecerá perguntando se você deseja excluir permanentemente essa notificação. Escolha Excluir.



Categorias e mensagens de eventos do Amazon DocumentDB

O Amazon DocumentDB gera um número significativo de eventos em categorias nas quais é possível fazer uma assinatura usando o console. Cada categoria aplica-se a um tipo de origem, que pode ser uma instância, um cluster, um snapshot ou um grupo de parâmetros.

> **Note**
>
> O Amazon DocumentDB usa definições e definições de eventos existentes do Amazon RDS.

Eventos do Amazon DocumentDB originados de instâncias

Categoria	Description
disponibilidade	A instância foi reiniciada.
disponibilidade	Desligamento da instância.
alteração de configuração	Como aplicar uma modificação à classe de instância.
alteração de configuração	Conclusão da aplicação de modificação à classe de instância.
alteração de configuração	Redefinir as credenciais principais.
criação	Instância criada.
exclusão	Instância excluída
falha	Houve falha na instância de banco de dados devido a uma configuração incompatível ou a um problema do armazenamento subjacente. Inicie um point-in-time-restore para a instância.
notificação	A Instância foi interrompida.
notificação	A instância foi iniciada.

Categoria	Description
notificação	A instância está sendo iniciada porque excede o tempo máximo permitido para permanecer parada.
recuperação	A recuperação da instância começou. O tempo de recuperação variará dependendo da quantidade de dados a serem recuperados.
recuperação	A recuperação da instância está completa.
patches de segurança	A atualização do sistema operacional está disponível para a sua instância. Para obter informações sobre a aplicação de atualizações, consulte Manutenção do Amazon DocumentDB.

Eventos do Amazon DocumentDB originados de um cluster

Categoria	Description
criação	Cluster criado
exclusão	Cluster excluído.
failover	Como promover novamente a primária anterior.
failover	Failover concluído para a instância.
failover	Iniciou o failover para a instância de banco de dados: %s
failover	Iniciou o mesmo failover da AZ para a instância de banco de dados: %s
failover	Iniciou o failover cruzado da AZ para a instância de banco de dados: %s

Categoria	Description
manutenção	O cluster foi corrigido.
manutenção	O cluster do banco de dados está em um estado que não pode ser atualizado: %s
notificação	O cluster parou.
notificação	O cluster iniciou.
notificação	Falha na parada do cluster.
notificação	O cluster está sendo iniciado porque excede o tempo máximo permitido para permanecer parado.
notificação	Cluster renomeado de %s para %s.

Eventos do Amazon DocumentDB originados de um snapshot de cluster

AA tabela a seguir mostra a categoria de evento e uma lista de eventos quando um snapshot de cluster do Amazon DocumentDB é o tipo de origem.

Categoria	Description
backup	Criação de um snapshot manual do cluster.
backup	Snapshot de cluster manual criado.
backup	Criando snapshot de cluster automatizado.
backup	Snapshot de cluster automatizado criado.

Eventos do Amazon DocumentDB originados de um grupo de parâmetros

A tabela a seguir mostra a categoria de evento e uma lista de eventos quando um parameter group é o tipo de origem.

Categoria	Description
alteração de configuração	Parâmetro atualizado de %s para %s com o método de aplicação %s

Monitorando o Amazon DocumentDB com CloudWatch

O Amazon DocumentDB (com compatibilidade com o MongoDB) se integra à Amazon CloudWatch para que você possa reunir e analisar métricas operacionais para seus clusters. Você pode monitorar essas métricas usando o CloudWatch console, o console do Amazon DocumentDB, o AWS Command Line Interface (AWS CLI) ou a CloudWatch API.

CloudWatch também permite definir alarmes para que você possa ser notificado se um valor de métrica ultrapassar um limite especificado por você. Você pode até mesmo configurar o Amazon CloudWatch Events para tomar medidas corretivas caso ocorra uma violação. Para obter mais informações sobre o uso CloudWatch de alarmes, consulte a CloudWatch documentação da Amazon.

Tópicos

- Métricas do Amazon DocumentDB
- Visualização CloudWatch de dados
- Dimensões do Amazon DocumentDB
- Monitoramento de métricas de contadores
- Monitorar conexões de banco de dados

Métricas do Amazon DocumentDB

Para monitorar a integridade e o desempenho do cluster e das instâncias do Amazon DocumentDB, é possível visualizar as seguintes métricas no console do Amazon DocumentDB.

> **Note**
>
> As métricas nas tabelas a seguir se aplicam tanto a clusters elásticos quanto baseados em instâncias.

Tópicos

- Métricas de utilização de recursos
- Métricas de latência
- NVMe-backed métricas de instância
- Métricas operacionais
- Command-level métricas
- Métricas de throughput
- Métricas do sistema
- Métricas de instância T3

Métricas de utilização de recursos

Métrica	Description	
`BackupRetentionPeriodStorageUsed`	A quantidade total de armazenamento de backup em bytes usada para oferecer suporte ao atributo de restauração pontual na janela de retenção do Amazon DocumentDB. Incluído no total relatado pela métrica `TotalBackupStorageBilled` . Calculado separadamente para cada cluster do Amazon DocumentDB.	
`ChangeStreamLogSize`	A quantidade de armazenamento usada pelo cluster para armazenar o log do fluxo de alterações em megabytes. Esse valor é um subconjunto do armazenamento total do cluster (`VolumeBytesUsed`)	

Métrica	Description
	e afeta o custo do cluster. Para obter informações sobre preço de armazenamento, consulte a página de produto do Amazon DocumentDB. O tamanho do log do fluxo de alterações dependerá do número de alterações que ocorrerem no cluster e da duração da retenção do log do fluxo de alterações. Para obter mais informações sobre fluxos de alterações, consulte Usar fluxos de alterações com o Amazon DocumentDB.
`CPUUtilization`	O percentual de CPU usado por uma instância.
`DatabaseConnections`	O número de conexões (ativas e ociosas) abertas em uma instância tomada com uma frequência de 1 minuto.
`DatabaseConnectionsMax`	O número máximo de conexões de banco de dados abertas (ativas e ociosas) em uma instância em um período de 1 minuto.
`DatabaseConnectionsLimit`	O número máximo de conexões simultâneas de banco de dados (ativas e ociosas) permitidas em uma instância a qualquer momento.

Métrica	Description
DatabaseConnectionsActiveMax	O número máximo de conexões ativas de banco de dados em uma instância em um período de 1 minuto. As conexões ativas são aquelas que estão executando uma operação no momento, em oposição às conexões ociosas. Para as cotas de conexão ativas por tipo de instância, consulteCotas de instâncias.
DatabaseCursors	O número de cursores abertos em uma instância obtida com uma frequência de 1 minuto.
DatabaseCursorsMax	O número máximo de cursores abertos em uma instância em um período de 1 minuto.
DatabaseCursorsLimit	O número máximo de cursores permitidos em uma instância em um determinado momento.
DatabaseCursorsTimedOut	O número de cursores que atingiram o tempo limite em um período de 1 minuto.
FreeableMemory	A quantidade de memória de acesso aleatório disponível, em bytes.

Métrica	Description
FreeLocalStorage	Essa métrica informa a quantidade de armazenam ento disponível para cada instância para tabelas temporárias e logs. Esse valor depende da classe da instância. É possível aumentar a quantidade de espaço de armazenamento gratuito de uma instância escolhendo uma classe de instância maior para ela. (Isso não se aplica ao DocumentDB Serverless.)
LowMemThrottleQueu eDepth	A profundidade da fila para solicitações que são limitadas devido à pouca memória disponível obtida com uma frequência de 1 minuto.
LowMemThrottleMaxQ ueueDepth	A profundidade máxima da fila para solicitações que são limitadas devido à pouca memória disponível em um período de 1 minuto.
LowMemNumOperation sThrottled	O número de solicitações que são limitadas devido à pouca memória disponível em um período de 1 minuto.

Métrica	Description
SnapshotStorageUsed	A quantidade total de armazenamento de backup em bytes consumida por todos os snapshots de um determinado cluster do Amazon DocumentDB fora da janela de retenção de backup. Incluído no total relatado pela métrica TotalBackupStorageBilled . Calculado separadamente para cada cluster do Amazon DocumentDB.
SwapUsage	A quantidade de troca usada na instância.
TotalBackupStorageBilled	A quantidade total de armazenamento de backup em bytes para a qual você é cobrado para determina do cluster do Amazon DocumentDB. Inclui o armazenamento de backup medido pelas métricas BackupRetentionPeriodStorageUsed e SnapshotStorageUsed . Calculado separadamente para cada cluster do Amazon DocumentDB.

Métrica	Description
`TransactionsOpen`	O número de transações abertas em uma instância realizada com uma frequência de 1 minuto.
`TransactionsOpenMax`	O número máximo de transações abertas em uma instância em um período de 1 minuto.
`TransactionsOpenLimit`	O número máximo de transações simultâneas permitidas em uma instância em um determinado momento.
`VolumeBytesUsed`	A quantidade de armazenamento usada pelo cluster em bytes. Esse valor afeta o custo do cluster. Para obter informações sobre preço, consulte a página do produto Amazon DocumentDB.

Métricas de latência

Métrica	Description
`DBClusterReplicaLagMaximum`	A quantidade máxima de atraso, em milissegundos, entre a instância principal e cada instância do Amazon DocumentDB no cluster.
`DBClusterReplicaLagMinimum`	A quantidade mínima de atraso, em milissegundos,

Métrica	Description
	entre a instância principal e cada instância de réplica no cluster.
`DBInstanceReplicaLag`	O tempo de atraso, em milissegundos, ao replicar atualizações da instância principal para uma instância de réplica.
`ReadLatency`	O tempo médio gasto por I/O operação de disco.
`WriteLatency`	O tempo médio, em milissegundos, gasto por I/O operação de disco.

NVMe-backed métricas de instância

Métrica	Description
`NVMeStorageCacheHitRatio`	O percentual de solicitações atendidas pelo cache hierárquico.
`FreeNVMeStorage`	A quantidade de espaço de armazenamento NVMe Ephemeral disponível.
`ReadIOPSNVMeStorage`	O número médio de I/O operações de leitura de disco para armazenamento NVMe temporário.
`ReadLatencyNVMeStorage`	O tempo médio gasto por I/O operação de leitura de disco

Métrica	Description
	para armazenamento NVMe temporário.
`ReadThroughputNVMeStorage`	O número médio de bytes lidos do disco por segundo para armazenamento local.
`WriteIOPSNVMeStorage`	O número médio de I/O operações de gravação em disco no armazenamento NVMe temporário.
`WriteLatencyNVMeStorage`	O tempo médio gasto por I/O operação de gravação em disco para armazenamento NVMe temporário.
`WriteThroughputNVMeStorage`	O número médio de bytes gravados no disco por segundo para o armazenamento local.

Métricas operacionais

Métrica	Description
`DocumentsDeleted`	O número de documentos excluídos em um período de 1 minuto.
`DocumentsInserted`	O número de documentos inseridos em um período de 1 minuto.

Métrica	Description
DocumentsReturned	O número de documentos devolvidos em um período de 1 minuto.
DocumentsUpdated	O número de documentos atualizados em um período de 1 minuto.
OpcountersCommand	O número de comandos emitidos em um período de 1 minuto.
OpcountersDelete	O número de operações de exclusão emitidas em um período de 1 minuto.
OpcountersGetmore	O número de getmores emitidos em um período de 1 minuto.
OpcountersInsert	O número de operações de inserção emitidas em um período de 1 minuto.
OpcountersQuery	O número de consultas emitidas em um período de 1 minuto.
OpcountersUpdate	O número de operações de atualização emitidas em um período de 1 minuto.
TransactionsStarted	O número de transações iniciadas em uma instância em um período de 1 minuto.

Métrica	Description
`TransactionsCommitted`	O número de transações confirmadas em uma instância em um período de 1 minuto.
`TransactionsAborted`	O número de transações abortadas em uma instância em um período de 1 minuto.
`TTLDeletedDocuments`	O número de documentos excluídos por um TTLMonitor em um período de 1 minuto.

Command-level métricas

O Amazon DocumentDB publica métricas de latência e simultaneidade por comando no. CloudWatch Essas métricas ajudam você a identificar quais operações contribuem mais para sua carga de trabalho e a detectar problemas de latência traseira ou pressão de simultaneidade.

Cada comando na tabela a seguir emite três métricas em um período de 1 minuto:

- `AvgDuration`— O tempo médio de execução do comando, em microssegundos.
- `P100Duration`— O tempo máximo de execução do comando (100º percentil), em microssegundos.
- `MaxConcurrent`— O número máximo de execuções simultâneas do comando.

Os nomes das métricas seguem o padrão`Command.`*`commandName`*`.`*`metricName`*. Por exemplo, `Command.find.AvgDuration` ou `Command.aggregate.MaxConcurrent`.

Métrica	Description
`Command.find`	Leia consultas usando find.
`Command.insert`	Operações de inserção de documentos.

Métrica	Description
`Command.findAndModify`	Operações atômicas de busca e modificação.
`Command.update`	Operações de atualização de documentos.
`Command.delete`	Operações de exclusão de documentos.
`Command.aggregate`	Execuções de pipeline de agregação.
`Command.count`	Operações de contagem.
`Command.distinct`	Consultas de valor distintas.
`Command.getMore`	Operações de recuperação em lote do cursor.
`Command.abortTransaction`	Operações de cancelamento de transação.
`Command.commitTransaction`	Operações de confirmação de transação.

Métricas de throughput

Métrica	Description
`NetworkReceiveThroughput`	A quantidade de throughput de rede, em bytes por segundo, recebida dos clientes por cada instância no cluster. Essa throughput não inclui o tráfego de rede entre instâncias no cluster e o volume do cluster.
`NetworkThroughput`	A quantidade de throughput de rede, em bytes por

Métrica	Description	
	segundo, recebida e transmiti da aos clientes por cada instância no cluster do Amazon DocumentDB. Essa throughput não inclui o tráfego de rede entre instâncias no cluster e o volume do cluster.	
`NetworkTransmitThr oughput`	A quantidade de throughpu t de rede, em bytes por segundo, enviada aos clientes por cada instância no cluster. Essa throughput não inclui o tráfego de rede entre instância s no cluster e o volume do cluster.	
`ReadIOPS`	O número médio de I/O operações de leitura de disco por segundo. O Amazon DocumentDB relata as IOPS de leitura e gravação separadamente em intervalos de 1 minuto.	
`ReadThroughput`	O número médio de bytes lidos do disco por segundo.	
`StorageNetworkRece iveThroughput`	O throughput de rede, em bytes por segundo, recebido do volume de armazenam ento do cluster do Amazon DocumentDB por cada instância no cluster.	

Métrica	Description
`StorageNetworkTransmitThroughput`	O throughput de rede, em bytes por segundo, enviado ao volume de armazenamento do cluster do Amazon DocumentDB por cada instância no cluster.
`StorageNetworkThroughput`	O throughput de rede, em bytes por segundo, recebido e transmitido ao volume de armazenamento do cluster do Amazon DocumentDB por cada instância no cluster do Amazon DocumentDB.

Métrica	Description	
`VolumeReadIOPs`	O número médio de I/O operações de leitura faturadas de um volume de cluster, relatado em intervalos de 5 minutos. As operações de leitura faturadas são calculada s no nível de volume do cluster, agregadas a partir de todas as instâncias no cluster de banco de dados e posteriormente relatadas em intervalos de 5 minutos. O valor é calculado tomando o valor da métrica de operações de leitura em um período de 5 minutos. É possível determinar a quantidade de operações de leitura faturadas por segundo, tomando o valor da métrica de operações de leitura faturadas e dividindo por 300 segundos. Por exemplo, se `VolumeRea dIOPs` retorna 13.686, então as operações de leitura cobradas por segundo são 45 (13.686/300 = 45,62). Você acumula operações de leitura faturadas para consultas que solicitam páginas de banco de dados que não estão presentes no cache do buffer e, portanto,	

Métrica	Description	
	devem ser carregadas a partir do armazenamento. É possível perceber picos em operações de leitura faturadas, pois os resultados da consulta são lidos a partir do armazenamento e depois são carregados no cache do buffer.	

Métrica	Description	
`VolumeWriteIOPs`	O número médio de I/O operações de gravação faturadas de um volume de cluster, relatado em intervalos de 5 minutos. As operações de gravação faturadas são calculadas no nível de volume do cluster, agregadas a partir de todas as instâncias no cluster e posteriormente relatadas em intervalos de 5 minutos. O valor é calculado tomando o valor da métrica de operações de leitura em um período de 5 minutos. É possível determinar a quantidade de operações de gravação faturadas por segundo, tomando o valor da métrica de operações de gravação faturadas e dividindo por 300 segundos. Por exemplo, se `VolumeWriteIOPs` retorna 13.686, então as operações de leitura cobradas por segundo são 45 (13.686/300 = 45,62). Observe que `VolumeReadIOPs` as `VolumeWriteIOPs` métricas são calculadas pela camada de armazenamento do DocumentDB e incluem o iOS	

Métrica	Description	
	executado pelas instância s primária e de réplica. Os dados são agregados a cada 20-30 minutos e depois reportados em intervalos de 5 minutos, emitindo assim o mesmo ponto de dados para a métrica no período de tempo. Se você estiver procurando por uma métrica para correlaci onar com suas operações de inserção em um intervalo de 1 minuto, é possível usar a métrica WriteIOps em nível de instância. A métrica está disponível na guia de monitoramento da sua instância primária do Amazon DocumentDB.	
`WriteIOPS`	O número médio de I/O operações de gravação em disco por segundo. Quando usados em nível de cluster, `WriteIOPs` são avaliados em todas as instâncias do cluster. As IOPS de leitura e gravação são relatadas separadamente, em intervalos de um minuto.	
`WriteThroughput`	O número médio de bytes gravados no disco por segundo.	

Métricas do sistema

Métrica	Description	
`AvailableMVCCIds`	Um contador que mostra o número de operações de gravação restantes disponíveis antes de chegar a zero. Quando esse contador chegar a zero, seu cluster entrará no modo somente de leitura até que os IDs sejam recuperados e reciclados. O contador diminui a cada operação de gravação e aumenta à medida que a coleta de resíduos recicla IDs de MVCC antigos.	
`BufferCacheHitRatio`	O percentual de solicitações atendidas pelo cache de buffer.	
`DiskQueueDepth`	O número de I/O operações que estão aguardando para serem gravadas ou lidas do disco.	
`EngineUptime`	A quantidade de tempo, em segundos, em que a instância está executando.	
`IndexBufferCacheHitRatio`	O percentual de solicitações de índice atendidas pelo cache de buffer. É possível ver um aumento maior que 100 por cento na métrica logo após eliminar um índice, uma coleção ou um banco	

Métrica	Description	
	de dados. Isto será corrigido automaticamente após 60 segundos. Essa limitação será corrigida em uma atualização de patch futura.	
`LongestActiveGCRuntime`	Duração em segundos do processo de coleta de resíduos ativo mais longo. Atualiza a cada minuto e rastreia somente as operações ativas, excluindo os processos concluídos dentro da janela de um minuto.	

Métricas de instância T3

Métrica	Description	
`CPUCreditUsage`	O número de créditos de CPU gastos durante o período de medição.	
`CPUCreditBalance`	O número de créditos de CPU que uma instância acumulou. Esse saldo é esgotado quando a CPU apresenta expansões e os créditos de CPU são gastos com mais rapidez do que são ganhos.	
`CPUSurplusCreditBalance`	O número de créditos de CPU excedentes gastos para manter o desempenho da	

Métrica	Description	
	CPU quando o CPUCredit Balance valor é zero.	
`CPUSurplusCreditsCharged`	O número de créditos de CPU excedentes que ultrapassam o número máximo de créditos de CPU que podem ser ganhos em um período de 24 horas, resultando em uma cobrança adicional. Para obter mais informações, consulte Monitoring your CPU credits.	

Visualização CloudWatch de dados

Você pode visualizar CloudWatch os dados da Amazon usando o CloudWatch console, o console do Amazon DocumentDB AWS Command Line Interface (AWS CLI) ou a CloudWatch API.

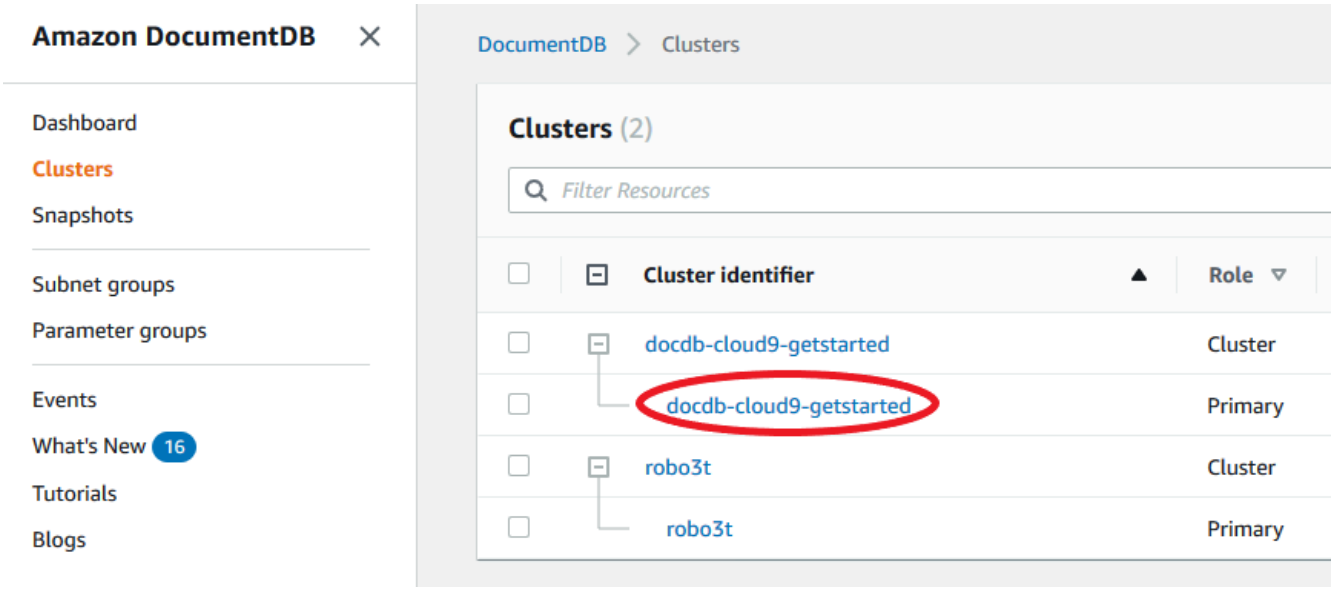
Using the Console de gerenciamento da AWS

Para visualizar CloudWatch as métricas usando o console de gerenciamento do Amazon DocumentDB, conclua as etapas a seguir.

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Clusters.

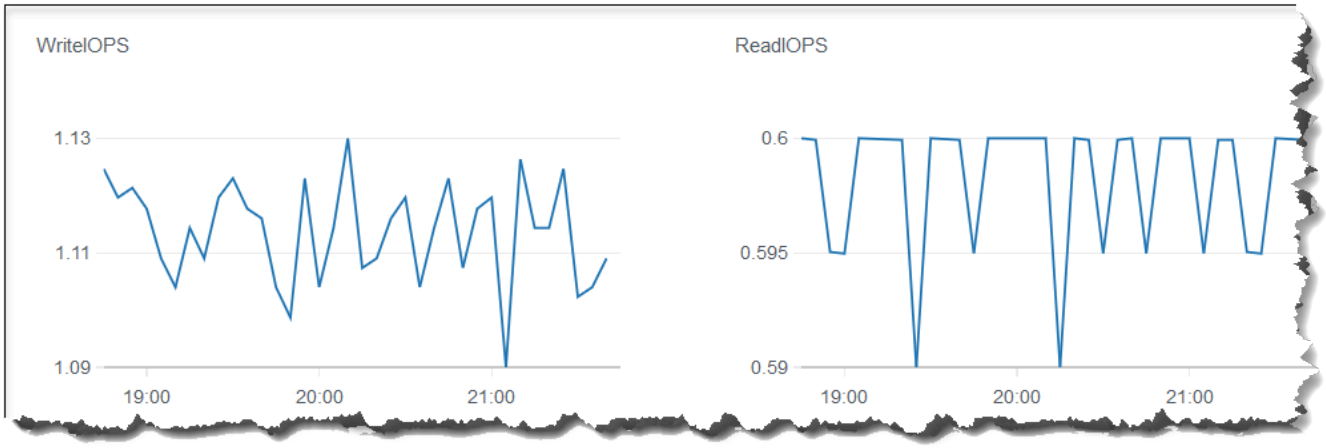
   > **Tip**
   >
   > Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

3. Na caixa de navegação Clusters, você verá a coluna Identificador do Cluster. Suas instâncias estão listadas em clusters, semelhante à captura de tela a seguir.



4. Na lista de instâncias, escolha o nome da instância para a qual você deseja métricas.
5. Na página de resumo da instância resultante, escolha a guia Monitoramento para visualizar representações gráficas das métricas da sua instância do Amazon DocumentDB. Como um gráfico deve ser gerado para cada métrica, pode levar alguns minutos para que os CloudWatch gráficos sejam preenchidos.

   A imagem a seguir mostra as representações gráficas de duas CloudWatch métricas no console do Amazon DocumentDB `WriteIOPS` e. `ReadIOPS`



Using the CloudWatch Management Console

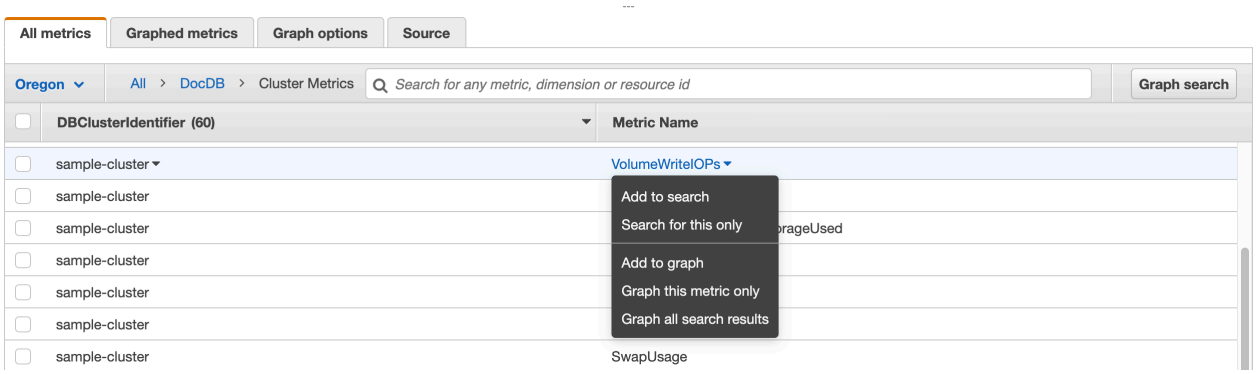
Para visualizar CloudWatch as métricas usando o CloudWatch Management Console, conclua as etapas a seguir.

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB emhttps://console.aws.amazon.com/cloudwatch.
2. No painel de navegação, selecione Métricas. Em seguida, na lista de nomes de serviços, escolha DocDB.
3. Escolha uma dimensão de métrica (por exemplo, Métricas de cluster).
4. A guia Todas as métricas exibe todas as métricas dessa dimensão no DocDB.

    a. Para classificar a tabela, use o cabeçalho da coluna.

    b. Para criar um gráfico de uma métrica, marque a caixa de seleção ao lado da métrica. Para selecionar todas as métricas, marque a caixa de seleção na linha de cabeçalho da tabela.

    c. Para filtrar por métrica, passe o mouse sobre o nome da métrica e selecione a seta suspensa ao lado do nome da métrica. Em seguida, escolha Adicionar à pesquisa, conforme mostrado na imagem a seguir.



Using the AWS CLI

Para visualizar CloudWatch os dados do Amazon DocumentDB, use a CloudWatch `get-metric-statistics` operação com os seguintes parâmetros.

Parâmetros

- **`--namespace`** — Obrigatório. O namespace de serviço cujas métricas do CloudWatch você deseja ver. Para o Amazon DocumentDB, isto deve ser. `AWS/DocDB`
- **`--metric-name`** — Obrigatório. O nome da métrica cujos dados você deseja dados.
- **`--start-time`** — Obrigatório. O timestamp que determina o primeiro ponto de dados a ser retornado.

O valor especificado é inclusivo; os resultados incluem pontos de dados com o timestamp especificado. O carimbo de data/hora deve estar no formato UTC ISO 8601 (por exemplo, 2016-10-03). T23:00:00Z

- **`--end-time`** — Obrigatório. O timestamp que determina o último ponto de dados a ser retornado.

  O valor especificado é inclusivo; os resultados incluem pontos de dados com o timestamp especificado. O carimbo de data/hora deve estar no formato UTC ISO 8601 (por exemplo, 2016-10-03). T23:00:00Z

- **`--period`** — Obrigatório. A granularidade, em segundos, dos pontos de dados retornados. Para métricas com resolução regular, um período pode ser tão curto quanto um minuto (60 segundos) e deve ser um múltiplo de 60. Para métricas de alta resolução coletadas em intervalos menores que um minuto, o período pode ser 1, 5, 10, 30, 60 ou qualquer múltiplo de 60.
- **`--dimensions`** — Opcional. Se a métrica contiver várias dimensões, você deverá incluir um valor para cada dimensão. CloudWatch trata cada combinação exclusiva de dimensões como uma métrica separada. Se uma combinação específica de dimensões não foi publicada, você não poderá recuperar estatísticas para ela. É necessário especificar as mesmas dimensões usadas ao criar as métricas.
- **`--statistics`** — Opcional. As estatísticas da métrica, além do percentil. Para estatísticas de percentil, use `ExtendedStatistics`. Ao chamar `GetMetricStatistics`, você deve especificar `Statistics` ou `ExtendedStatistics`, mas não ambos.

  Valores permitidos:

  - `SampleCount`
  - `Average`
  - `Sum`
  - `Minimum`
  - `Maximum`
- **`--extended-statistics`** — Opcional. As estatísticas de `percentile`. Especifique os valores entre p0.0 e p100. Ao chamar `GetMetricStatistics`, você deve especificar `Statistics` ou `ExtendedStatistics`, mas não ambos.
- **`--unit`** — Opcional. A unidade para uma determinada métrica. As métricas podem ser relatadas em várias unidades. Não fornecer uma unidade resulta em todas as unidades sendo

retornadas. Se você especificar apenas uma unidade que a métrica não reporta, os resultados da chamada serão nulos.

Possíveis valores:

- 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

O exemplo a seguir localiza o `CPUUtilization` máximo para um período de 2 horas, tirando uma amostra a cada 60 segundos.

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante à seguinte.

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

Dimensões do Amazon DocumentDB

As métricas para o Amazon DocumentDB são qualificadas de acordo com os valores de conta ou operação. Você pode usar o CloudWatch console para recuperar dados do Amazon DocumentDB filtrados por qualquer uma das dimensões na tabela a seguir.

Dimensão	Description
`DBClusterIdentifier`	Filtra os dados solicitados para um cluster específico do Amazon DocumentDB.
`DBClusterIdentifier, Role`	Filtra os dados que você solicita para um cluster específico do Amazon DocumentDB, agregando a métrica por função de instância ()WRITER/READER. Por exemplo, é possível agregar métricas para todas as instâncias de READER que pertençam a um cluster.
`DBInstanceIdentifier`	Filtra os dados solicitados para uma instância de banco de dados específica.

Monitoramento de métricas de contadores

Métricas do Opcounter têm um valor diferente de zero (geralmente ~ 50) para clusters ociosos. Isto ocorre porque o Amazon DocumentDB realiza verificações de saúde periódicas, operações internas e tarefas de coleta de métricas.

Monitorar conexões de banco de dados

Quando você visualiza o número de conexões usando comandos do mecanismo de banco de dados `db.runCommand( { serverStatus: 1 })`, como, você pode ver até 10 conexões a mais do que as que você vê CloudWatch. `DatabaseConnections` Isto ocorre porque o Amazon DocumentDB realiza verificações de integridade periódicas e tarefas de coleta de métricas que não são contabilizadas no `DatabaseConnections`. `DatabaseConnections` representa apenas conexões iniciadas pelo cliente.

Registro de chamadas de API do Amazon DocumentDB com AWS CloudTrail

O Amazon DocumentDB (com compatibilidade com o MongoDB) é integrado a um serviço que fornece um registro das ações realizadas por usuários AWS CloudTrail, funções ou um serviço no AWS Amazon DocumentDB (com compatibilidade com o MongoDB). CloudTrail captura todas as chamadas de AWS CLI API para o Amazon DocumentDB como eventos, incluindo chamadas do console do Amazon DocumentDB e de chamadas de código para o SDK do Amazon DocumentDB. Se você criar uma trilha, poderá habilitar a entrega contínua de CloudTrail eventos para um bucket do Amazon S3, incluindo eventos para o Amazon DocumentDB. Se você não configurar uma trilha, ainda poderá ver os eventos mais recentes no CloudTrail console no Histórico de eventos. Usando as informações coletadas por CloudTrail, você pode determinar a solicitação que foi feita ao Amazon DocumentDB (com compatibilidade com o MongoDB), o endereço IP do qual a solicitação foi feita, quem fez a solicitação, quando ela foi feita e outros detalhes.

> **Important**
>
> Para alguns recursos de gerenciamento, o Amazon DocumentDB usa a tecnologia operacional que é compartilhada com o Amazon Relational Database Service (Amazon RDS). As chamadas de console e API do Amazon DocumentDB são registradas como chamadas feitas para a API do Amazon RDS. AWS CLI

Para saber mais sobre isso AWS CloudTrail, consulte o Guia AWS CloudTrail do usuário.

Informações do Amazon DocumentDB em CloudTrail

CloudTrail é ativado no seu Conta da AWS quando você cria a conta. Quando a atividade ocorre no Amazon DocumentDB (com compatibilidade com o MongoDB), essa atividade é registrada em um CloudTrail evento junto com outros AWS eventos de serviço no histórico de eventos. Você pode visualizar, pesquisar e baixar eventos recentes no seu Conta da AWS. Para obter mais informações, consulte Visualização de eventos com histórico de CloudTrail eventos.

Para obter um registro contínuo dos eventos em seu Conta da AWS, incluindo eventos do Amazon DocumentDB (com compatibilidade com o MongoDB), crie uma trilha. Uma trilha permite CloudTrail entregar arquivos de log para um bucket do Amazon S3. Por padrão, quando você cria uma trilha no console, ela é aplicada a todas as Regiões da AWS. A trilha registra eventos de todas as regiões na partição da AWS e entrega os arquivos de log no bucket do Amazon S3 que você especifica. Além disso, você pode configurar outros AWS serviços para analisar e agir com base nos dados de eventos coletados nos CloudTrail registros. Para obter mais informações, consulte os seguintes tópicos no Guia do usuário do AWS CloudTrail :

- Visão geral da criação de uma trilha
- CloudTrail Serviços e integrações compatíveis
- Configurando notificações do Amazon SNS para CloudTrail
- Recebendo arquivos de CloudTrail log de várias regiões
- Recebendo arquivos de CloudTrail log de várias contas

Cada entrada de log ou evento inclui informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar o seguinte:

- Se a solicitação foi feita com credenciais de usuário da raiz ou do .
- Se a solicitação foi feita com credenciais de segurança temporárias de um perfil ou de um usuário federado.
- Se a solicitação foi feita por outro AWS serviço.

Para obter mais informações, consulte Elemento userIdentity do CloudTrail.

Definir o perfil das operações do Amazon DocumentDB

É possível usar o criador de perfil no Amazon DocumentDB (compatível com MongoDB) para registrar o tempo de execução e os detalhes das operações que foram realizadas no seu cluster. O criador de perfil é útil para monitorar as operações mais lentas em seu cluster para ajudar você a melhorar o desempenho de consultas individuais e o desempenho geral do cluster.

Por padrão, o recurso profiler está desabilitado. Quando ativado, o criador de perfil registra operações que estão demorando mais do que um valor limite definido pelo cliente (por exemplo, 100 ms) no Amazon Logs. CloudWatch Os detalhes registrados incluem o comando perfilado, a hora, o resumo do plano e os metadados do cliente. Depois que as operações forem registradas no CloudWatch Logs, você poderá usar o CloudWatch Logs Insights para analisar, monitorar e arquivar seus dados de criação de perfil do Amazon DocumentDB. Consultas comuns são fornecidas na seção Consultas comuns.

Quando habilitado, o profiler usa recursos adicionais em seu cluster. Comece com um valor limite alto (por exemplo, 500 ms) e diminua gradualmente o valor para identificar operações lentas. Começar com um valor limite de 50 ms pode causar problemas de desempenho no cluster para aplicações de alto throughput. O profiler é habilitado no nível do cluster e funciona em todas as instâncias e bancos de dados em um cluster. O Amazon DocumentDB registra as operações no Amazon CloudWatch Logs da melhor forma possível.

Embora o Amazon DocumentDB não imponha nenhum custo adicional para habilitar o criador de perfil, você pagará as taxas padrão pelo uso de registros. CloudWatch Para obter informações sobre CloudWatch os preços do Logs, consulte os CloudWatch preços da Amazon.

Tópicos

- Operações compatíveis
- Limitações
- Ativar o profiler do Amazon DocumentDB
- Desativar o profiler do Amazon DocumentDB
- Desativar a exportação de logs do profiler
- Acessar seus logs do profiler do Amazon DocumentDB
- Consultas comuns

Operações compatíveis

O profiler do Amazon DocumentDB suporta as seguintes operações:

- `aggregate`
- `count`
- `delete`
- `distinct`
- `find` (OP_QUERY e comando)
- `findAndModify`
- `insert`
- `update`

Limitações

O criador de perfil de consulta lenta só poderá emitir registros do criador de perfil se todo o conjunto de resultados da consulta couber em um lote e se o conjunto de resultados estiver abaixo de 16 MB (tamanho máximo do BSON). Conjuntos de resultados maiores que 16 MB são divididos automaticamente em vários lotes.

A maioria dos drivers ou shells pode definir um pequeno tamanho de lote padrão. Você pode especificar o tamanho do lote como parte da sua consulta. Para capturar registros de consultas lentos, use um tamanho de lote que exceda o tamanho do conjunto de resultados esperado. Se você não tiver certeza do tamanho do conjunto de resultados ou se ele variar, defina o tamanho do lote para um número grande (por exemplo, 100k).

No entanto, usar um lote maior significa que mais resultados precisarão ser recuperados do banco de dados antes que uma resposta seja enviada ao cliente. Para algumas consultas, isso pode criar maiores atrasos antes de você obter resultados. Se você não planeja consumir todo o conjunto de resultados, é possível que você gaste mais I/Os para processar a consulta e descartar o resultado.

Ativar o profiler do Amazon DocumentDB

A habilitação do profiler em um cluster é um processo de três etapas. Certifique-se de que todas as etapas sejam concluídas, ou os registros de criação de perfil não serão enviados para CloudWatch

os Registros. O profiler é definido no nível do cluster e executado em todos os bancos de dados e instâncias do cluster.

Como habilitar o profiler em um cluster

1. Como você não pode modificar um grupo de parâmetros de cluster padrão, verifique se tem um grupo de parâmetros de cluster personalizado disponível. Para obter mais informações, consulte Criando grupos de parâmetros de cluster do Amazon DocumentDB.
2. Ao usar um grupo de parâmetros de cluster personalizado disponível, modifique os seguintes parâmetros: `profiler`, `profiler_threshold_ms` e `profiler_sampling_rate`. Para obter mais informações, consulte Modificando grupos de parâmetros de cluster do Amazon DocumentDB.
3. Crie ou modifique seu cluster para usar o grupo de parâmetros de cluster personalizado e permitir a exportação de `profiler` CloudWatch registros para Logs.

As seções a seguir mostram como implementar essas etapas usando o Console de gerenciamento da AWS e o AWS Command Line Interface (AWS CLI).

Using the Console de gerenciamento da AWS

1. Antes de começar, crie um cluster do Amazon DocumentDB e um grupo de parâmetros de cluster personalizado se você ainda não tiver um. Para obter mais informações, consulte Criando grupos de parâmetros de cluster do Amazon DocumentDB e Criar um cluster do Amazon DocumentDB.
2. Ao usar um grupo de parâmetros de cluster personalizado disponível, modifique os seguintes parâmetros. Para obter mais informações, consulte Modificando grupos de parâmetros de cluster do Amazon DocumentDB.

   - `profiler` - Habilita ou desabilita a criação de perfis de consulta. Os valores permitidos são `enabled` e `disabled`. O valor padrão é `disabled`. Para habilitar a criação de perfis, defina o valor como `enabled`.
   - `profiler_threshold_ms`— Quando `profiler` definido como`enabled`, todos os comandos que estão demorando mais do que `profiler_threshold_ms` estão registrados. CloudWatch Os valores permitidos são `[50-INT_MAX]`. O valor padrão é `100`.
   - `profiler_sampling_rate` - a parcela das operações lentas que devem ser perfiladas ou registradas em log. Os valores permitidos são `[0.0-1.0]`. O valor padrão é `1.0`.

3. Modifique seu cluster para usar o grupo de parâmetros de cluster personalizado e defina as exportações de log do profiler para publicar na Amazon CloudWatch.

   a. No painel de navegação, escolha Clusters para adicionar o grupo de parâmetros personalizado a um cluster.

   b. Selecione o botão à esquerda do nome do cluster que deseja associar ao grupo de parâmetros. Selecione Actions e Modify para modificar seu cluster.

   c. Em Cluster options, escolha o grupo de parâmetros personalizado na etapa acima para adicioná-lo ao cluster.

   d. Em Exportações de registros, selecione Registros do Profiler para publicar na Amazon CloudWatch.

   e. Escolha Continue para exibir um resumo das modificações.

   f. Depois de verificar suas alterações, é possível aplicá-las imediatamente ou durante a próxima janela de manutenção em Scheduling of modifications.

   g. Escolha Modify cluster para atualizar seu cluster com o novo grupo de parâmetros.

Using the AWS CLI

O procedimento a seguir habilita o profiler em todas as operações compatíveis para o cluster `sample-cluster`.

1. Antes de começar, verifique se você tem um grupo de parâmetros de cluster personalizado disponível executando o seguinte comando e revisando a saída de um grupo de parâmetros de cluster que não tem `default` no nome e tem `docdb3.6` como família do grupo de parâmetros. Se você não tiver um grupo de parâmetros de cluster não padrão, consulte Criando grupos de parâmetros de cluster do Amazon DocumentDB.

   ```
   aws docdb describe-db-cluster-parameter-groups \
       --query 'DBClusterParameterGroups[*].
   [DBClusterParameterGroupName,DBParameterGroupFamily]'
   ```

   Na saída a seguir, somente `sample-parameter-group` atende ambos os critérios.

   ```
   [
          [
              "default.docdb3.6",
              "docdb3.6"
   ```

```
        ],
        [
            "sample-parameter-group",
            "docdb3.6"
        ]
]
```

2. Usando seu grupo de parâmetros de cluster personalizado, modifique os seguintes parâmetros:

   - `profiler` - Habilita ou desabilita a criação de perfis de consulta. Os valores permitidos são `enabled` e `disabled`. O valor padrão é `disabled`. Para habilitar a criação de perfis, defina o valor como `enabled`.
   - `profiler_threshold_ms`— Quando `profiler` definido como`enabled`, todos os comandos demoram mais do que `profiler_threshold_ms` os registrados. CloudWatch Os valores permitidos são `[50-INT_MAX]`. O valor padrão é `100`.
   - `profiler_sampling_rate` - a parcela das operações lentas que devem ser perfiladas ou registradas em log. Os valores permitidos são `[0.0-1.0]`. O valor padrão é `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 o cluster do Amazon DocumentDB para que ele use o grupo de parâmetro de cluster personalizado `sample-parameter-group` da etapa anterior e defina o parâmetro `--enable-cloudwatch-logs-exports` como `profiler`.

   O código a seguir modifica o cluster `sample-cluster` para usar o `sample-parameter-group` da etapa anterior e adiciona `profiler` às exportações de CloudWatch registros ativadas.

```
aws docdb modify-db-cluster \
       --db-cluster-identifier sample-cluster \
       --db-cluster-parameter-group-name sample-parameter-group \
       --cloudwatch-logs-export-configuration '{"EnableLogTypes":["profiler"]}'
```

A saída dessa operação é semelhante à seguinte.

```
{
    "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
    }
}
```

Desativar o profiler do Amazon DocumentDB

Para desativar o criador de perfil, você desativa o `profiler` parâmetro e a exportação dos `profiler` registros para os CloudWatch registros.

Desabilitar o profiler

Você pode desativar o `profiler` parâmetro usando o Console de gerenciamento da AWS ou AWS CLI, da seguinte forma.

Using the Console de gerenciamento da AWS

O procedimento a seguir usa o Console de gerenciamento da AWS para desativar o Amazon DocumentDB`profiler`.

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha grupos de parâmetros. Selecione o nome do grupo de parâmetros de cluster ao qual você deseja desabilitar o profiler.
3. Na página Parâmetros de cluster resultante, selecione o botão à esquerda do parâmetro `profiler` e escolha Editar.

4. Na caixa de diálogo Modificar profiler escolha `disabled` na lista.
5. Escolha Modificar parâmetro de cluster.

Using the AWS CLI

Para desabilitar o `profiler` em um cluster usando a AWS CLI, modifique o cluster da seguinte forma.

```
aws docdb modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name sample-parameter-group \
    --parameters
 ParameterName=profiler,ParameterValue=disabled,ApplyMethod=immediate
```

Desativar a exportação de logs do profiler

Você pode desativar a exportação de `profiler` registros para CloudWatch registros usando o Console de gerenciamento da AWS ou AWS CLI, da seguinte forma.

Using the Console de gerenciamento da AWS

O procedimento a seguir usa o Console de gerenciamento da AWS para desativar a exportação de registros do Amazon DocumentDB para o. CloudWatch

1. Abra o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Clusters. Escolha o botão à esquerda do nome do cluster para o qual você deseja desativar a exportação de logs.
3. No menu Ações, escolha Modificar.
4. Role para baixo até a seção Log exports e desmarque Profiler logs.
5. Escolha Continuar.
6. Analise as alterações e escolha quando você deseja que essa mudança seja aplicada ao seu cluster:
   - Aplicar durante a próxima janela de manutenção programada
   - Aplicar imediatamente
7. Escolha Modificar Cluster.

Using the AWS CLI

O código a seguir modifica o cluster `sample-cluster` e desativa os registros do CloudWatch profiler.

Example

Para Linux, macOS ou 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"]}'
```

A saída dessa operação é semelhante à seguinte.

```
{
    "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
    }
}
```

Acessar seus logs do profiler do Amazon DocumentDB

Siga estas etapas para acessar seus registros de perfil na Amazon CloudWatch.

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. Certifique-se de estar na mesma região que seu cluster do Amazon DocumentDB.
3. No painel de navegação, selecione Logs.

4. Para encontrar os logs do profiler do cluster, selecione `/aws/docdb/yourClusterName/profiler` na lista.

   Os logs de perfil para cada uma das suas instâncias estão disponíveis em cada um dos respectivos nomes de instância.

Consultas comuns

Veja as seguintes consultas comuns que você pode usar para analisar seus comandos perfilados. Para obter mais informações sobre o CloudWatch Logs Insights, consulte Análise de dados de registro com o CloudWatch Logs Insights e exemplos de consultas.

Obter as 10 operações mais lentas de uma coleção especificada

```
filter ns="test.foo" | sort millis desc | limit 10
```

Obter todas as operações de atualização em uma coleção que levou mais de 60ms

```
filter millis > 60 and op = "update"
```

Obter as 10 operações mais lentas do último mês

```
sort millis desc | limit 10
```

Obter todas as consultas com um resumo de planos COLLSCAN

```
filter planSummary="COLLSCAN"
```

Monitoramento com o Performance Insights

O Performance Insights adiciona atributos de monitoramento existentes do Amazon DocumentDB para ilustrar o desempenho do cluster e ajudar você a analisar quaisquer problemas que o afetem. Com o painel do Performance Insights, você pode visualizar a carga do banco de dados e filtrá-la por esperas, instruções de consulta, hosts ou aplicação.

> **Note**
>
> O Performance Insights está disponível somente para clusters baseados em instâncias do Amazon DocumentDB 3.6, 4.0, 5.0 e 8.0.

Como isso é útil?

- Visualize a performance do banco de dados — Visualize a carga para determinar quando e onde ela está no banco de dados
- Determine o que está causando a carga no banco de dados: determine quais consultas, hosts e aplicações estão contribuindo para a carga na sua instância
- Determine quando há carga em seu banco de dados — amplie o painel Performance Insights para se concentrar em eventos específicos ou diminua o zoom para observar as tendências em um período maior
- Alerta sobre o carregamento do banco de dados — Acesse novas métricas de carregamento do banco de dados automaticamente, de CloudWatch onde você pode monitorar as métricas de carga do banco de dados junto com outras métricas do Amazon DocumentDB e definir alertas sobre elas

Quais são as limitações do Amazon DocumentDB Performance Insights?

- Os insights de desempenho nas regiões AWS GovCloud (US-East) e AWS GovCloud (US-West) não estão disponíveis
- O Performance Insights for Amazon DocumentDB retém até 7 dias de dados de performance
- Consultas com mais de 1.024 bytes não são agregadas nos Insights de performance

Tópicos

- Conceitos de Performance Insights
- Ativar e desativar o Performance Insights
- Configurar políticas de acesso para o Performance Insights
- Análise de métricas usando o painel do Performance Insights
- Recuperar métricas com a API do Performance Insights
- CloudWatch Métricas da Amazon para insights de desempenho
- Métricas de contadores do Performance Insights

Conceitos de Performance Insights

Tópicos

- Média de sessões ativas
- Dimensões
- Máx. vCPU

Média de sessões ativas

Database load (DB load) (Carga do banco de dados) mede o nível de atividade no seu banco de dados. A métrica chave do Performance Insights é `DB Load`, que é coletada a cada segundo. A unidade para a métrica `DBLoad` é a Average Active Sessions (AAS) para uma instância do Amazon DocumentDB.

Uma sessão ativa é uma conexão que enviou trabalho para a instância do Amazon DocumentDB e está aguardando uma resposta. Por exemplo, se você enviar uma consulta a uma instância do Amazon DocumentDB, a sessão do banco de dados estará ativa enquanto a instância estiver processando a consulta.

Para obter a média de sessões ativas, o Performance Insights faz uma amostra do número de sessões executando simultaneamente uma consulta. O AAS é o número total de sessões divididas pelo número total de amostras. A tabela a seguir mostra 5 amostras consecutivas de uma consulta em execução.

Amostra	Número de sessões que executam a consulta	AAS	Cálculo
1	2	2	2 sessões/1 amostra
2	0	1	2 sessões/2 amostras
3	4	2	6 sessões/3 amostras
4	0	1.5	6 sessões/4 amostras
5	4	2	10 sessões/5 amostras

No exemplo anterior, a carga do banco de dados para o intervalo de tempo 1-5 é 2 AAS. Um aumento na carga do banco de dados significa que, em média, mais sessões estão sendo executadas no banco de dados.

Dimensões

A métrica `DB Load` é diferente das outras métricas da série temporal, pois você pode fragmentá-la em subcomponentes chamados de dimensões. É possível pensar em dimensões como categorias para as diferentes características da métrica `DB Load`. Quando você está diagnosticando problemas de performance, as dimensões mais úteis são estados de espera e consulta principal.

estados de espera

Um evento de espera faz com que uma instrução de consulta aguarde que um evento específico aconteça antes que possa continuar a execução. Por exemplo, a execução da instrução de consulta pode aguardar até que um recurso bloqueado seja desbloqueado. Ao combinar `DB Load` com estados de espera, é possível obter uma imagem completa do estado da sessão. Aqui estão vários estados de espera do Amazon DocumentDB:

Estado de espera do Amazon DocumentDB	Descrição do estado de espera
Latch	O estado de espera Latch ocorre quando a sessão está aguardando para paginar o buffer pool. A entrada e saída frequentes do buffer pool podem ocorrer com mais frequência quando há consultas grandes e frequentes sendo processadas pelo sistema, varredura s de coleção ou quando o buffer pool é muito pequeno para lidar com o conjunto de trabalho.
CPU	O estado de espera da CPU ocorre quando a sessão está aguardando a CPU.
CollectionLock	O estado de CollectionLock espera ocorre quando a sessão está aguardando a aquisição de um bloqueio na coleção. Esses eventos ocorrem quando há operações de DDL na coleção.

Estado de espera do Amazon DocumentDB	Descrição do estado de espera
DocumentLock	O estado de DocumentLock espera ocorre quando a sessão está aguardando a aquisição de um bloqueio em um documento. Um alto número de gravações simultâneas no mesmo documento contribuirá para mais estados de DocumentLock espera nesse documento.
SystemLock	O estado de SystemLock espera ocorre quando a sessão está aguardando no sistema. Isso pode ocorrer quando há consultas frequentes de longa duração, transações de longa duração ou alta simultaneidade no sistema.
IO	O estado de espera IO ocorre quando a sessão está aguardando pela conclusão de IO.
BufferLock	O estado de BufferLock espera ocorre quando a sessão está esperando para adquirir um bloqueio em uma página compartilhada no buffer. BufferLockos estados de espera podem ser prolongados se outros processos estiverem mantendo os cursores abertos nas páginas solicitadas.
LowMemThrottle	O estado de LowMemThrottle espera ocorre quando a sessão está aguardando devido à forte pressão de memória na instância do Amazon DocumentDB. Se esse estado persistir por muito tempo, considere escalar a instância para fornecer memória adicional. Para obter mais informações, consulte Regulador de recursos.

Estado de espera do Amazon DocumentDB	Descrição do estado de espera
BackgroundActivity	O estado de BackgroundActivity espera ocorre quando a sessão está aguardando processos internos do sistema.
Outros	O estado de espera Outros é um estado de espera interno. Se esse estado persistir por muito tempo, considere encerrar essa consulta. Para mais informações, consulte Como faço para localizar e encerrar consultas bloqueadas ou de longa execução?

Principais consultas

Enquanto os eventos de espera mostram gargalos, as principais consultas mostram quais consultas estão contribuindo mais para a carga do banco de dados. Por exemplo, muitas consultas podem estar atualmente em execução no banco de dados, mas uma única consulta pode consumir 99% da carga do banco de dados. Nesse caso, a carga alta pode indicar um problema com a consulta.

Máx. vCPU

No painel, o gráfico Carga de banco de dados coleta, agrega e exibe informações da sessão. Para ver se as sessões ativas estão excedendo o máximo de CPU, observe sua relação com a linha Máx. vCPU. O valor de Máx. vCPU é determinado pelo número de núcleos de vCPU (CPUs virtuais) da instância do Amazon DocumentDB.

Se a carga de banco de dados estiver com frequência acima da linha Máx. vCPU e o estado de espera primário for CPU, isso indicará que a CPU está sobrecarregada. Nesse caso, convém controlar a utilização as conexões com a instância, ajustar todas as consultas com uma alta carga de CPU ou considerar uma classe de instância maior. As instâncias altas e consistentes de qualquer estado de espera indicam que pode haver problemas de gargalos ou de contenção de recursos que você deve resolver. Isso pode ser válido mesmo quando a carga do banco de dados não ultrapassa a linha de Máx. vCPU.

Ativar e desativar o Performance Insights

Para usar o Performance Insights, ative-o em sua instância de banco de dados. É possível desativá-lo mais tarde. Habilitar e desabilitar o Performance Insights não causa tempo de inatividade, reinicialização ou failover.

O agente do Performance Insights consome CPU e memória limitadas no host do banco de dados. Quando a carga do banco de dados é alta, o agente limita o impacto sobre a performance coletando dados com menos frequência.

Habilitar o Performance Insights ao criar um cluster

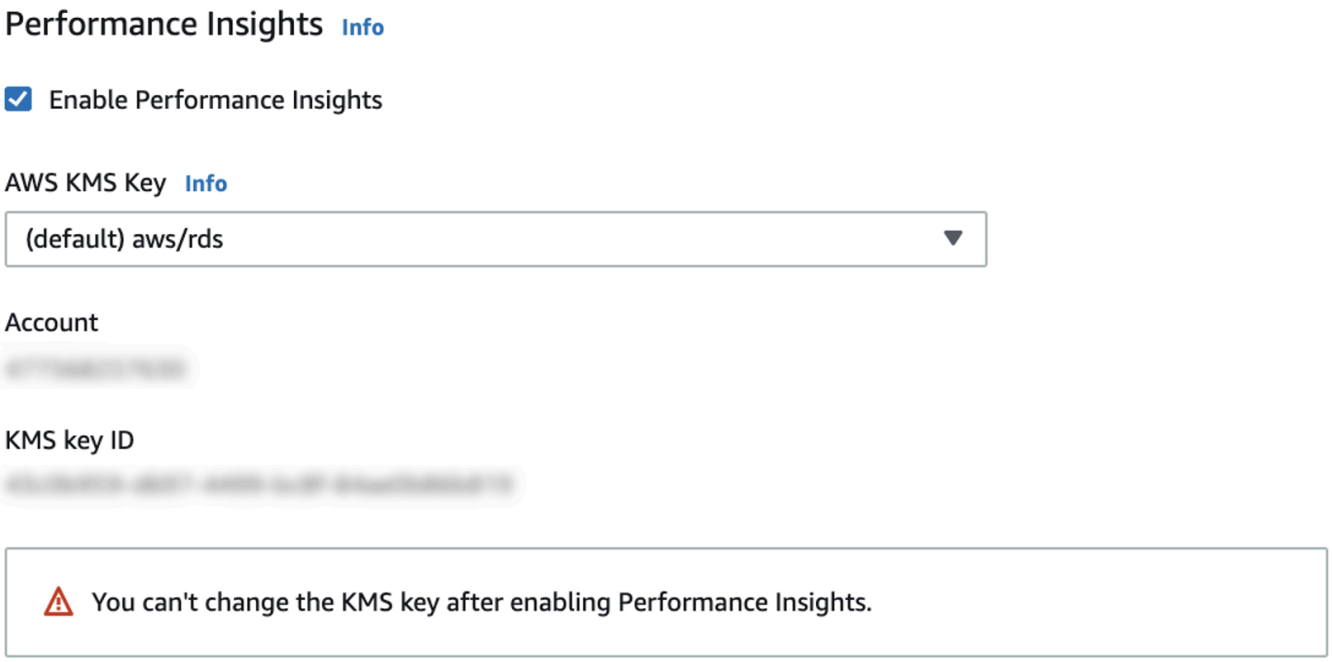
No console, você pode habilitar ou desabilitar o Performance Insights ao criar ou modificar uma nova instância de banco de dados.

Usar o Console de gerenciamento da AWS

No console, você pode ativar o recurso Performance Insights ao criar um cluster do Amazon DocumentDB. Ao criar um novo cluster do Amazon DocumentDB, ative o Performance Insights selecionando Ativar o Performance Insights na seção Performance Insights.

Instruções do console

1. Para criar um cluster, siga as instruções para Criar um cluster do Amazon DocumentDB.
2. Na seção Performance Insights, escolha Habilitar Performance Insights.



> **Note**
>
> O período de retenção de dados do Performance Insights será de sete dias.

AWS KMS chave — Especifique sua chave AWS KMS. O Performance Insights criptografa todos os dados potencialmente confidenciais usando sua AWS KMS chave. Os dados são criptografados em repouso e em trânsito. Para obter mais informações, consulte Configurando uma AWS AWS KMS política para o Performance Insights.

Habilitar e desabilitar ao modificar uma instância

É possível modificar uma instância de banco de dados para habilitar ou desabilitar o Performance Insights usando o console ou AWS CLI.

Using the Console de gerenciamento da AWS

Instruções do console

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. Escolha Clusters.

3. Escolha uma instância de banco de dados e Modificar.
4. Na seção Performance Insights, escolha Habilitar o Performance Insights ou Desabilitar o Performance Insights.

> **Note**
>
> Se você escolher Ativar Performance Insights, poderá especificar sua AWS AWS KMS chave. O Performance Insights criptografa todos os dados potencialmente confidenciais usando sua AWS KMS chave. Os dados são criptografados em repouso e em trânsito. Para obter mais informações, consulte Criptografando o Amazon DocumentDB de dados em repouso.

5. Escolha Continue.
6. Em Programação das modificações, selecione Aplicar imediatamente. Se você escolher Aplicar durante a próxima janela de manutenção agendada, sua instância ignorará essa configuração e habilitará o Performance Insights imediatamente.
7. Escolha Modificar instância.

Using the AWS CLI

Ao usar os `modify-db-instance` AWS AWS CLI comandos `create-db-instance` ou, você pode ativar o Performance Insights `--enable-performance-insights` especificando ou desativá-lo `--no-enable-performance-insights` especificando.

O procedimento a seguir descreve como habilitar ou desabilitar o Performance Insights para uma instância de banco de dados usando a AWS AWS CLI.

AWS AWS CLI instruções

Chame o `modify-db-instance` AWS AWS CLI comando e forneça os seguintes valores:

- `--db-instance-identifer` — o nome da instância de banco de dados.
- `--enable-performance-insights` para habilitar ou `--no-enable-performance-insights` para desabilitar

Example

O exemplo a seguir habilita o Performance Insights para a `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
```

Configurar políticas de acesso para o Performance Insights

Para acessar o Performance Insights, é necessário ter as permissões apropriadas do AWS Identity and Access Management (IAM). Você tem as seguintes opções para conceder acesso:

- Anexe a política gerenciada `AmazonRDSPerformanceInsightsReadOnly` a um conjunto de permissões ou perfil.
- Crie uma política do IAM personalizada e anexe ela a um conjunto de permissões ou perfil.

Além disso, se você especificou uma chave gerenciada pelo cliente quando ativou o Performance Insights, certifique-se de que os usuários em sua conta têm as permissões `kms:Decrypt` e `kms:GenerateDataKey` na chave do KMS.

> **Note**
>
> Para criptografia em repouso com gerenciamento de AWS KMS chaves e grupos de segurança, o Amazon DocumentDB aproveita a tecnologia operacional que é compartilhada com o Amazon RDS. https://aws.amazon.com/rds

Anexando a AmazonRDSPerformanceInsightsReadOnly política a um principal do IAM

`AmazonRDSPerformanceInsightsReadOnly`é uma política AWS gerenciada que concede acesso a todas as operações somente de leitura da API Amazon DocumentDB Performance Insights. Atualmente, todas as operações nesta API são somente leitura. Se você anexar

`AmazonRDSPerformanceInsightsReadOnly` a um conjunto de permissões ou perfil, o destinatário poderá usar o Performance Insights com outros recursos do console.

Criação de uma política de IAM personalizada para o Performance Insights

Para usuários que não têm a política `AmazonRDSPerformanceInsightsReadOnly`, é possível conceder acesso ao Performance Insights criando ou modificando uma política do IAM gerenciada pelo usuário. Quando você anexa a política a um conjunto de permissões ou perfil, o destinatário pode usar o Performance Insights.

Para criar uma política personalizada

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, selecione Políticas.
3. Selecione Criar política.
4. Na página Create Policy (Criar política), escolha a guia JSON.
5. Copie e cole o texto a seguir, *`us-east-1`* substituindo-o pelo nome da sua AWS região e *`111122223333`* pelo número da sua conta de 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. Escolha Review policy (Revisar política).
7. Forneça um nome para a política e, se preferir, uma descrição. Em seguida, escolha Criar política.

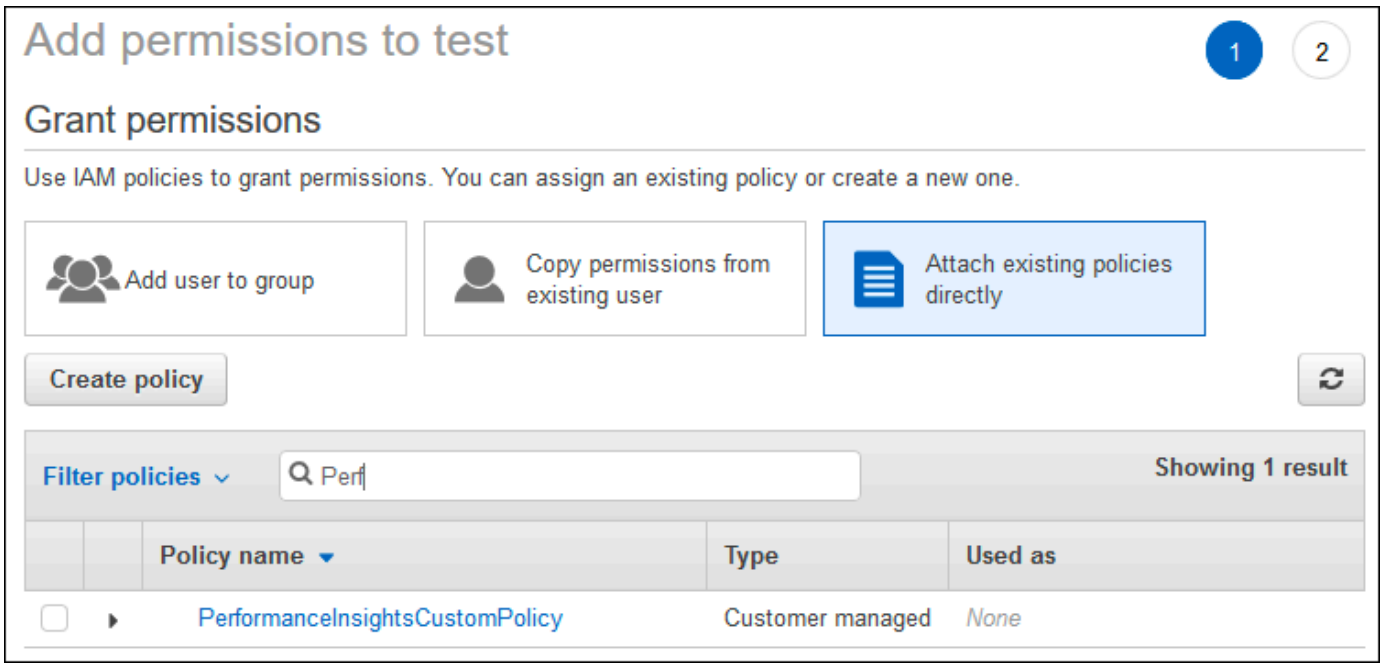
Agora você pode anexar a política a um conjunto de permissões ou perfil. O procedimento a seguir pressupõe que você já tem um usuário disponível para essa finalidade.

Como anexar a política a um usuário

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, escolha Usuários.
3. Escolha um usuário existente na lista.

> **Important**
>
> Para usar o Performance Insights, você deve ter acesso ao Amazon DocumentDB e à política personalizada. Por exemplo, a política AmazonDocDBReadOnlyAccess predefinida fornece acesso somente para leitura à Amazon. DocDB.For Mais informações, consulte Gerenciamento do acesso usando políticas.

4. Na página Summary (Resumo), escolha Add permissions (Adicionar permissões).
5. Escolha Anexar políticas existentes diretamente. Em Pesquisar, digite os primeiros caracteres do nome da sua política, conforme mostrado a seguir.



6. Escolha a política e, em seguida, escolha Próximo: revisar.
7. Escolha Adicionar permissões.

Configurando um AWS KMS política para Performance Insights

O Performance Insights usa um AWS KMS key para criptografar dados confidenciais. Ao habilitar o Performance Insights por meio da API ou do console, você tem as seguintes opções:

- Escolha o padrão Chave gerenciada pela AWS.

  O Amazon DocumentDB usa o Chave gerenciada pela AWS para sua nova instância de banco de dados. O Amazon DocumentDB cria um Chave gerenciada pela AWS para sua AWS conta. Sua

AWS conta tem um diferente Chave gerenciada pela AWS para o Amazon DocumentDB para cada AWS região.

- Escolha uma chave gerenciada pelo cliente.

  Se você especificar uma chave gerenciada pelo cliente, os usuários em sua conta que chamam a API do Performance Insights precisarão das permissões `kms:Decrypt` e `kms:GenerateDataKey` na chave do KMS. Gerencie as permissões da chave KMS usando a política de AWS KMS chaves em vez de usar políticas do IAM. Para obter mais informações, consulte Usar políticas de chaves no AWS KMS.

Example

A política de chave de exemplo a seguir mostra como adicionar instruções à sua política da chaves do KMS. Essas instruções permitem acesso ao Performance Insights. Dependendo de como você usa o AWS KMS, talvez você queira alterar algumas restrições. Antes de adicionar instruções à política, remova todos os comentários.

Análise de métricas usando o painel do Performance Insights

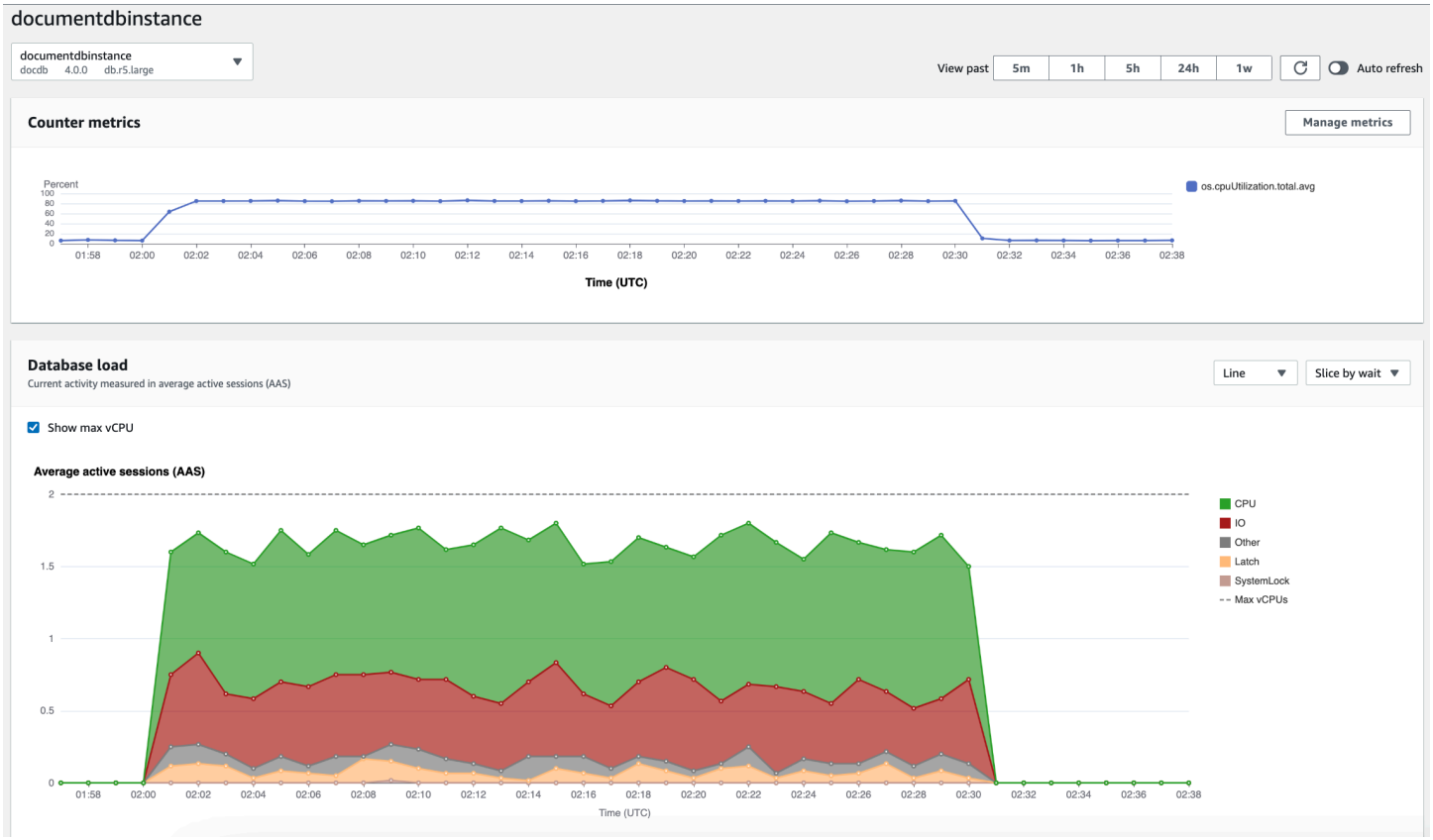
O painel do Performance Insights contém informações de performance do banco de dados para ajudar você a analisar e solucionar problemas de performance. Na página do painel principal, você pode visualizar informações sobre a carga do banco de dados (DB load). É possível “fatiar” a carga de banco de dados por dimensões como eventos de espera ou consulta.

Tópicos

- Visão geral do painel do Performance Insights
- Abrir o painel do Performance Insights
- Analisar a carga do banco de dados por estados de espera
- Visão geral da guia Principais consultas
- Ampliar o gráfico de carga de banco de dados

Visão geral do painel do Performance Insights

O painel é a maneira mais fácil de interagir com o Performance Insights. O exemplo a seguir mostra o painel de uma instância de Amazon DocumentDB. Por padrão, o painel do Performance Insights exibe dados da última hora.



O painel é dividido nas seguintes partes:

1. Counter metrics: mostra dados das métricas de contador de performance específicas.
2. Database load: mostra como a carga de banco de dados se compara à capacidade da instância de banco de dados conforme representada pela linha Máx. vCPU.
3. Dimensões principais – Mostra as principais dimensões que contribuem para a carga do banco de dados. Essas dimensões incluem `waits`, `queries`, `hosts`, `databases` e `applications`.

Tópicos

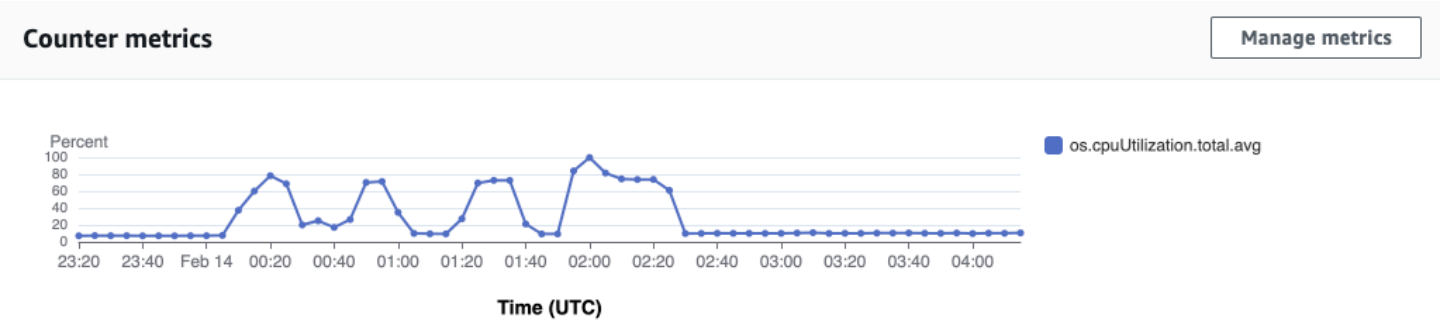
- Gráfico de métricas de contador
- Gráfico de carga do banco de dados
- Tabela Top dimensions

Gráfico de métricas de contador

Com métricas de contador, você pode personalizar o painel do Performance Insights para incluir até 10 gráficos adicionais. Esses gráficos mostram uma seleção de dezenas de métricas de performance

do sistema operacional. É possível correlacionar essas informações à carga do banco de dados para ajudar a identificar e analisar problemas de performance.

O gráfico Métricas de contador exibe dados dos contadores de performance.



Para alterar os contadores de performance, escolha Gerenciar métricas. É possível selecionar várias Métricas de SO, conforme mostrado na captura de tela a seguir. Para ver detalhes de qualquer métrica, passe o mouse sobre o nome da métrica.

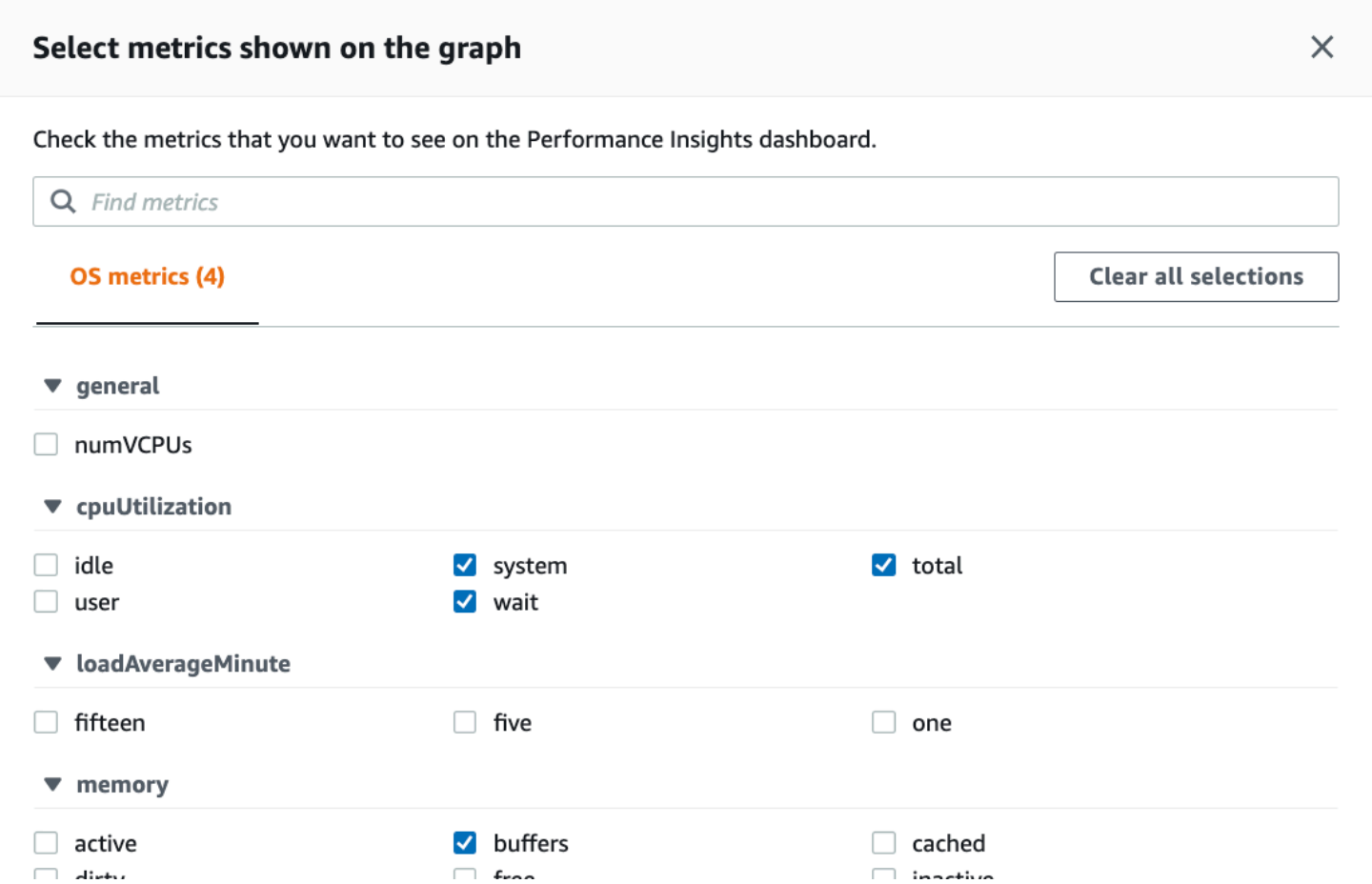


Gráfico de carga do banco de dados

O gráfico Carga do banco de dados mostra como a atividade do banco de dados se compara à capacidade da instância de banco de dados representada pela linha Máximo de vCPU. Por padrão, o gráfico de linhas empilhadas representa a carga do banco de dados como sessões ativas médias por unidade de tempo. A carga do banco de dados é separada (agrupada) por estados de espera.



Carga de banco de dados separada por dimensões

É possível optar por exibir a carga como sessões ativas agrupadas por quaisquer dimensões aceitas. A imagem a seguir mostra as dimensões de uma instância de banco de dados do Amazon DocumentDB.



Detalhes de carga de banco de dados para um item de dimensão

Para ver detalhes sobre um item de carga de banco de dados dentro de uma dimensão, passe o mouse sobre o nome do item. A imagem a seguir mostra detalhes de uma instrução de consulta.



Para ver detalhes de qualquer item do período selecionado na legenda, passe o mouse sobre esse item.

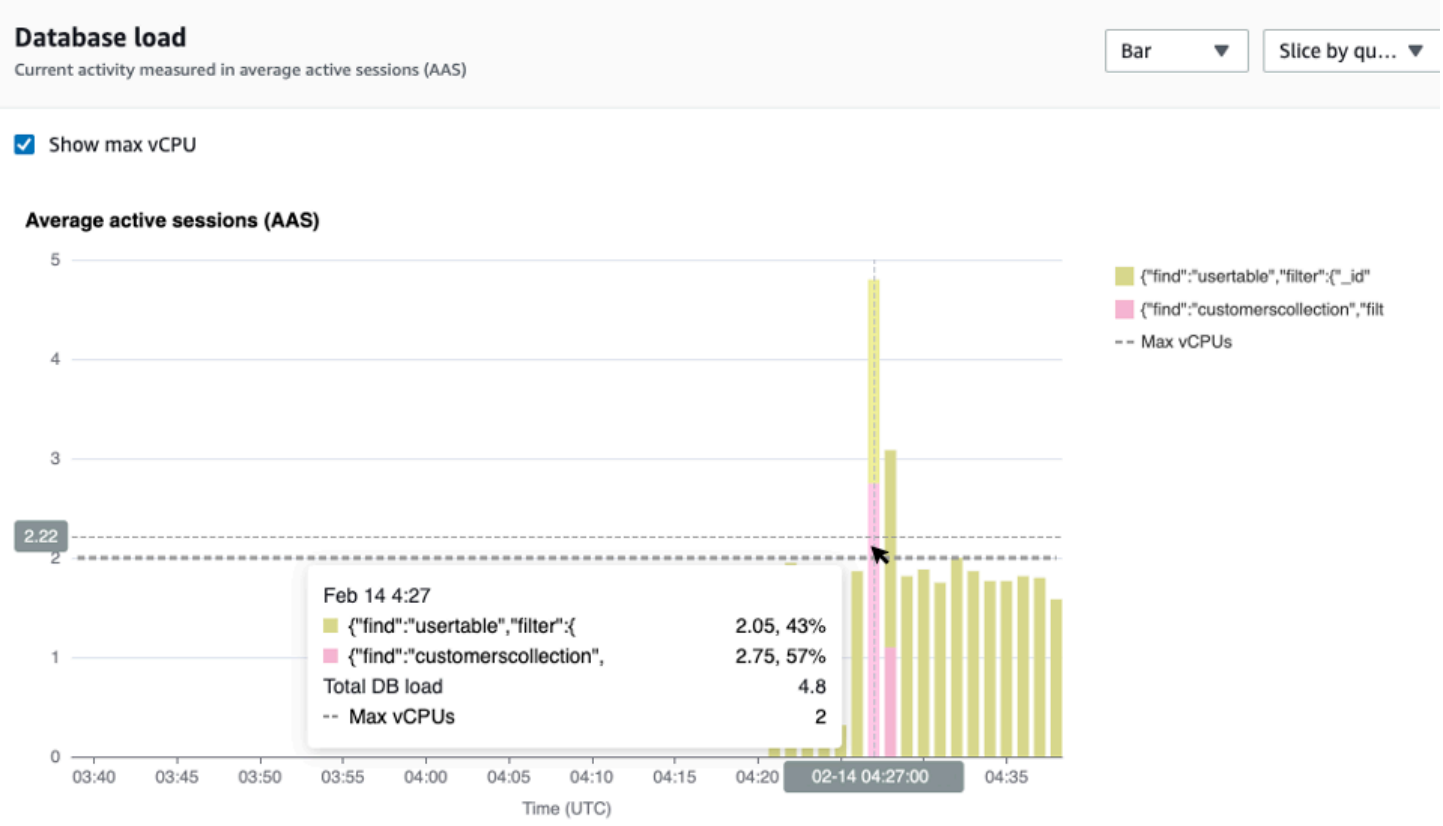
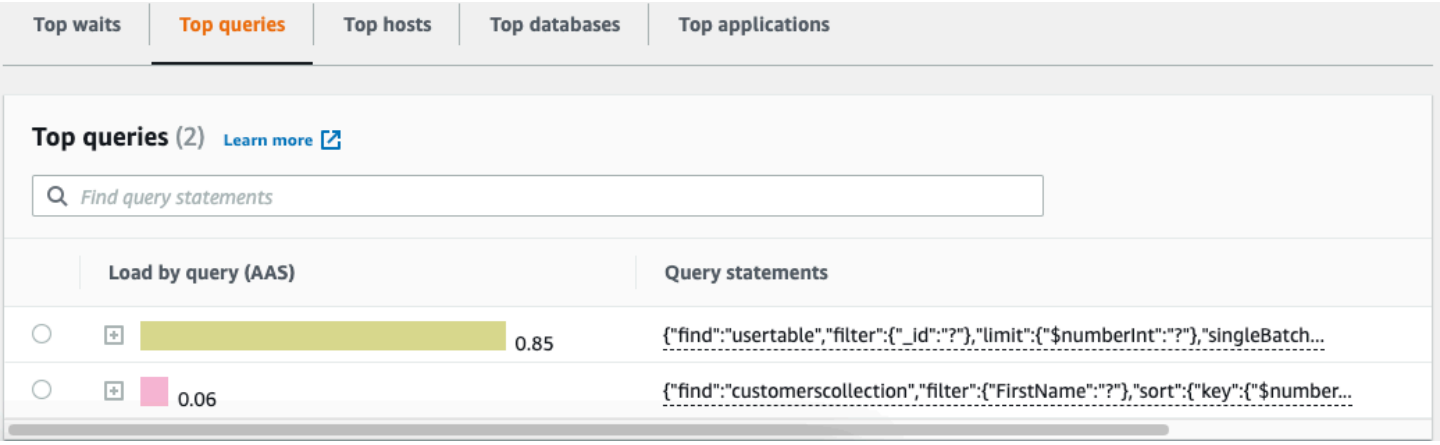


Tabela Top dimensions

A tabela Principais dimensões separa a carga do banco de dados com base em diferentes dimensões. Uma dimensão é uma categoria ou “pedaços” de diferentes características de uma carga de banco de dados. Se a dimensão for consulta, Principais consultas mostrará as instruções SQL que mais contribuem para a carga do banco de dados.

Escolha qualquer uma das guias de dimensão a seguir.



A tabela a seguir fornece uma breve descrição de cada guia.

Descripti on
Esperas principais para o qual o backend do banco de dados está aguardando
Principais instruções consultas de consulta que estão sendo executadas no momento
Hosts principais IP e porta do

Descripti on
cliente conectado
Principais nomes dos bancos de dados ao qual o cliente está conectado
Principais nomes da aplicações aplicação que está conectada ao banco de dados

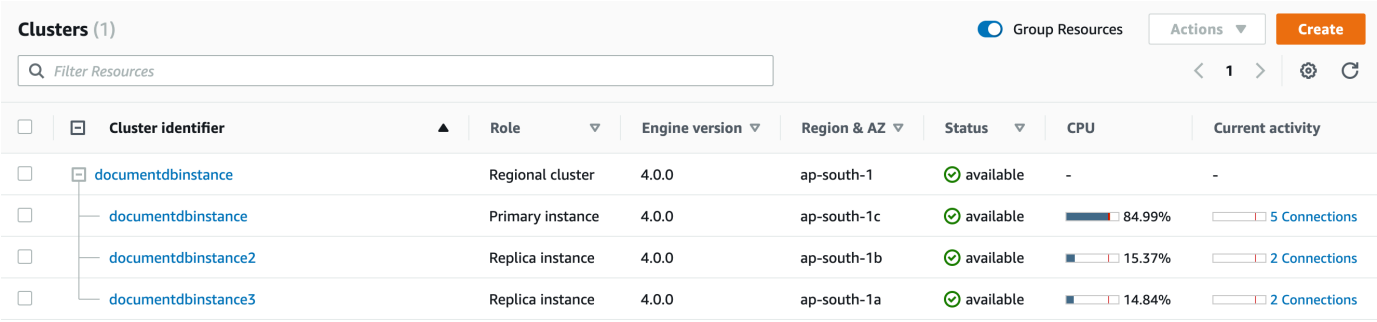
Para aprender a analisar consultas utilizando a guia Principais consultas, consulte Visão geral da guia Principais consultas.

Abrir o painel do Performance Insights

Para visualizar o painel Performance Insights no AWS Console de gerenciamento, use as seguintes etapas:

1. Abra o console do Performance Insights em https://console.aws.amazon.com/docdb/.
2. Escolha uma instância de banco de dados. O painel do Performance Insights é exibido para essa instância dp Amazon DocumentDB.

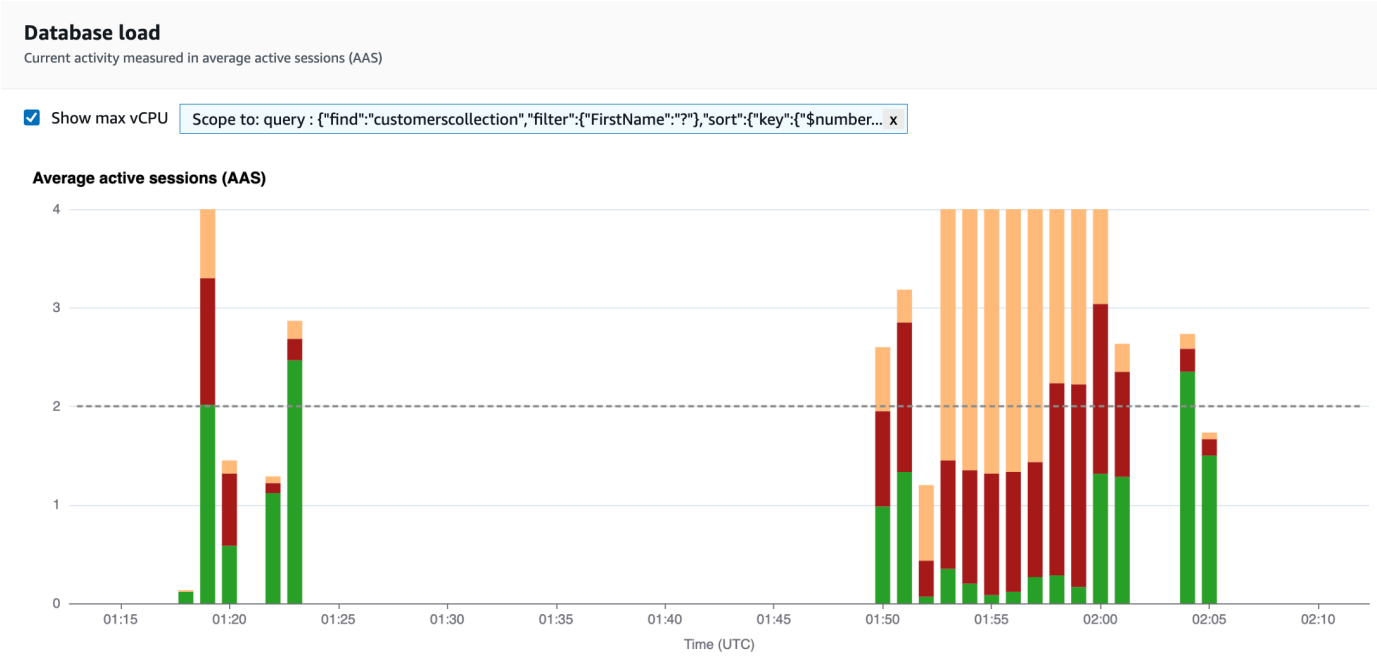
   Para as instâncias do Amazon DocumentDB com o Performance Insights habilitado, você também pode acessar o painel escolhendo o item Sessões na lista de instâncias. Em Atividade atual, o item Sessões mostra a carga de banco de dados em sessões ativas médias nos últimos cinco minutos. A carga é mostrada graficamente por meio de barras. Quando a barra está vazia, a instância está ociosa. À medida que a carga aumenta, a barra é preenchida com a cor azul. Quando a carga ultrapassa o número de CPUs virtuais (vCPUs) na classe da instância, a barra se torna vermelha, indicando um possível gargalo.



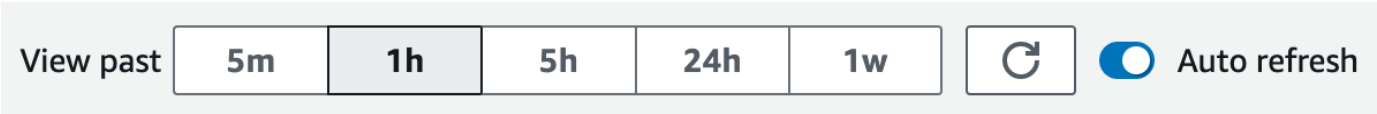
3. (Opcional) Escolha um intervalo de tempo diferente selecionando um botão no canto superior direito. Por exemplo, para alterar o intervalo para 1 hora, selecione 1h.



   Na captura de tela a seguir, o intervalo da carga do banco de dados é de 1 hora.



4. Para atualizar seus dados automaticamente, habilite a Atualização automática.



O painel do Performance Insights é atualizado automaticamente com novos dados. A taxa de atualização depende da quantidade de dados exibida:

- 5 minutos atualiza a cada 5 segundos.
- 1 hora atualiza a cada minuto.
- 5 horas atualiza a cada minuto.
- 24 horas atualiza a cada 5 minutos.
- Uma semana atualiza a cada hora.

Analisar a carga do banco de dados por estados de espera

Se o gráfico Carregamento do banco de dados mostrar um gargalo, será possível descobrir de onde vem essa carga. Para fazer isso, examine a tabela de principais itens de carga abaixo do gráfico Carregamento do banco de dados. Escolha um item específico, como uma consulta ou uma aplicação, para aprofundar neste item e ver detalhes sobre ele.

A carga do banco de dados agrupada por espera e as principais consultas normalmente fornecem mais informações sobre problemas de performance. A carga de banco de dados agrupada por

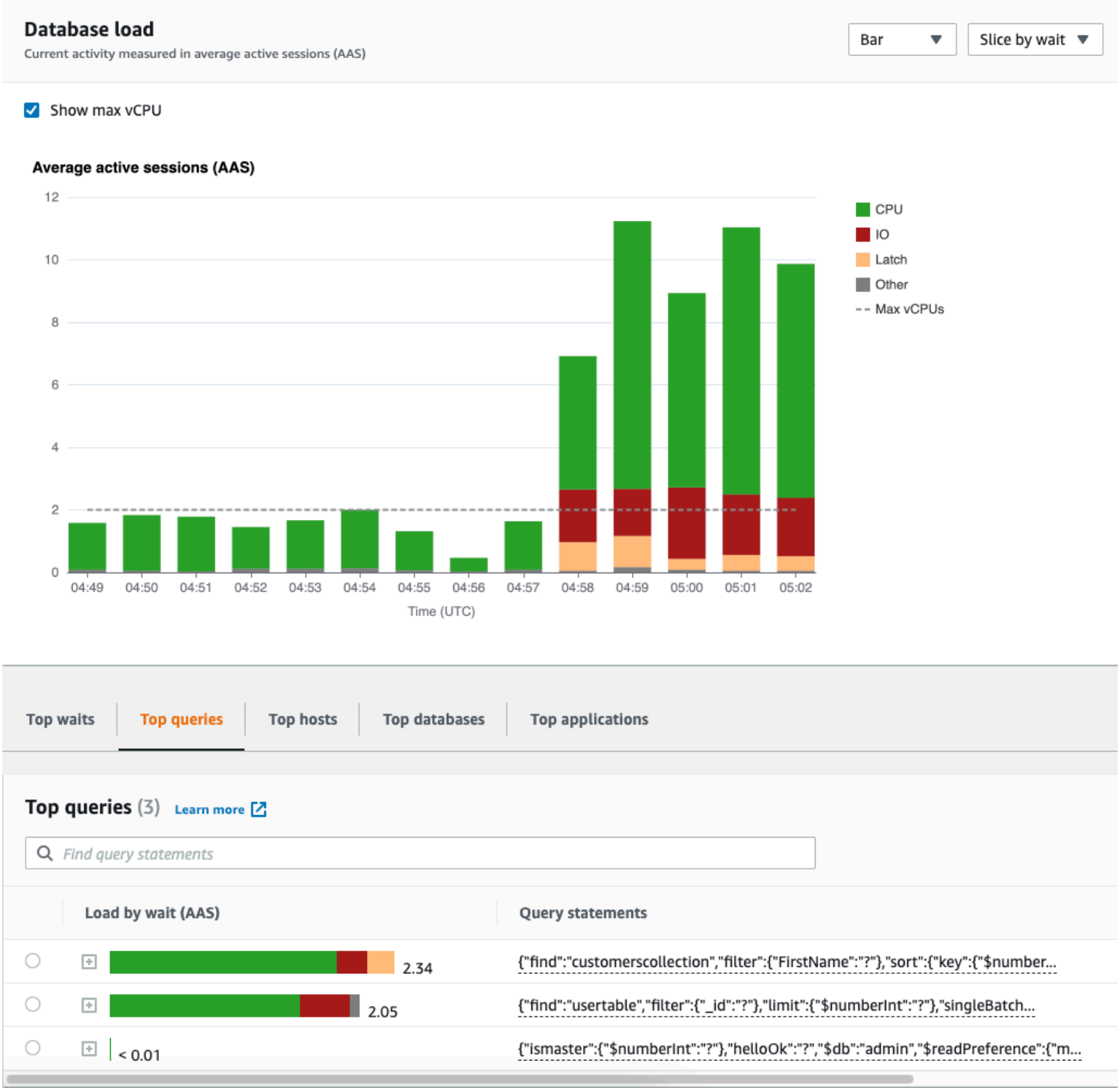
espera mostra se há algum gargalo de recursos ou de concorrências no banco de dados. Nesse caso, a guia Principais consultas da tabela Top Load Items mostra quais consultas estão gerando essa carga.

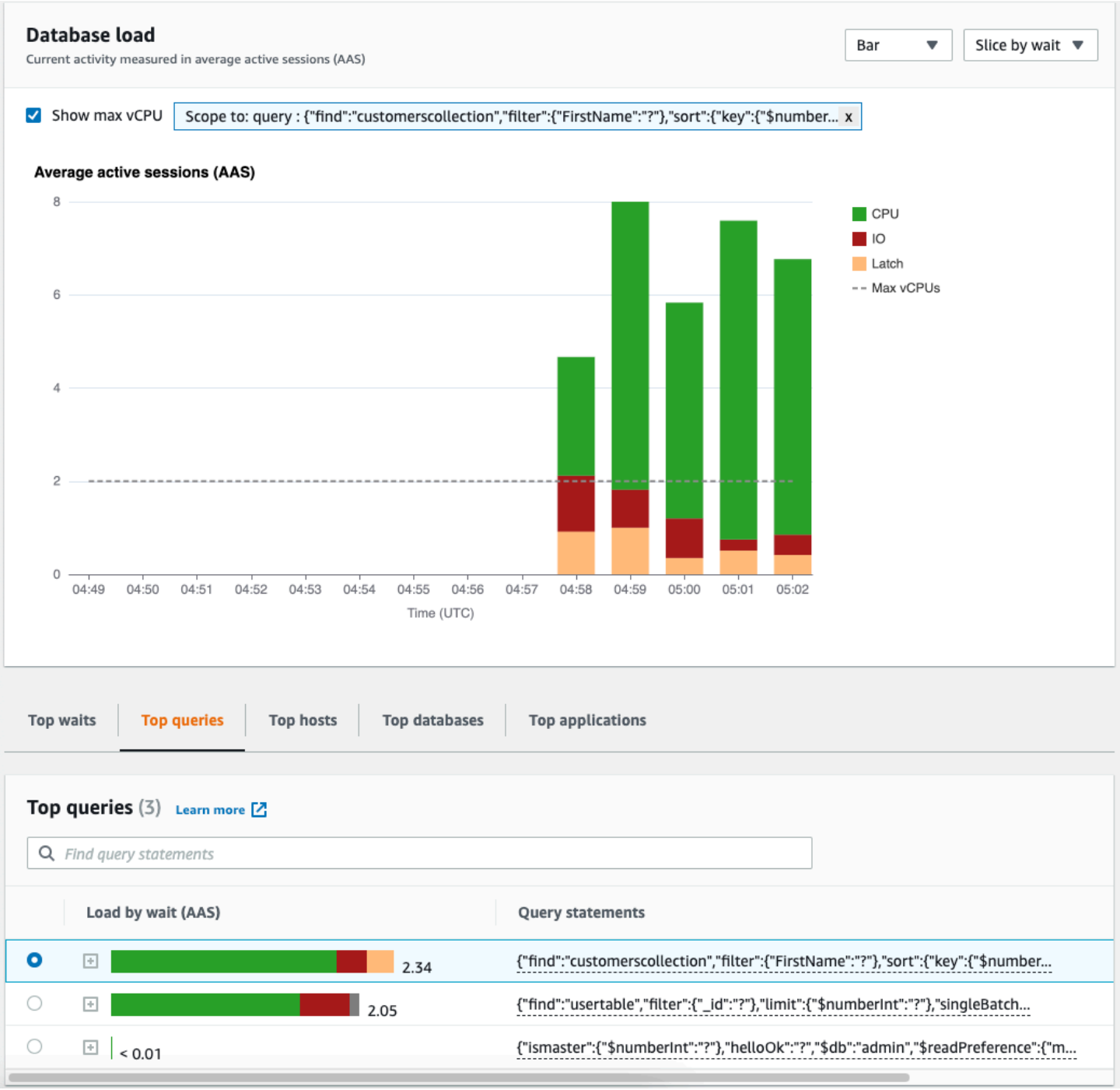
Seu fluxo de trabalho típico para diagnosticar problemas de performance é o seguinte:

1. Analise o gráfico Carregamento do banco de dados e veja se há casos de cargas de banco de dados que estejam ultrapassando a linha Máximo de CPU.
2. Se houver, examine o gráfico Carregamento do banco de dados e identifique quais estados de espera são os principais responsáveis por isso.
3. Identifique as consultas resumidas que estão gerando a carga examinando quais consultas na guia Top queries da tabela Top Load Items estão contribuindo mais para aqueles estados de espera. É possível identificar essas consultas na coluna Carga por espera (AAS).
4. Escolha uma dessas consultas resumidas na guia Top queries para expandi-la e exibir as consultas secundárias que a compõem.

Você também pode ver quais hosts ou aplicações estão contribuindo com a maior carga selecionando Principais hosts ou Principais aplicações, respectivamente. Os nomes das aplicações são especificados na cadeia de conexão com a instância Amazon DocumentDB. `Unknown` indica que o campo da aplicação não foi especificado.

Por exemplo, no painel a seguir, as esperas de CPU compõem a maior parte da carga de banco de dados. Selecionar a consulta principal em Principais consultas definirá o gráfico de carga do banco de dados para se concentrar na maior carga que está sendo contribuída pela consulta selecionada.


Database load
Current activity measured in average active sessions (AAS)
Bar
Slice by wait
Show max vCPU
Average active sessions (AAS)
12
10
8
6
4
2
0
04:49
04:50
04:51
04:52
04:53
04:54
04:55
04:56
04:57
04:58
04:59
05:00
05:01
05:02
Time (UTC)
CPU
IO
Latch
Other
Max vCPUs
Top waits
Top queries
Top hosts
Top databases
Top applications
Top queries (3)
Learn more
Find query statements
Load by wait (AAS)
Query statements
2.34
{"find":"customerscollection","filter":{"FirstName":"?"},"sort":{"key":{"$number...
2.05
{"find":"usertable","filter":{"_id":"?"},"limit":{"$numberInt":"?"},"singleBatch...
< 0.01
{"ismaster":{"$numberInt":"?"},"helloOk":"?","$db":"admin","$readPreference":{"m...



Visão geral da guia Principais consultas

Por padrão, a guia Principais consultas mostra as consultas que mais estão contribuindo para a carga do banco de dados. É possível analisar o texto da consulta para ajudar a ajustar suas consultas.

Tópicos

- Resumos de consultas
- Carga por esperas (AAS)
- Visualizando informações detalhadas da consulta
- Acessando o texto da consulta da instrução
- Visualizando e baixando o texto da consulta de instrução

Resumos de consultas

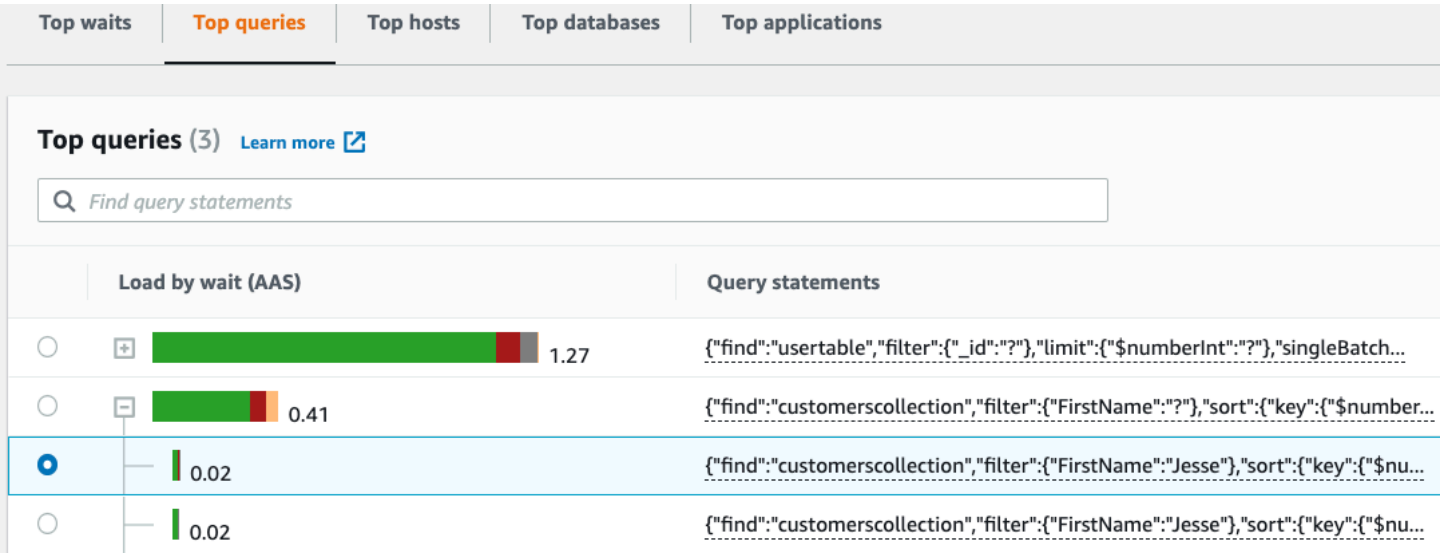
Um resumo de consulta é formado por várias consultas reais com estruturas semelhantes, mas que possivelmente apresentam valores literais diferentes. O resumo substitui valores codificados por um ponto de interrogação. Por exemplo, um resumo de consulta pode ser semelhante a este:

```
{"find":"customerscollection","filter":{"FirstName":"?"},"sort":{"key":
{"$numberInt":"?"}},"limit":{"$numberInt":"?"}}
```

Esse resumo pode incluir as seguintes consultas subordinadas:

```
{"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"}}
```

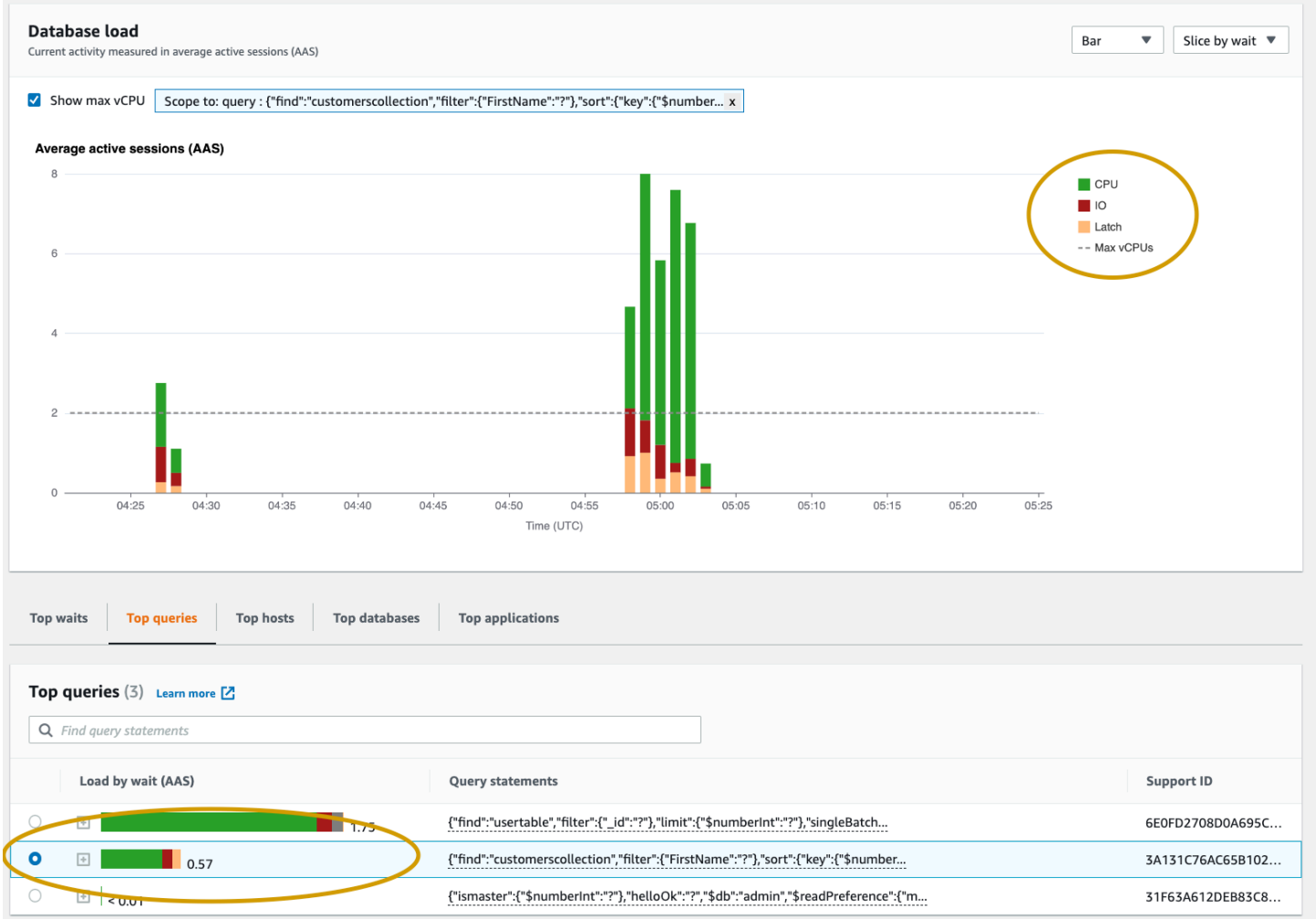
Para ver as instruções consultas literais em um resumo, escolha a consulta e depois o sinal de mais (+). Na captura de tela a seguir, a consulta selecionada é um resumo.



> **Note**
>
> Um resumo de consulta agrupa instruções de consulta semelhantes, mas não edita informações confidenciais.

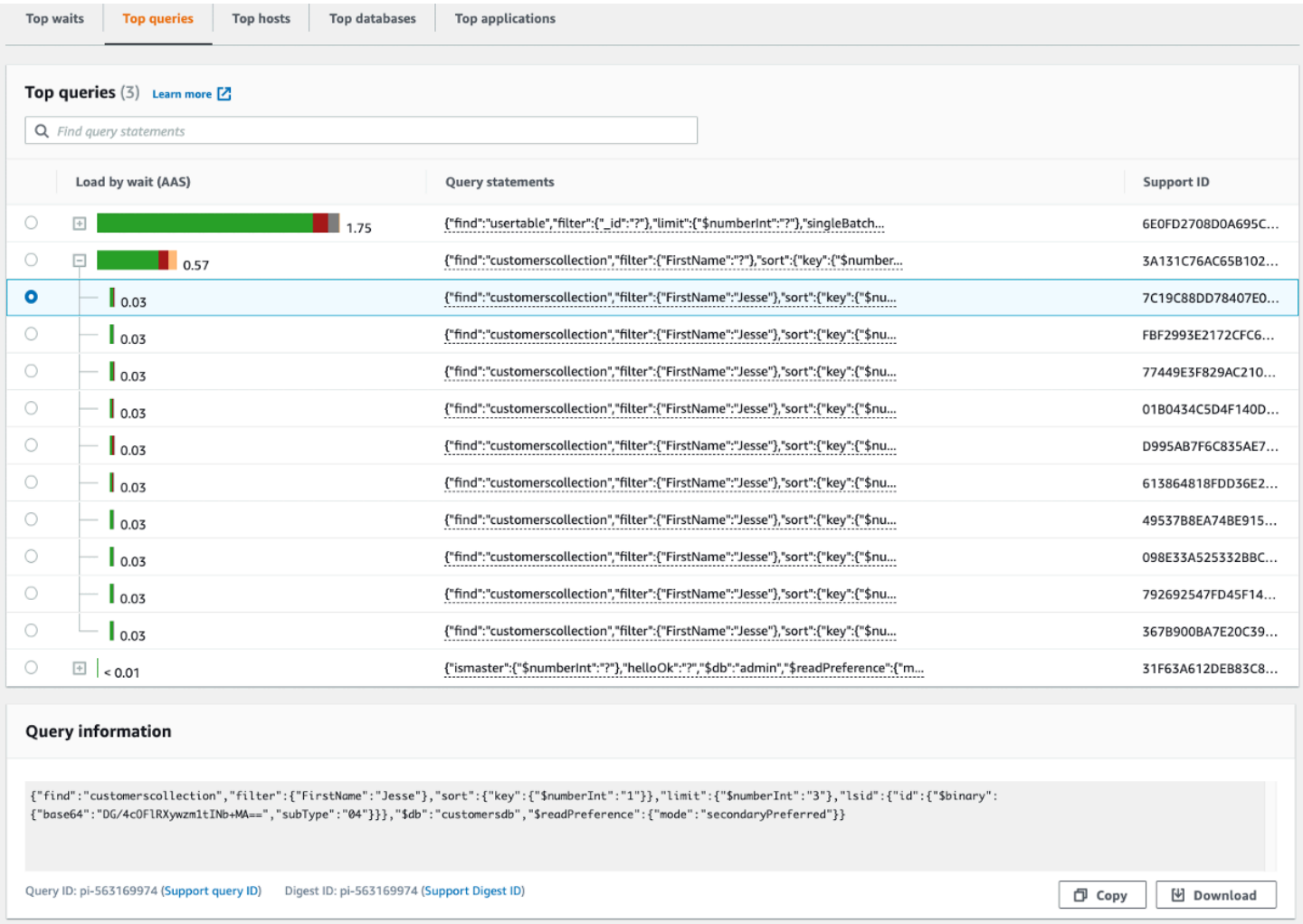
Carga por esperas (AAS)

Em Consultas principais, a coluna Carga por esperas (AAS) mostra o percentual da carga do banco de dados associada a cada item de carga principal. Essa coluna reflete a carga desse item por qualquer agrupamento atualmente selecionado no Gráfico de carga de banco de dados. Por exemplo, é possível agrupar o gráfico Carregamento do banco de dados com base em estados de espera. Nesse caso, a barra Carregamento do banco de dados por esperas é dimensionada, segmentada e codificada por cores para mostrar com quanto de um determinado estado de espera a consulta está contribuindo. Ela também mostra quais estados de espera estão afetando a consulta selecionada.



Visualizando informações detalhadas da consulta

Na tabela Principais consultas, é possível abrir uma instrução de resumo para visualizar suas informações. As informações são exibidas no painel inferior.



Os seguintes tipos de identificadores (IDs) associados a instruções de consulta:

1. ID da consulta de suporte — Um valor de hash do ID da consulta. Esse valor serve somente para referenciar um ID de consulta quando você está trabalhando com o AWS Support. AWS O suporte não tem acesso aos seus IDs de consulta reais e ao texto da consulta.
2. ID de arquivo de resumo de suporte: um valor de hash do ID de arquivo de resumo. Esse valor serve apenas para referenciar um ID de resumo quando você está trabalhando com AWS o Support. AWS O suporte não tem acesso aos seus IDs de resumo reais e ao texto da consulta.

Acessando o texto da consulta da instrução

Por padrão, cada linha na tabela Principais consultas mostra 500 bytes de texto para cada instrução. Quando uma instrução de resumo é maior que 500 bytes, você pode visualizar uma parte maior dela abrindo-a no painel do Performance Insights. Nesse caso, o comprimento máximo para a consulta

mostrada é de 1 KB. Se você vir uma instrução de consulta completa, também poderá escolher Download.

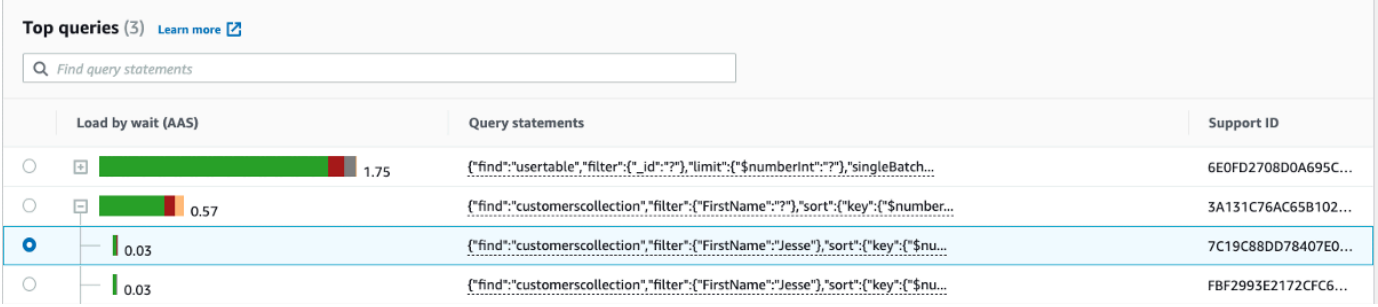
Visualizando e baixando o texto da consulta de instrução

No painel do Performance Insights, é possível visualizar ou baixar o texto da consulta.

Para visualizar mais texto de consulta no painel do Performance Insights

1. Abra o console do Amazon DocumentDB em: https://console.aws.amazon.com/docdb/
2. No painel de navegação, escolha Performance Insights.
3. Escolha uma instância de banco de dados. O painel do Performance Insights será exibido nessa instância de banco de dados.

   Instruções de consulta com texto maior que 500 bytes serão semelhantes à imagem a seguir.



4. Examine a seção de informações de consulta para visualizar mais do texto da consulta.



O painel do Performance Insights pode exibir até 1 KB para cada instrução completa de consulta.

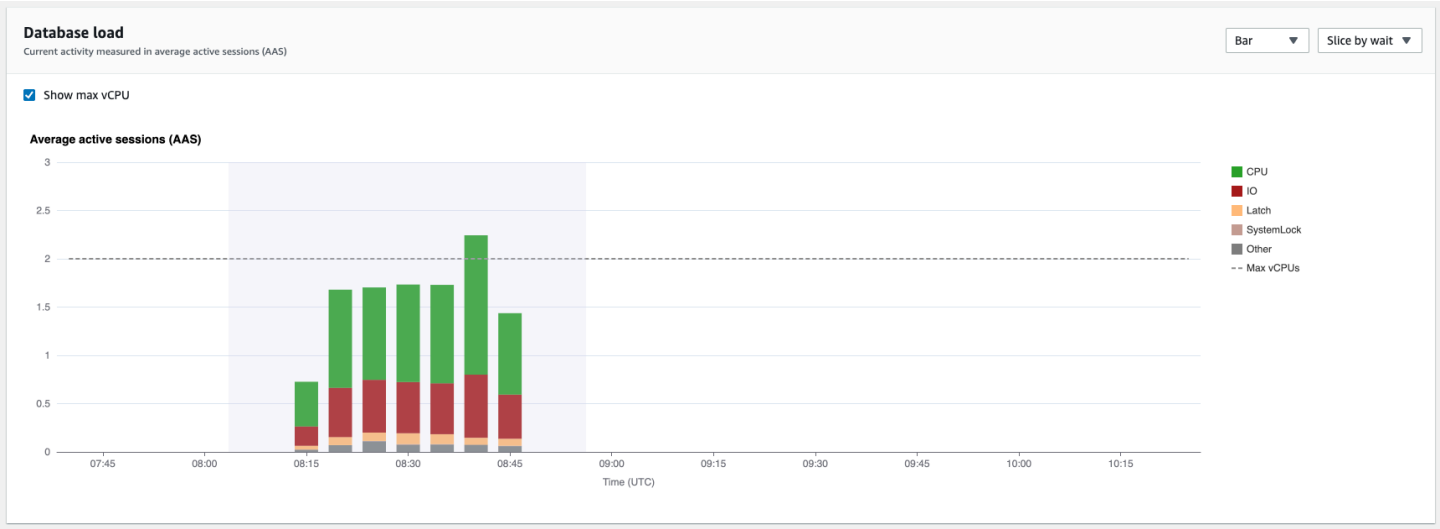
> **Note**
>
> Para copiar ou baixar a instrução de consulta, desabilite bloqueadores de pop-up.

Ampliar o gráfico de carga de banco de dados

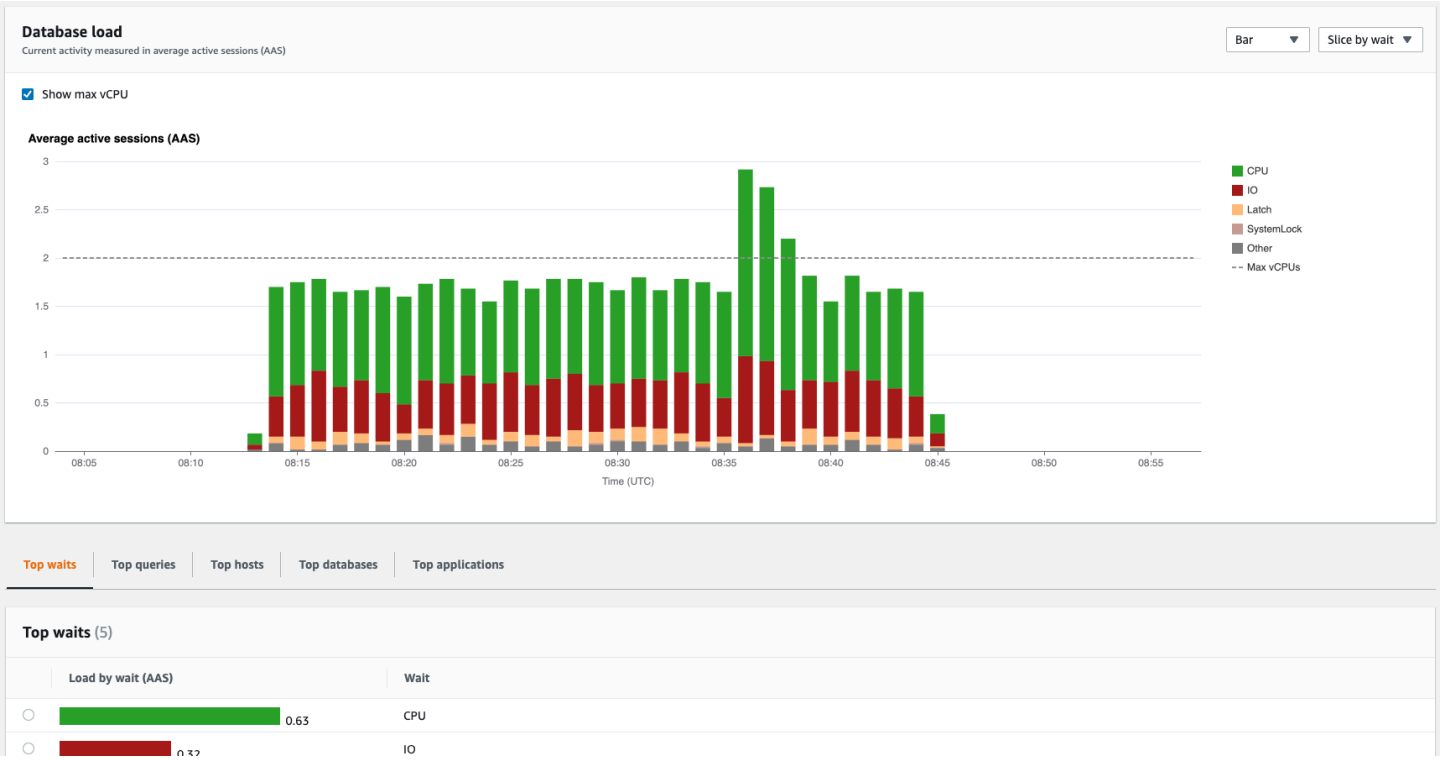
Há outros recursos da interface do usuário do Performance Insights para ajudar você a analisar dados de performance.

Click-and-Drag Ampliar

Na interface do Performance Insights, escolha uma pequena parte do gráfico de carga e amplie os detalhes.



Para ampliar uma parte do gráfico de carga, escolha a hora de início e arraste até o final do período desejado. Quando você faz isso, a área selecionada fica destacada. Ao soltar o mouse, o gráfico de carga amplia a área selecionada e a tabela Itens principais é recalculada.



Recuperar métricas com a API do Performance Insights

Quando o Performance Insights está habilitado, a API fornece visibilidade à performance da instância. O Amazon CloudWatch Logs fornece a fonte confiável de métricas de monitoramento distribuído para serviços. AWS

O Performance Insights oferece uma visão específica do domínio da carga do banco de dados medida como sessões ativas médias (AAS). Essa métrica aparece para os consumidores de API como um conjunto de dados bidimensional de séries temporais. A dimensão de tempo dos dados fornece a carga do banco de dados para cada ponto de tempo no intervalo de tempo consultado. Cada ponto de tempo decompõe a carga geral em relação às dimensões solicitadas, como `Query`, `Wait-state`, `Application` ou `Host`, medidas naquele ponto de tempo.

O Performance Insights do Amazon DocumentDB monitora sua instância de banco de dados do Amazon DocumentDB, para que você possa analisar e solucionar problemas relacionados ao desempenho do seu banco de dados. Uma maneira de visualizar os dados do Performance Insights está no Console de gerenciamento da AWS. O Performance Insights também fornece uma API pública para que você possa consultar seus próprios dados. É possível usar a API para fazer o seguinte:

- Descarregar dados em um banco de dados

- Adicione dados do Performance Insights aos painéis de monitoramento existentes
- Criar ferramentas de monitoramento

Para usar a API do Performance Insights, habilite o Performance Insights em uma das suas instâncias do Amazon DocumentDB. Para obter informações sobre como habilitar o Performance Insights, consulte Ativar e desativar o Performance Insights. Para obter mais informações sobre a API Insights de performance, consulte a Referência de API dos Insights de performance.

A API do Performance Insights fornece as operações a seguir.

Ação do Performance Insights	AWS CLI comando	Description
`DescribeDimensionKeys`	`aws pi describe-dimension-keys`	Recuperar as N principais chaves de dimensão de uma métrica por um período específico.
`GetDimensionKeyDetails`	`aws pi get-dimension-key-details`	Recupera os atributos do grupo de dimensões especificado para uma instância de banco de dados ou fonte de dados. Por exemplo, se você especificar um ID de consulta e se os detalhes da dimensão estiverem disponíveis, `GetDimensionKeyDetails` recuperará o texto completo da dimensão `db.query.statement` associada a esse ID. Essa operação é útil porque `GetResourceMetrics` e `DescribeDimensionKeys` não oferecem suporte à recuperação de texto grande de instrução da consulta.

Ação do Performance Insights	AWS CLI comando	Description
GetResourceMetadata	aws pi get-resource-metadata	Recupere os metadados para diferentes recursos. Por exemplo, os metadados podem indicar que um recurso está ativado ou desativado em uma instância de banco de dados específica.
GetResourceMetrics	aws pi get-resource-metrics	Recupera as métricas do Performance Insights para um conjunto de fontes de dados, ao longo de um período. É possível fornecer grupos de dimensão e dimensões específicos e fornecer critérios de filtragem e agregação para cada grupo.
ListAvailableResourceDimensions	aws pi list-available-resource-dimensions	Recupere as dimensões que podem ser consultadas para cada tipo de métrica especificado em uma instância especificada.
ListAvailableResourceMetrics	aws pi list-available-resource-metrics	Recupere todas as métricas disponíveis dos tipos de métrica especificados que podem ser consultados para uma instância de banco de dados especificada.

Tópicos

- AWS CLI para insights de desempenho
- Recuperar métricas de séries temporais

- AWS CLI exemplos de Performance Insights

AWS CLI para insights de desempenho

É possível visualizar dados do Performance Insights usando o AWS CLI. É possível visualizar a ajuda dos comandos da AWS CLI para o Performance Insights, inserindo o seguinte na linha de comando.

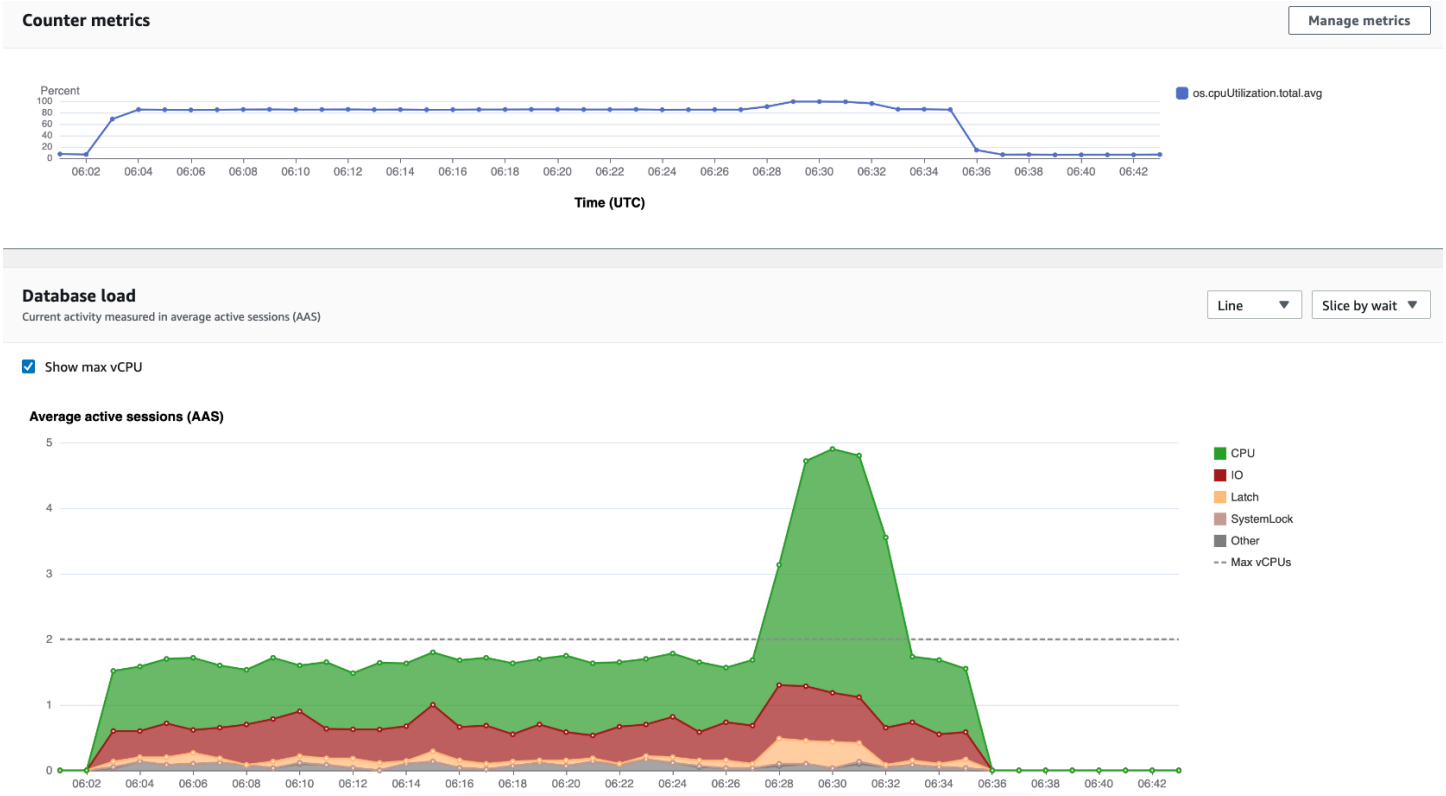
```
aws pi help
```

Se você não tiver o AWS CLI instalado, consulte Instalando a interface de linha de AWS comando no Guia do AWS CLI usuário para obter informações sobre como instalá-la.

Recuperar métricas de séries temporais

A operação `GetResourceMetrics` recupera uma ou mais métricas de séries temporais dos dados do Performance Insights. `GetResourceMetrics` requer uma métrica e um período de tempo e retorna uma resposta com uma lista de pontos de dados.

Por exemplo, os Console de gerenciamento da AWS usos `GetResourceMetrics` para preencher o gráfico Counter Metrics e o gráfico de carregamento do banco de dados, conforme mostrado na imagem a seguir.



Todas as métricas retornadas por `GetResourceMetrics` são métricas de séries temporais padrão, com exceção de `db.load`. Essa métrica é exibida no gráfico Carregamento do banco de dados. A métrica `db.load` é diferente das outras métricas da série temporal, pois você pode fragmentá-la em subcomponentes chamados de dimensões. Na imagem anterior, `db.load` é dividido e agrupado pelos estados de espera que compõem o `db.load`.

> **Note**
>
> `GetResourceMetrics` também pode retornar a métrica `db.sampleload`, mas a métrica `db.load` é apropriada na maioria dos casos.

Para obter informações sobre as métricas de contador retornadas pelo `GetResourceMetrics`, consulte Métricas de contadores do Performance Insights.

Os cálculos a seguir são compatíveis com as métricas:

- Média: o valor médio para a métrica por um período. Adicione `.avg` ao nome da métrica.
- Mínimo: o valor mínimo para a métrica por um período. Adicione `.min` ao nome da métrica.
- Máximo: o valor máximo para a métrica por um período. Adicione `.max` ao nome da métrica.

- Soma: a soma dos valores da métrica por um período. Adicione `.sum` ao nome da métrica.
- Contagem de amostra: o número de vezes que a métrica foi coletada por um período. Adicione `.sample_count` ao nome da métrica.

Por exemplo, considere que uma métrica é coletada por 300 segundos (5 minutos) e que a métrica seja coletada uma vez por minuto. Os valores de cada minuto são 1, 2, 3, 4 e 5. Nesse caso, os seguintes cálculos são retornados:

- Média: 3
- Mínimo: 1
- Máximo: 5
- Soma: 15
- Contagem de amostras: 5

Para obter informações sobre como usar o `get-resource-metrics` AWS CLI comando, consulte `get-resource-metrics`.

Para a opção `--metric-queries`, especifique uma ou mais consultas para as quais deseja obter resultados. Cada consulta consiste em um parâmetro obrigatório `Metric` e opcional `GroupBy` e em parâmetros `Filter`. Veja a seguir um exemplo de uma especificação de opção `--metric-queries`.

```
{
   "Metric": "string",
   "GroupBy": {
     "Group": "string",
     "Dimensions": ["string", ...],
     "Limit": integer
   },
   "Filter": {"string": "string"
     ...}
```

AWS CLI exemplos de Performance Insights

Os exemplos a seguir mostram como usar o AWS CLI para Performance Insights.

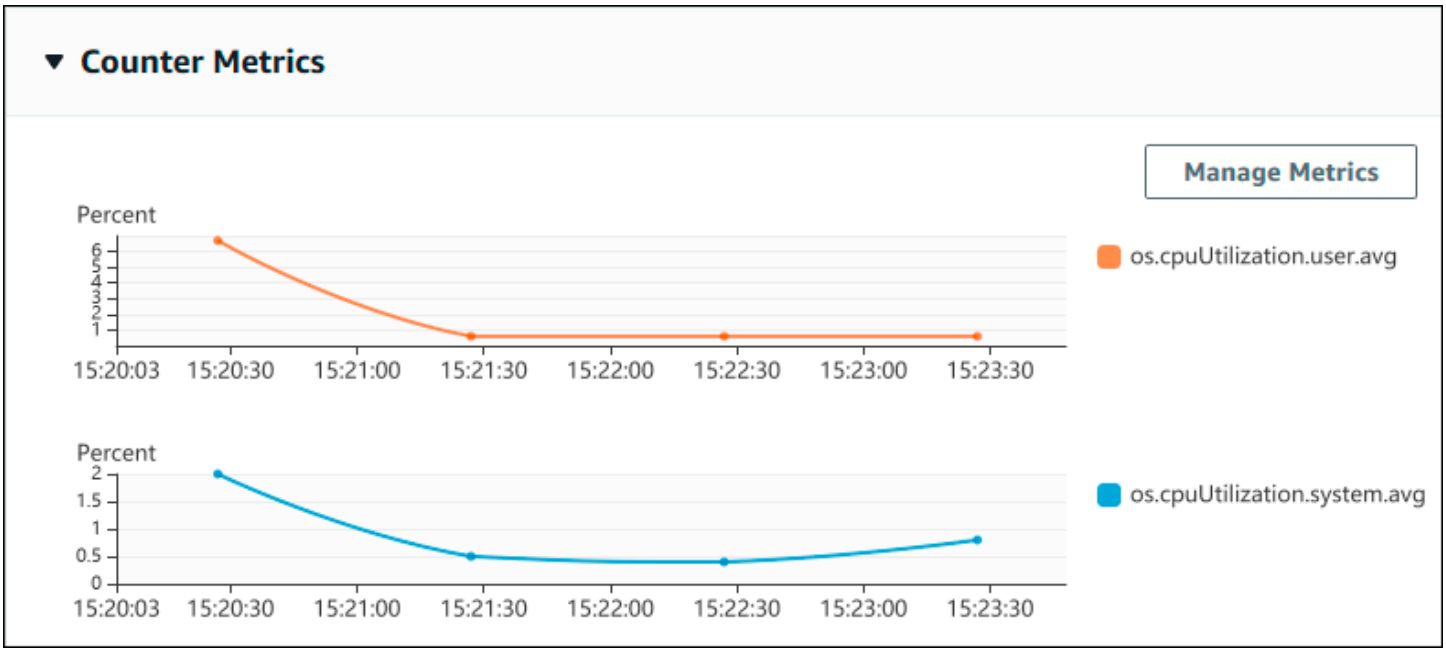
Tópicos

- Recuperar métricas de contador

- Recuperar a média de carga de banco de dados para eventos de espera superior
- Recuperar a média de carga de banco de dados para consulta principal
- Recuperação da média de carga de banco de dados filtrada por Consulta

Recuperar métricas de contador

A captura de tela a seguir mostra dois gráficos de métricas de contador no Console de gerenciamento da AWS.



O exemplo a seguir mostra como reunir os mesmos dados que o Console de gerenciamento da AWS usa para gerar os dois gráficos de métricas de contador.

Para Linux, macOS ou 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"}]'
```

Você também pode tornar um comando mais fácil de ler, especificando um arquivo para a opção `--metrics-query`. O exemplo a seguir usa um arquivo chamado query.json para a opção. O arquivo tem o seguinte conteúdo.

```
[
    {
        "Metric": "os.cpuUtilization.user.avg"
    },
    {
        "Metric": "os.cpuUtilization.idle.avg"
    }
]
```

Execute o seguinte comando para usar o arquivo.

Para Linux, macOS ou 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
```

O exemplo anterior especifica os seguintes valores para as opções:

- `--service-type`— DOCDB para Amazon DocumentDB
- `--identifier` – O ID do recurso para a instância do banco de dados
- `--start-time` e `--end-time` – Os valores ISO 8601 de `DateTime` para o período a consultar, com vários formatos compatíveis

Ele consulta um intervalo de tempo de uma hora:

- `--period-in-seconds` – `60` para uma consulta por minuto
- `--metric-queries` – uma matriz de duas consultas, cada uma apenas para uma métrica.

  O nome da métrica usa pontos para classificar a métrica em uma categoria útil, com o elemento final sendo uma função. No exemplo, a função é `avg` para cada consulta. Assim como na Amazon CloudWatch, as funções suportadas são `min` `maxtotal`, `avg` e.

A resposta é semelhante à seguinte.

```
{
    "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
```

A resposta tem `Identifier`, `AlignedStartTime` e `AlignedEndTime`. Se o valor de `--period-in-seconds` fosse `60`, as horas de início e término seriam alinhadas ao minuto. Se `--period-in-seconds` fosse `3600`, as horas de início e término teriam sido alinhadas à hora.

O `MetricList` na resposta tem um número de entradas, cada uma com uma entrada `Key` e `DataPoints`. Cada `DataPoint` tem um `Timestamp` e um `Value`. Cada lista `Datapoints` tem 60 pontos de dados, pois as consultas são para dados por minuto ao longo de uma hora, com `Timestamp1/Minute1`, `Timestamp2/Minute2` e assim por diante, até `Timestamp60/Minute60`.

Como a consulta é para duas métricas de contador diferentes, há dois elementos na resposta `MetricList`.

Recuperar a média de carga de banco de dados para eventos de espera superior

O exemplo a seguir é a mesma consulta Console de gerenciamento da AWS usada para gerar um gráfico de linha de área empilhada. Este exemplo recupera o `db.load.avg` para a última hora com carga dividida de acordo com os sete principais eventos de espera. O comando é o mesmo que o comando em Recuperar métricas de contador. No entanto, o arquivo query.json tem o seguinte conteúdo.

```
[
    {
```

```
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.wait_state", "Limit": 7 }
    }
]
```

Execute o seguinte comando.

Para Linux, macOS ou 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
```

O exemplo especifica a métrica de `db.load.avg` e um `GroupBy` dos sete principais estados de espera. Para obter detalhes sobre valores válidos para este exemplo, consulte DimensionGroup a Referência da API Performance Insights.

A resposta é semelhante à seguinte.

```
{
    "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
```

Nessa resposta, há três entradas no `MetricList`. Há uma entrada para o `db.load.avg` total, e três entradas cada para o `db.load.avg`, divididas de acordo com um dos três principais eventos de espera. Ao contrário do primeiro exemplo, como havia uma dimensão de agrupamento, deve

haver uma chave para cada agrupamento da métrica. Não pode haver apenas uma chave para cada métrica, como no caso de uso de métricas de contador.

Recuperar a média de carga de banco de dados para consulta principal

O exemplo a seguir agrupa `db.wait_state` pelas 10 principais instruções de consulta. Existem dois grupos diferentes para instruções de consulta:

- `db.query` – a instrução de consulta completa, como `{"find":"customers","filter":{"FirstName":"Jesse"},"sort":{"key":{"$numberInt":"1"}}}`
- `db.query_tokenized` – a instrução de consulta tokenizada, como `{"find":"customers","filter":{"FirstName":"?"},"sort":{"key":{"$numberInt":"?"}},"limit":{"$numberInt":"?"}}`

Ao analisar a performance do banco de dados, pode ser útil considerar instruções de consulta que diferem apenas por seus parâmetros como um item lógico. Então, você pode usar `db.query_tokenized` ao consultar. No entanto, especialmente quando você está interessado em `explain()`, às vezes é mais útil examinar instruções de consulta completas com parâmetros. Existe um relacionamento pai-filho entre as consultas tokenizadas e completas, com várias consultas completas (filhos) agrupadas sob a mesma consulta tokenizada (pai).

O comando neste exemplo é semelhante ao comando em Recuperar a média de carga de banco de dados para eventos de espera superior. No entanto, o arquivo query.json tem o seguinte conteúdo.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.query_tokenized", "Limit": 10 }
    }
]
```

O exemplo a seguir usa `db.query_tokenized`.

Para Linux, macOS ou 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 exemplo consulta mais de 1 horas, com um período de um minuto em segundos.

O exemplo especifica a métrica de `db.load.avg` e um `GroupBy` dos sete principais estados de espera. Para obter detalhes sobre valores válidos para este exemplo, consulte DimensionGroup a Referência da API Performance Insights.

A resposta é semelhante à seguinte.

```
{
    "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
```

Essa resposta tem 11 entradas no `MetricList` (1 total, 10 principais consultas tokenizadas), e cada entrada com 24 `DataPoints` por hora.

Para consultas tokenizadas, existem três entradas em cada lista de dimensões:

- `db.query_tokenized.statement` – a instrução de consulta tokenizada.
- `db.query_tokenized.db_id` — O ID sintético que o Performance Insights gera para você. Este exemplo retorna o ID sintético `pi-1064184600`.
- `db.query_tokenized.id` – o ID da consulta dentro do Performance Insights.

  No Console de gerenciamento da AWS, esse ID é chamado de ID de suporte. É chamado assim porque o ID é um dado que o AWS Support pode examinar para ajudar você a solucionar um problema com seu banco de dados. AWS leva a segurança e a privacidade de seus dados muito a sério, e quase todos os dados são armazenados criptografados com seus AWS KMS key. Portanto, ninguém lá dentro AWS pode ver esses dados. No exemplo precedente, `tokenized.statement` e `tokenized.db_id` são armazenados em formato criptografado. Se você tiver um problema com seu banco de dados, o AWS Support pode ajudá-lo fazendo referência à ID do Support.

Ao consultar, pode ser conveniente especificar `Group` em `GroupBy`. No entanto, para um controle mais refinado sobre os dados retornados, especifique a lista de dimensões. Por exemplo, se tudo o que for necessário for o `db.query_tokenized.statement`, um atributo `Dimensions` poderá ser adicionado ao arquivo query.json.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": {
            "Group": "db.query_tokenized",
```

```
            "Dimensions":["db.query_tokenized.statement"],
            "Limit": 10
        }
    }
]
```

Recuperação da média de carga de banco de dados filtrada por Consulta

A consulta da API correspondente neste exemplo é semelhante ao comando em Recuperar a média de carga de banco de dados para consulta principal. No entanto, o arquivo query.json tem o seguinte conteúdo.

```
[
 {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.wait_state", "Limit": 5  },
        "Filter": { "db.query_tokenized.id": "AKIAIOSFODNN7EXAMPLE" }
    }
]
```

Nessa resposta, todos os valores são filtrados de acordo com a contribuição de consulta tokenizada AKIAIOSFODNN7EXAMPLE especificada no arquivo query.json. As chaves também podem seguir uma ordem diferente de uma consulta sem um filtro, porque são os cinco principais estados de espera que afetaram a consulta filtrada.

CloudWatch Métricas da Amazon para insights de desempenho

O Performance Insights publica automaticamente as métricas na Amazon CloudWatch. Os mesmos dados podem ser consultados no Performance Insights, mas ter as métricas inseridas CloudWatch facilita a adição de CloudWatch alarmes. Também facilita a adição de métricas aos painéis do CloudWatch existentes.

Métrica	Descrição
DBLoad	O número de sessões ativas do Amazon DocumentDB. Normalmente, você deseja os dados para o número médio de sessões ativas. No Performance Insights, esses dados são consultados como `db.load.avg` .

Métrica	Descrição
DBLoadCPU	O número de sessões ativas em que o tipo do estado de espera é CPU. No Performance Insights, esses dados são consultados como `db.load.avg` , filtrados pelo tipo de estado de espera `CPU`.
DBLoadNonCPU	O número de sessões ativas em que o tipo do estado de espera não é CPU.

> **Note**
>
> Essas métricas são publicadas CloudWatch somente se houver carga na instância de banco de dados.

Você pode examinar essas métricas usando o CloudWatch console AWS CLI, o ou a CloudWatch API.

Por exemplo, você pode obter as estatísticas da métrica `DBLoad` executando o 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 exemplo gera uma saída semelhante à seguinte.

```
{
    "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"
}
```


Você pode usar a função matemática `DB_PERF_INSIGHTS` métrica no CloudWatch console para consultar as métricas do contador do Amazon DocumentDB Performance Insights. A função `DB_PERF_INSIGHTS` também inclui a métrica `DBLoad` em intervalos de menos de um minuto. Você pode definir CloudWatch alarmes nessas métricas. Para obter mais detalhes sobre como criar um alarme, consulte Criar um alarme nas métricas do contador do Performance Insights a partir de um AWS banco de dados.

Para obter mais informações sobre CloudWatch, consulte O que é a Amazon CloudWatch? no Guia do CloudWatch usuário da Amazon.

Métricas de contadores do Performance Insights

Métricas de contador são métricas de performance do sistema operacional no painel do Performance Insights. Para ajudar a identificar e analisar problemas de performance, é possível correlacionar métricas de contadores com a carga de banco de dados.

Contadores de sistema operacional do Performance Insights

Os contadores de sistema operacional a seguir estão disponíveis no Performance Insights para o Amazon DocumentDB.

Contador	Type	Métrica
ativo	memory	os.memory.active
buffers	memory	os.memory.buffers
cached	memory	os.memory.cached
dirty	memory	os.memory.dirty
free	memory	os.memory.free
inactive	memory	os.memory.inactive
mapped	memory	os.memory.mapped
pageTables	memory	os.memory.pageTables
slab	memory	os.memory.slab
total	memory	os.memory.total
writeback	memory	os.memory.writeback
idle	cpuUtilization	os.cpu Utilization.idle
system	cpuUtilization	os.cpu Utilization.system
total	cpuUtilization	os.cpu Utilization.total

Contador	Type	Métrica
usuário	cpuUtilization	os.cpu Utilization.user
wait	cpuUtilization	os.cpu Utilization.wait
one	carregar AverageMinute	os.load AverageMinute.one
fifteen	carregar AverageMinute	os.load AverageMinute.fifteen
cinco	carregar AverageMinute	os.load AverageMinute.five
cached	swap	os.swap.cached
free	swap	os.swap.free
em	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 do Amazon DocumentDB Sem Servidor

O Amazon DocumentDB Sem Servidor é uma configuração sob demanda de ajuste de escala automático que automatiza os processos de ajuste dinâmico da capacidade do seu banco de dados Amazon DocumentDB com base na demanda da aplicação. Você será cobrado apenas pelos recursos que seus clusters consumirem. Dessa forma, o DocumentDB Sem Servidor pode ajudar você a ficar dentro do orçamento e evitar pagar por recursos computacionais não utilizados.

Esse tipo de automação é especialmente valioso para ambientes com workloads altamente variáveis e imprevisíveis, como os bancos de dados multilocatário, bancos de dados distribuídos e sistemas de desenvolvimento e teste.

Tópicos

- Casos de uso do DocumentDB Sem Servidor
- Vantagens do Amazon DocumentDB Sem Servidor
- Como o Amazon DocumentDB Sem Servidor funciona
- Requisitos e limitações do DocumentDB Sem Servidor
- Criação de um cluster de banco de dados que use o Amazon DocumentDB Sem Servidor
- Migração para o Amazon DocumentDB Sem Servidor
- Gerenciamento do Amazon DocumentDB Sem Servidor
- Limites das instâncias do Amazon DocumentDB Sem Servidor
- Configuração de escalabilidade do Amazon DocumentDB Sem Servidor
- Monitoramento do Amazon DocumentDB Sem Servidor

Casos de uso do DocumentDB Sem Servidor

Tanto os clusters provisionados do Amazon DocumentDB quanto o DocumentDB Sem Servidor oferecem suporte a muitos tipos de workloads de banco de dados, desde ambientes de desenvolvimento e teste até as aplicações mais exigentes e essenciais para negócios que exijam alta escala e disponibilidade. Mas o DocumentDB Sem Servidor acrescenta outra dimensão às workloads do cliente, a saber, a capacidade de oferecer suporte a sites e aplicações com workloads imprevisíveis.

O DocumentDB Serverless é útil principalmente para os casos de uso a seguir:

- Workloads variáveis: você executa workloads que têm aumentos repentinos e imprevisíveis na atividade. Um exemplo é um site de tráfego que tem um surto de atividades quando começa a chover. Outro é um site de comércio eletrônico com maior tráfego quando você oferece vendas ou promoções especiais. Com o DocumentDB Sem Servidor, seu banco de dados faz escalabilidade automática da capacidade para atender às necessidades da carga de pico da aplicação e reduz a escala novamente na vertical quando o pico de atividades termina. Com o DocumentDB Sem Servidor, não é mais necessário provisionar para capacidade de pico ou média. É possível especificar um limite de capacidade posterior para lidar com a pior situação, e essa capacidade não é usada, a menos que seja necessária.
  - A granularidade da escalabilidade no DocumentDB Sem Servidor ajuda você a combinar a capacidade de forma próxima às necessidades do seu banco de dados. Para um cluster provisionado, o aumento da escala na vertical exige a adição de uma instância inteiramente nova. O DocumentDB Sem Servidor pode adicionar meia DCU quando for necessário apenas um pouco mais de capacidade. Ele pode adicionar 0,5, 1, 1,5, 2 ou meia DCU com base na capacidade adicional necessária para lidar com um aumento na workload. E ele pode remover 0,5, 1, 1,5, 2 ou meia DCU quando a workload diminuir e essa capacidade não for mais necessária.
- Multi-tenant aplicativos — com o DocumentDB serverless, você não precisa gerenciar individualmente a capacidade do banco de dados para cada aplicativo em sua frota. O DocumentDB Sem Servidor gerencia a capacidade individual do banco de dados para você.
  - É possível criar um cluster para cada locatário. Dessa forma, é possível usar recursos como clonagem e restauração de snapshots para aprimorar a alta disponibilidade e a recuperação de desastres, conforme apropriado para cada locatário.
  - Cada locatário pode ter períodos ociosos e de ocupação específicos, dependendo da hora do dia, da época do ano, dos eventos promocionais, etc. Cada cluster pode ter um amplo intervalo de capacidade. Dessa forma, os clusters com baixa atividade geram cobranças mínimas de instância. Qualquer cluster pode ter a escala aumentada na vertical rapidamente para lidar com períodos de alta atividade.
- Novas aplicações: você implanta uma nova aplicação e não tem certeza sobre o tamanho da instância de que precisa. Com o uso do DocumentDB Sem Servidor, é possível configurar um cluster com uma ou várias instâncias e realizar a escalabilidade automática do banco de dados para os requisitos de capacidade de sua aplicação.
- Mixed-use aplicativos — suponha que você tenha um aplicativo de processamento de transações on-line (OLTP), mas experimente periodicamente picos no tráfego de consultas. Ao especificar níveis de promoção para as instâncias do DocumentDB Sem Servidor em um cluster, é possível

configurar seu cluster para que as instâncias do leitor possam ser escaladas independentemente da instância do gravador para lidar com a carga adicional. Quando o pico de uso diminui, as instâncias do leitor têm a escala reduzida na vertical para corresponder à capacidade da instância do gravador.

- Planejamento de capacidade: suponha que você geralmente ajuste a capacidade do seu banco de dados ou verifique a capacidade ideal do banco de dados para sua workload, modificando as classes de todas as instâncias em um cluster. Com o DocumentDB Sem Servidor, é possível evitar essa sobrecarga administrativa. É possível determinar a capacidade mínima e máxima apropriada executando a workload e verificando o quanto as instâncias realmente são escaladas.
  - É possível modificar instâncias existentes de provisionadas para DocumentDB Sem Servidor ou de DocumentDB Sem Servidor para provisionadas. Não é necessário criar um cluster nem uma instância nesses casos.
- Desenvolvimento e testes: além de executar suas aplicações mais exigentes, você também pode usar o DocumentDB Sem Servidor em ambientes de desenvolvimento e testes. Com o DocumentDB Sem Servidor, é possível criar instâncias com uma capacidade mínima baixa em vez de usar classes de instância db.t* com capacidade de intermitência. É possível definir a capacidade máxima alta o suficiente para que essas instâncias ainda possam executar workloads substanciais sem ficar com memória insuficiente. Quando o banco de dados não estiver em uso, todas as instâncias reduzem a escala verticalmente para evitar cobranças desnecessárias.

Uso do Amazon DocumentDB Sem Servidor para workloads provisionadas existentes

Suponha que você já tenha um DocumentDB em execução em um cluster provisionado. É possível verificar como a aplicação funcionaria com o DocumentDB Sem Servidor adicionando uma ou mais instâncias do DocumentDB Sem Servidor ao cluster existente como instâncias do leitor. É possível conferir com que frequência a escala das instâncias do leitor é aumentada ou reduzida verticalmente. Você pode usar o mecanismo de failover do DocumentDB para promover uma instância sem servidor do DocumentDB como gravadora e verificar como ela lida com a carga de trabalho. read/write Dessa forma, é possível alternar com o mínimo de tempo de inatividade e sem alterar o endpoint que suas aplicações cliente usam. Para obter detalhes sobre o procedimento de conversão de clusters existentes para o DocumentDB Sem Servidor, consulte Migração para o Amazon DocumentDB Sem Servidor.

Vantagens do Amazon DocumentDB Sem Servidor

O DocumentDB Sem Servidor destina-se a workloads variáveis. Com essas workloads imprevisíveis, você pode ter dificuldade em planejar quando alterar a capacidade do banco de dados. Você também pode ter problemas para fazer alterações de capacidade com rapidez suficiente usando os mecanismos conhecidos, como adicionar instâncias ou alterar classes de instâncias. O DocumentDB Sem Servidor oferece as vantagens a seguir para ajudar nesses casos de uso:

- Gerenciamento de capacidade mais simples do que o provisionado: o DocumentDB Sem Servidor reduz o esforço para o planejamento de tamanhos de instâncias e seu redimensionamento conforme a workload é alterada. Ele também reduz o esforço para manutenção da capacidade consistente para todas as instâncias em um cluster.

- Escalabilidade mais rápida e fácil durante períodos de alta atividade: o DocumentDB Sem Servidor escala a capacidade computacional e de memória conforme necessário, sem interrupções nas transações do cliente nem em sua workload geral. A capacidade de usar instâncias do leitor com o DocumentDB Sem Servidor ajuda você a aproveitar a escalabilidade horizontal e vertical. A capacidade de usar clusters globais do DocumentDB significa que você pode distribuir sua carga de trabalho de leitura sem servidor do DocumentDB entre vários. Regiões da AWS Esse recurso é mais conveniente do que os mecanismos de escalabilidade de clusters provisionados.

- Cost-effective durante períodos de baixa atividade — o DocumentDB serverless ajuda você a evitar o superprovisionamento de suas instâncias. O DocumentDB Sem Servidor adiciona recursos em incrementos granulares quando as instâncias aumentam de escala. Você paga somente pelos recursos de banco de dados que consumir: O uso de recursos do DocumentDB Sem Servidor é medido por segundo. Dessa forma, quando a escala de uma instância de banco de dados é reduzida verticalmente, o uso reduzido de recursos é registrado imediatamente.

- Paridade de recursos com provisionado: é possível pode usar todos os recursos do DocumentDB com o DocumentDB Sem Servidor. Por exemplo, com o DocumentDB serverless, você pode usar instâncias de leitor, autenticação de banco de dados AWS Identity and Access Management (IAM) e Performance Insights.

  Especificamente, com o DocumentDB Sem Servidor, é possível aproveitar os recursos de clusters provisionados a seguir:

- Instâncias do leitor: o DocumentDB Sem Servidor pode aproveitar as instâncias do leitor para escalar na horizontal. Quando um cluster contém uma ou mais instâncias do leitor, ele pode realizar failover imediatamente em caso de problemas com a instância do gravador.
- Multi-AZ clusters — Você pode distribuir as instâncias sem servidor do DocumentDB de um cluster em várias zonas de disponibilidade (AZs). A configuração de um Multi-AZ cluster ajuda a garantir a continuidade dos negócios, mesmo no caso raro de problemas que afetem uma AZ inteira.
- Clusters globais — Você pode usar o DocumentDB sem servidor em combinação com clusters globais do DocumentDB para criar cópias adicionais somente para leitura do seu cluster em outros para fins de recuperação de desastres. Regiões da AWS

Como o Amazon DocumentDB Sem Servidor funciona

Tópicos

- Visão geral do
- Configurações para clusters do Amazon DocumentDB
- Capacidade de escalabilidade Amazon DocumentDB Sem Servidor
- Escalabilidade do Amazon DocumentDB Sem Servidor
- Estado ocioso (0,5 DCUs)

Visão geral do

O Amazon DocumentDB Sem Servidor é adequado para as workloads mais exigentes e altamente variáveis. Por exemplo, o uso do banco de dados pode ser pesado por um curto período de tempo, seguido por longos períodos de atividade leve ou nenhuma atividade. Alguns exemplos são sites de varejo, jogos ou esportes com eventos promocionais periódicos e bancos de dados que produzem relatórios quando necessário. Outros são ambientes de desenvolvimento e teste e novas aplicações em que o uso pode aumentar rapidamente. Para casos como esses e muitos outros, configurar a capacidade corretamente antecipadamente nem sempre é possível com o modelo provisionado. Também pode resultar em custos mais elevados se você provisionar em excesso e tem capacidade que você não usa.

Em contrapartida, os clusters provisionados do DocumentDB são adequados para workloads estáveis. Com clusters provisionados, você escolhe uma classe de instância que tem uma

quantidade predefinida de memória, potência da CPU, I/O largura de banda e assim por diante. Se sua workload for alterada, modifique manualmente a classe da instância do gravador e dos leitores. O modelo provisionado funciona bem quando você pode ajustar a capacidade antes dos padrões de consumo esperados e é aceitável ter breves interrupções enquanto você altera a classe da instância do gravador e dos leitores do cluster.

O DocumentDB Sem Servidor foi arquitetado desde o início para oferecer suporte a clusters com tecnologia sem servidor que sejam instantaneamente escaláveis. O DocumentDB Sem Servidor foi projetado para fornecer o mesmo grau de segurança e isolamento dos gravadores e leitores provisionados. Esses aspectos são cruciais em ambientes de nuvem sem servidor multilocatário. O mecanismo de escalabilidade dinâmica tem muito pouca sobrecarga para que ele possa responder rapidamente às alterações na workload do banco de dados. Também é poderoso o suficiente para atender a aumentos dramáticos na demanda de processamento.

Usando o DocumentDB Sem Servidor, é possível criar um cluster do DocumentDB sem ficar limitado a uma capacidade de banco de dados específica para cada gravador e leitor. Você especifica apenas o intervalo mínimo e máximo de capacidade. O DocumentDB escala cada gravador ou leitor do DocumentDB Sem Servidor no cluster dentro desse intervalo de capacidade. Ao usar um Multi-AZ cluster em que cada gravador ou leitor pode escalar dinamicamente, você pode aproveitar a escalabilidade dinâmica e a alta disponibilidade.

O DocumentDB Sem Servidor escala os recursos do banco de dados automaticamente com base nas especificações de capacidade mínima e máxima. A escalabilidade é rápida porque a maioria das operações de eventos de escalabilidade mantém o gravador ou o leitor no mesmo host. Nos raros casos em que um gravador ou leitor do DocumentDB Sem Servidor é movido de um host para outro, o DocumentDB Sem Servidor gerencia as conexões automaticamente. Você não precisa alterar o código da aplicação cliente do seu banco de dados nem suas strings de conexão do banco de dados.

Com o DocumentDB Sem Servidor, como acontece com os clusters provisionados, a capacidade de armazenamento e a capacidade computacional são separadas. Quando nos referimos à capacidade e à escalabilidade do DocumentDB Sem Servidor, é sempre a capacidade computacional que está sendo aumentada ou reduzida. Assim, seu cluster pode conter muitos terabytes de dados, mesmo quando a capacidade da CPU e da memória são reduzidas para níveis baixos.

Em vez de provisionar e gerenciar servidores, você especifica a capacidade do banco de dados. A capacidade real de cada gravador ou leitor do DocumentDB Sem Servidor varia com o tempo, dependendo da workload. Para obter detalhes sobre esse mecanismo, consulte Escalabilidade do Amazon DocumentDB Sem Servidor.

Configurações para clusters do Amazon DocumentDB

Para cada um dos clusters de banco de dados do Amazon DocumentDB, é possível escolher qualquer combinação de capacidade do DocumentDB Sem Servidor, capacidade provisionada ou ambas.

É possível configurar um cluster que contenha a capacidade do DocumentDB Sem Servidor e a capacidade provisionada, denominado cluster de configuração mista. Por exemplo, suponha que você precise de mais read/write capacidade do que a disponível para um gravador sem servidor DocumentDB. Nesse caso, é possível configurar o cluster com um gravador provisionado muito grande. Então você ainda pode usar o DocumentDB Sem Servidor para os leitores. Ou suponha que a workload de gravação do cluster varie, mas a workload de leitura permaneça estável. Nesse caso, é possível configurar seu cluster com um gravador do DocumentDB Sem Servidor e um ou mais leitores provisionados.

Você também pode configurar um cluster no qual toda a capacidade seja gerenciada pelo DocumentDB Sem Servidor. Para fazer isso, é possível criar um cluster e usar o DocumentDB Sem Servidor desde o início. Ou é possível substituir toda a capacidade provisionada em um cluster existente pelo DocumentDB Sem Servidor. Para saber os procedimentos para criar um cluster com o DocumentDB Sem Servidor ou para alternar um cluster existente para o DocumentDB Sem Servidor, consulte Criação de um cluster de banco de dados que use o Amazon DocumentDB Sem Servidor e Migração para o Amazon DocumentDB Sem Servidor.

Se você nunca usar o DocumentDB Sem Servidor em um cluster, todos os gravadores e leitores no cluster serão provisionados. Esse é o tipo mais comum de cluster que a maioria dos usuários conhece. A capacidade provisionada é constante. As cobranças são relativamente fáceis de prever. No entanto, você precisa prever antecipadamente de quanta capacidade precisa. Em alguns casos, suas previsões podem ser imprecisas ou suas necessidades de capacidade podem mudar. Nesses casos, seu cluster pode ficar subprovisionado (mais lento do que o desejável) ou superprovisionado (mais caro do que o desejável).

Capacidade de escalabilidade Amazon DocumentDB Sem Servidor

A unidade de medida para o Amazon DocumentDB Sem Servidor é a Unidade de capacidade do DocumentDB (DCU). A capacidade de escalabilidade do DocumentDB Sem Servidor não está vinculada às classes de instância que você usa para clusters provisionados.

Cada DCU é uma combinação de aproximadamente 2 gigabytes (GB) de memória, CPU correspondente e rede. É necessário especificar o intervalo de capacidade do banco de

dados usando essa unidade de medida. As métricas `ServerlessDatabaseCapacity` e `DCUUtilization` CloudWatch ajudam você a determinar quanta capacidade o banco de dados está realmente usando e em que ponto está essa capacidade no intervalo especificado.

A qualquer momento, cada gravador ou leitor do DocumentDB Sem Servidor tem uma capacidade. A capacidade é um número de ponto flutuante que representa DCUs. A capacidade aumenta ou diminui sempre que o gravador ou o leitor é escalado. Esse valor é medido a cada segundo. Para cada cluster em que você pretende usar o DocumentDB Sem Servidor, é necessário definir um intervalo de capacidade: os valores mínimo e máximo de capacidade que cada escritor ou leitor do DocumentDB Sem Servidor pode escalar entre eles. O intervalo de capacidade é o mesmo para cada gravador ou leitor do DocumentDB Sem Servidor em um cluster. Cada gravador ou leitor do DocumentDB Sem Servidor tem sua própria capacidade, inserindo-se em algum ponto desse intervalo.

O DocumentDB serverless é suportado no DocumentDB 5.0 e 8.0 com uma faixa de capacidade de 0,5 a 256 DCUs.

A menor capacidade do DocumentDB Sem Servidor que pode ser definida é 0,5 DCUs. É possível especificar um número maior se for menor ou igual ao valor máximo da capacidade com suporte. Definir a capacidade mínima para um número pequeno permite que clusters levemente carregados consumam recursos de computação mínimos. Ao mesmo tempo, eles ficam prontos para aceitar conexões imediatamente e aumentam a escala na vertical quando ficam ocupados.

Defina o mínimo como um valor que permita que cada gravador ou leitor mantenha o conjunto de trabalho do aplicativo no pool de buffer. Dessa forma, o conteúdo do grupo de buffer não é descartado durante períodos ociosos. Para conhecer todas as considerações ao escolher o intervalo de capacidade de escalabilidade, consulte Escolha do intervalo de capacidade de escalabilidade para um cluster do DocumentDB Sem Servidor.

Dependendo de como você configura os leitores em uma Multi-AZ implantação, suas capacidades podem ser vinculadas à capacidade do gravador ou de forma independente. Para obter detalhes sobre como fazer isso, consulte Visualização e modificação do nível de promoção de leitores com tecnologia sem servidor.

O monitoramento do DocumentDB Sem Servidor envolve medir os valores de capacidade para o gravador e os leitores em seu cluster ao longo do tempo. Se a escala do banco de dados não for reduzida na vertical para a capacidade mínima, será possível realizar ações como ajustar o mínimo e otimizar sua aplicação de banco de dados. Se o banco de dados atingir consistentemente

a capacidade máxima, será possível realizar ações como aumentar o máximo. Também é possível otimizar sua aplicação de banco de dados e distribuir a carga de consulta por mais leitores.

As cobranças pela capacidade sem servidor do DocumentDB são medidas em termos de. DCU-hours Para obter informações sobre como as cobranças do DocumentDB Sem Servidor são calculadas, consulte a definição de preços do Amazon DocumentDB. Suponha que o número total de gravadores e leitores no cluster seja n. Nesse caso, o cluster consome aproximadamente n x mínimo de DCUs quando você não estiver executando nenhuma operação de banco de dados. O próprio Amazon DocumentDB pode realizar operações de monitoramento ou manutenção que causem uma pequena quantidade de carga. Esse cluster não consome mais do que n x máximo de DCUs quando o banco de dados estiver sendo executado na capacidade total.

Para obter mais detalhes sobre como escolher os valores mínimo e máximo apropriados de DCU, consulte Escolha do intervalo de capacidade de escalabilidade para um cluster do DocumentDB Sem Servidor. Os valores mínimo e máximo de DCU que você especifica também afetam alguns limites de instância do Amazon DocumentDB. Para obter detalhes sobre a interação entre o intervalo de capacidade e os limites das instâncias, consulte Limites das instâncias do Amazon DocumentDB Sem Servidor.

Escalabilidade do Amazon DocumentDB Sem Servidor

Para cada gravador ou leitor do DocumentDB Sem Servidor, o Amazon DocumentDB monitora continuamente a utilização de recursos, como CPU, memória e rede. Essas medições são coletivamente denominadas carga. A carga inclui as operações de banco de dados realizadas pela aplicação. Ela também inclui processamento em segundo plano para o servidor de banco de dados e tarefas administrativas do Amazon DocumentDB. Quando a capacidade é restringida por qualquer um desses fatores, o DocumentDB Sem Servidor aumenta a escala verticalmente. O DocumentDB Sem Servidor também aumenta a escala verticalmente quando detecta problemas de performance que possam ser resolvidos ao fazê-lo. É possível monitorar a utilização de recursos e como isso afeta a escalabilidade do DocumentDB Sem Servidor seguindo os procedimentos em Monitoramento do Amazon DocumentDB Sem Servidor.

A carga pode variar entre o gravador e os leitores em seu cluster. O gravador trata as operações de gravação, além de realizar todas as modificações de dados no volume do cluster. Os leitores podem processar solicitações somente de leitura.

Escalabilidade é a operação que aumenta ou diminui a capacidade do DocumentDB Sem Servidor para seu banco de dados. Com o DocumentDB Sem Servidor, cada gravador e leitor tem seu próprio

valor de capacidade atual, medido em DCUs. O DocumentDB Sem Servidor expande um gravador ou leitor para uma capacidade maior quando a capacidade atual é muito baixa para lidar com a carga. Ele dimensiona o gravador ou o leitor para uma capacidade menor quando sua capacidade atual é maior do que a necessária.

O DocumentDB Sem Servidor pode aumentar a capacidade de forma incremental. Quando sua demanda de workload começa a atingir a capacidade atual do banco de dados de um gravador ou um leitor, o DocumentDB Sem Servidor aumenta o número de DCUs para esse gravador ou leitor. O DocumentDB Sem Servidor escala a capacidade em incrementos necessários para fornecer a melhor performance para os recursos consumidos. A escalabilidade ocorre em incrementos tão pequenos quanto 0,5 DCUs. Quanto maior a capacidade atual, maior o incremento de escalabilidade e, portanto, com maior rapidez a escalabilidade poderá acontecer.

Como a escalabilidade do DocumentDB Sem Servidor é tão frequente, granular e não disruptiva, ela não causa eventos pontuais no Console de gerenciamento da AWS. Em vez disso, você pode medir as CloudWatch métricas da Amazon, como `serverlessDatabaseCapacity` e`DCUUtilization`, e rastrear seus valores mínimo, máximo e médio ao longo do tempo. Para saber mais sobre como monitorar o DocumentDB Sem Servidor, consulte Monitoramento do Amazon DocumentDB Sem Servidor.

O aumento ou a redução da escala na vertical pode ser causada pelo seguinte:

- Utilização da memória
- Utilização da CPU
- Utilização da rede
- Utilização de armazenamento

Você pode monitorar essas causas de escalabilidade em instâncias sem up/down servidor do DocumentDB. Para obter mais informações, consulte Monitoramento do Amazon DocumentDB Sem Servidor.

É possível optar por fazer uma escala do leitor ao mesmo tempo que o gravador associado ou independentemente do gravador. Faça isso especificando o nível de promoção para esse leitor.

- Leitores do DocumentDB Sem Servidor, nos níveis de promoção 0 e 1, são escalados ao mesmo tempo que o gravador. Esse comportamento de escalabilidade torna os leitores nos níveis prioritários 0 e 1 ideais para disponibilidade. O motivo disso é que eles são sempre dimensionados para a capacidade certa a fim de assumir a workload do gravador em caso de failover.

- Os leitores em níveis de promoção 2 a 15 são escalados independentemente do gravador. Cada leitor permanece dentro dos valores mínimo e máximo de DCU especificados para o cluster. Quando um leitor é escalado independentemente do banco de dados do gravador associado, ele pode ficar ocioso e diminuir a escala na vertical enquanto o gravador continua processando um alto volume de transações. Ele ainda estará disponível como destino de failover, se nenhum outro leitor estiver disponível em níveis de promoção mais baixos. No entanto, se for promovido para ser o gravador, talvez seja necessário aumentar a escala na vertical para lidar com toda a workload do gravador.

Para obter detalhes sobre como visualizar e alterar os níveis de promoção de instâncias com tecnologia sem servidor, consulte Visualização e modificação do nível de promoção de leitores com tecnologia sem servidor.

O escalonamento do DocumentDB Sem Servidor pode ocorrer enquanto as conexões do banco de dados estão abertas, enquanto as transações estão em andamento, etc. O DocumentDB Sem Servidor não espera por um ponto silencioso para começar a escalar. A escalabilidade não interrompe nenhuma operação de banco de dados em andamento.

Se sua workload exigir mais capacidade de leitura do que está disponível com um único gravador e um único leitor, será possível adicionar vários leitores do DocumentDB Sem Servidor ao cluster. Cada leitor do DocumentDB Sem Servidor pode ser escalado dentro dos valores de capacidade mínima e máxima especificadas para o seu cluster. É possível usar o endpoint leitor do cluster para direcionar sessões somente leitura para os leitores e reduzir a carga no gravador.

Se o DocumentDB Sem Servidor realiza ou não a escalabilidade e com que rapidez ela ocorre assim que é iniciada, isso também depende das configurações mínima e máxima de DCU para o cluster. Além disso, depende do fato de um leitor estar ou não configurado para ser escalado junto com o gravador ou independentemente dele. Para obter detalhes sobre a configuração de escalabilidade, consulte Configuração de escalabilidade do Amazon DocumentDB Sem Servidor.

Estado ocioso (0,5 DCUs)

Quando os gravadores ou leitores sem servidor do Amazon DocumentDB estão ociosos, as instâncias sem servidor do DocumentDB suportam a redução para um estado ocioso de 0,5 DCUs se o cluster estiver configurado para 0,5. MinCapacity

No estado ocioso, as instâncias do DocumentDB Sem Servidor não têm capacidade computacional de CPU suficiente para lidar com a maioria das workloads de produção, mas estão prontas para aumentar a escala verticalmente rapidamente para o suporte de uma nova workload. Em um estado

não ocioso, as instâncias do DocumentDB Sem Servidor normalmente exigem pelo menos 1,0 a 2,5 DCUs. Portanto, quando as instâncias sem servidor do DocumentDB forem escaladas de um estado ocioso para um estado não ocioso, elas serão escaladas diretamente para 1,0 a 2,5 DCUs (ou o valor de, se for menor). MaxCapacity

Para permitir a redução para 0,5 DCUs quando ociosa, os limites da instância são limitados se o MinCapacity estiver configurado para ser menor ou igual a 1,0 DCUs. Para obter mais informações sobre como os limites são afetados pela MinCapacity configuração, consulteLimites das instâncias do Amazon DocumentDB Sem Servidor.

Requisitos e limitações do DocumentDB Sem Servidor

Requisitos de atributos

Disponibilidade de regiões

Para obter mais informações sobre Regiões da AWS onde o tipo de instância sem servidor do Amazon DocumentDB está disponível, consulte a coluna Serverless em. Classes de instância compatíveis por região

O AWS CLI comando a seguir pode ser usado para verificar as opções exatas de instância sem servidor do DocumentDB oferecidas em uma região específica:

```
aws docdb describe-orderable-db-instance-options \
        --region my_region \
        --db-instance-class db.serverless \
        --engine docdb
```

Disponibilidade da versão do mecanismo

O DocumentDB serverless é compatível com as versões do mecanismo Amazon DocumentDB 5.0 e 8.0. Ele não está disponível nas versões de motor 3.6 ou 4.0.

> **Note**
>
> Há suporte para o DocumentDB Sem Servidor somente nas versões de patch mais recentes da 5.0.0. Certifique-se de que seu cluster esteja atualizado para uma versão recente do patch do mecanismo. Para obter mais informações sobre a manutenção dos clusters do Amazon DocumentDB, consulte Manutenção do Amazon DocumentDB.

Configuração do cluster

Antes de adicionar qualquer instância do Amazon DocumentDB Sem Servidor a um cluster do Amazon DocumentDB, o cluster também deve ter o parâmetro `ServerlessV2ScalingConfiguration` definido. Isso define a faixa de capacidade das instâncias do DocumentDB Sem Servidor no cluster. Para obter mais informações sobre a configuração da escalabilidade, consulte Configuração de escalabilidade do Amazon DocumentDB Sem Servidor.

Configurações de faixa mínima de capacidade de escalabilidade para determinados recursos do Amazon DocumentDB

Alguns recursos do Amazon DocumentDB funcionam com o DocumentDB Sem Servidor, mas poderão causar problemas se o intervalo de capacidade for menor do que o necessário para os requisitos de memória desses recursos com sua workload específica. Nesse caso, seu banco de dados pode não funcionar tão bem quanto de costume ou encontrar erros de falta de memória.

Os recursos a seguir exigem a configuração de um `MinCapacity` and/or `MaxCapacity` valor maior para melhor operação:

- Insights de Performance
- Clusters globais (aplica-se somente ao primário Região da AWS)
- Criação de instâncias com tecnologia sem servidor em um cluster com grande volume de dados

  Isso inclui a criação de instâncias com tecnologia sem servidor como parte de uma restauração de cluster.

Para obter recomendações sobre como definir o intervalo de capacidade apropriado (se você estiver usando esse recurso), consulte Escolha do intervalo de capacidade de escalabilidade para um cluster do DocumentDB Sem Servidor. Para obter informações sobre solução de problemas se o banco de dados encontrar erros de falta de memória devido a um intervalo de capacidade configurado incorretamente, consulte Evitar erros de falta de memória.

Criação de um cluster de banco de dados que use o Amazon DocumentDB Sem Servidor

Criação de um cluster do Amazon DocumentDB Sem Servidor

Com o Amazon DocumentDB Sem Servidor, seus clusters são intercambiáveis com clusters provisionados. É possível ter clusters em que algumas instâncias usem tecnologia sem servidor e algumas sejam provisionadas.

Verifique se a região e a versão do mecanismo desejadas oferecem suporte ao DocumentDB sem servidor. Consulte Requisitos e limitações do DocumentDB Sem Servidor.

Para criar um cluster do Amazon DocumentDB ao qual você possa adicionar instâncias com tecnologia sem servidor, siga o mesmo procedimento usado em Criar um cluster do Amazon DocumentDB. A única diferença é que o argumento `ServerlessV2ScalingConfiguration` também deve ser fornecido.

O argumento `ServerlessV2ScalingConfiguration` especifica a faixa de capacidade de escalabilidade de suas instâncias do DocumentDB Sem Servidor. Ele consiste nos valores mínimo e máximo de unidade de capacidade do DocumentDB (DCU) que se aplicam a todas as instâncias do DocumentDB Sem Servidor no cluster:

- O valor `MinCapacity` especifica a capacidade mínima de escalabilidade.
- O valor `MaxCapacity` especifica a capacidade máxima de escalabilidade.

Para ter mais informações sobre a escalabilidade, consulte Configuração de escalabilidade do Amazon DocumentDB Sem Servidor.

Using the Console de gerenciamento da AWS

O exemplo Console de gerenciamento da AWS de configuração a seguir mostra como criar um cluster sem servidor DocumentDB.

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon DocumentDB.
2. No painel de navegação, escolha Clusters.

> **Tip**
>
> Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

A tabela Clusters será exibida.

3. Escolha Criar.

   A página Criação de cluster do Amazon DocumentDB será exibida.

4. Na página Criar cluster do Amazon DocumentDB, na seção Tipo de cluster, escolha Instance-based cluster (essa é a opção padrão).

5. Na seção Configuração do cluster:

   a. Em Identificador do cluster, insira um nome exclusivo, como **`myserverlesscluster`**. Observe que o console mudará todos os nomes de cluster para minúsculas, independentemente de como eles sejam inseridos.

   b. Em Versão do mecanismo, escolha 5.0.0 (essa é a opção padrão).

6. Na seção Configuração de armazenamento do cluster, escolha Amazon DocumentDB Padrão (essa é a opção padrão).

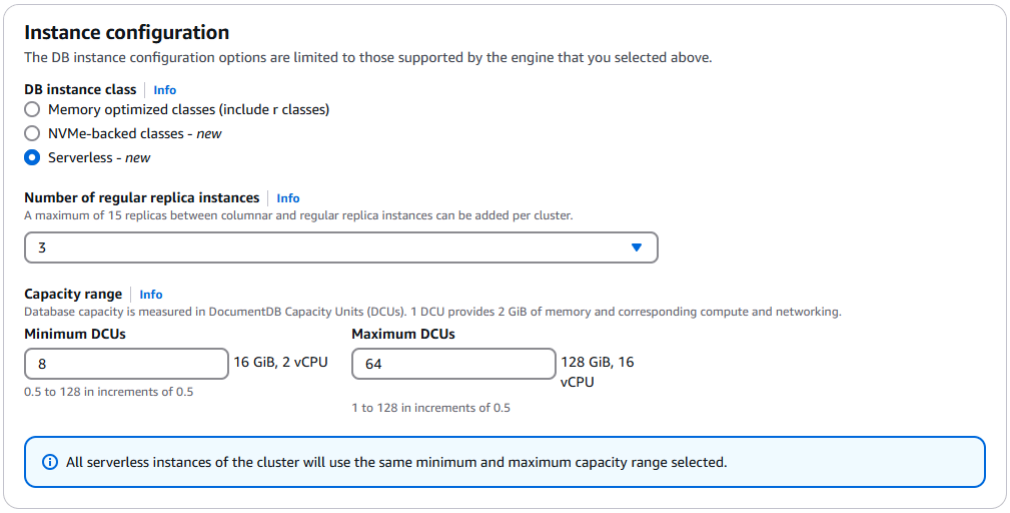
> **Note**
>
> A outra opção nessa categoria é o Amazon DocumentDB I/O-Optimized. Para saber mais sobre as opções, consulte Configurações de armazenamento de cluster do Amazon DocumentDB.

7. Na seção Instance configuration:

   a. Em Classe da instância de banco de dados, escolha Serverless.

   b. Em Número de instâncias de réplica regulares, escolha 3 (essa é a opção padrão).

   c. Na seção Intervalo de capacidade, deixe os valores padrão para DCUs mínimas e DCUs máximas. Para receber informações sobre como definir parâmetros, consulte Limites das instâncias do Amazon DocumentDB Sem Servidor.



8. Na seção Conectividade, mantenha a configuração padrão Não se conectar a um recurso de computação do EC2.
9. Na seção Autenticação, insira um nome de usuário para o usuário principal e escolha Autogerenciado. Insira uma senha e confirme-a.

   Se, em vez disso, você escolheu Gerenciado em AWS Secrets Manager, consulte Gerenciamento de senhas com o Amazon DocumentDB e AWS Secrets Manager para obter mais informações.
10. Deixe todas as outras opções conforme o padrão e escolha Criar cluster.

Using the AWS CLI

No exemplo a seguir, substitua cada um *`user input placeholder`* por suas próprias informações ou parâmetros de configuração.

Para criar um cluster compatível com instâncias sem servidor do DocumentDB usando o. AWS CLICriação de um cluster usando o AWS CLI

Inclua os parâmetros adicionais a seguir em seu comando `create-db-cluster`:

```
--serverless-v2-scaling-configuration
     MinCapacity=minimum_capacity,MaxCapacity=maximum_capacity
```

Exemplo:

```
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
```

Adição de uma instância do Amazon DocumentDB Sem Servidor

Para adicionar uma instância do DocumentDB Sem Servidor, siga o mesmo procedimento em Adicionar uma instância do Amazon DocumentDB a um cluster, certificando-se de especificar db.serverless como classe da instância.

Adicionar uma instância sem servidor usando o Console de gerenciamento da AWS.

Para adicionar uma instância do DocumentDB Sem Servidor usando o console, consulte Adicionar uma instância do Amazon DocumentDB a um cluster e escolha a guia Uso do Console de gerenciamento da AWS.

Adicionar uma instância sem servidor usando o AWS CLI

Para adicionar instâncias sem servidor do DocumentDB usando o AWS CLI, consulte Adicionar uma instância do Amazon DocumentDB a um cluster e escolha a guia Usando a. AWS CLI

Use o argumento da CLI de classe de instância a seguir:

```
--db-instance-class db.serverless
```

Exemplo:

```
aws docdb create-db-instance \
      --db-cluster-identifier sample-cluster \
      --db-instance-identifier sample-instance \
      --db-instance-class db.serverless \
      --engine docdb
```

Migração para o Amazon DocumentDB Sem Servidor

Tópicos

- Migração de clusters do DocumentDB existentes para a tecnologia sem servidor
- Migração do MongoDB para o DocumentDB Sem Servidor

Migração de clusters do DocumentDB existentes para a tecnologia sem servidor

Atualização da versão de mecanismo do cluster

Se o cluster provisionado estiver executando uma versão anterior do mecanismo que não ofereça suporte ao DocumentDB Sem Servidor, você precisará primeiro atualizar o cluster para uma versão do mecanismo com suporte. Para obter mais informações, consulte Atualização da versão principal implementada do Amazon DocumentDB no local.

Migração de um cluster provisionado para o DocumentDB Sem Servidor

Para alternar um cluster provisionado para usar o DocumentDB Sem Servidor, siga estas etapas:

1. Verifique se a versão do mecanismo do cluster provisionado precisa ser atualizada para ser usada com o DocumentDB Sem Servidor. Consulte Requisitos e limitações do DocumentDB Sem Servidor.

   > **Note**
   >
   > Se o cluster provisionado estiver executando uma versão do mecanismo que não esteja disponível para o DocumentDB Sem Servidor, atualize a versão do mecanismo do cluster. Consulte Atualização da versão principal implementada do Amazon DocumentDB no local.

2. Defina a configuração de escalabilidade do cluster. Para obter detalhes sobre como escolher a configuração de escalabilidade, consulte Escolha do intervalo de capacidade de escalabilidade para um cluster do DocumentDB Sem Servidor. Para modificar a configuração de escalabilidade de um cluster, consulte Visualização e modificação da configuração do intervalo de capacidade de escalabilidade de um cluster.
3. Configure todas as outras propriedades do cluster para considerar os requisitos e limitações do DocumentDB Sem Servidor em Requisitos e limitações do DocumentDB Sem Servidor.
4. Adicione uma ou mais instâncias do DocumentDB Sem Servidor ao cluster. Siga o procedimento em Adição de uma instância do Amazon DocumentDB Sem Servidor.

> **Note**
>
> Em alguns casos, talvez você já tenha uma ou mais instâncias do leitor provisionadas no cluster. Se esse for o caso, será possível optar por converter um dos leitores em uma instância do DocumentDB Sem Servidor em vez de criar outra instância. Para fazer isso, siga o procedimento em Alterar a classe de uma instância.

5. (Opcional) Execute uma operação de failover para transformar uma instância do DocumentDB Sem Servidor no gravador do cluster. Consulte Failover do Amazon DocumentDB.
6. (Opcional) Converta todas as instâncias provisionadas restantes do Amazon DocumentDB em instâncias do DocumentDB Sem Servidor (consulte Alterar a classe de uma instância) ou as remova do cluster (consulte Excluir uma instância do Amazon DocumentDB).

Using the Console de gerenciamento da AWS

O exemplo de Console de gerenciamento da AWS configuração a seguir mostra o processo de migração usando um cluster provisionado do Amazon DocumentDB que está executando o Amazon DocumentDB 5.0.0, que não exige uma atualização da versão do mecanismo para começar a usar o DocumentDB sem servidor. O cluster é nomeado como `sample` e começa com três instâncias provisionadas chamadas de `sample`, `sample2` e `sample3`. Neste exemplo, essas três instâncias serão substituídas por três instâncias com tecnologia sem servidor.

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon DocumentDB.
2. No painel de navegação, escolha Clusters.

   > **Tip**
   >
   > Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

   A tabela Clusters será exibida.
3. Na tabela Clusters, marque a caixa de seleção do cluster ao qual você deseja adicionar uma instância com tecnologia sem servidor.

4. Escolha Actions (Ações) e escolha Add instances (Adicionar instâncias).
5. Na caixa de diálogo Adicionar instâncias, selecione Tecnologia sem servidor na seção Classe de instância de banco de dados para cada nova instância sem servidor que você deseja criar.
6. Em Configurações de capacidade de tecnologia sem servidor, defina a configuração de escalabilidade com base na descrição da capacidade na caixa de diálogo.
7. (Opcional) Para adicionar outra instância, escolha Adicionar instância. Continue a adicionar instâncias até atingir o número desejado de novas instâncias.

   Neste exemplo, três novas instâncias com tecnologia sem servidor são criadas.
8. Escolha Criar.

   Leva alguns minutos para criar uma instância. Você pode usar o console ou o AWS CLI para ver o status da instância. Para obter mais informações, consulte Monitoramento do status de um cluster do Amazon DocumentDB.
9. Voltando à tabela Clusters, marque as caixas de seleção de todas as três instâncias provisionadas originais.
10. Escolha Ações e, em seguida, escolha Excluir.

Durante a exclusão, uma operação de failover é executada automaticamente para promover uma das instâncias restantes a gravador. Após alguns minutos, o processo de exclusão se concluirá. O cluster existente agora contém três instâncias do DocumentDB Sem Servidor (conforme definido na coluna Tamanho).

Using the AWS CLI

O exemplo de AWS CLI configuração a seguir mostra o processo de migração usando um cluster provisionado do Amazon DocumentDB que está executando o Amazon DocumentDB 5.0.0, que não exige uma atualização da versão do mecanismo para começar a usar o DocumentDB sem servidor. O cluster é nomeado como `sample` e começa com três instâncias provisionadas chamadas de `sample`, `sample2` e `sample3`. Neste exemplo, essas três instâncias serão substituídas por três instâncias com tecnologia sem servidor. O cluster é chamado de `sample-cluster`, e começa com duas instâncias provisionadas chamadas de `sample-provisioned-instance-1` e `sample-provisioned-instance-2`, uma instância de gravador e uma instâmcia de leitor.

No exemplo a seguir, substitua cada um *user input placeholder* por suas próprias informações ou parâmetros de configuração.

Use a operação `aws docdb describe-db-clusters` para determinar o status de um cluster. O código a seguir descobre o status do cluster `sample-cluster` e exibe os resultados em uma tabela:

```
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             |
+---------------------------------+---------------------------------
+------------------+----------------+
```

Adicione a configuração de escalabilidade do cluster:

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --serverless-v2-scaling-configuration MinCapacity=0.5,MaxCapacity=16
```

Adicione as instâncias com tecnologia sem servidor. Neste exemplo, novas instâncias com tecnologia sem servidor chamadas de `sample-serverless-instance-1` e `sample-serverless-instance-2` são adicionadas:

```
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
```

Insira o comando a seguir para esperar que as instâncias com tecnologia sem servidor estejam disponíveis 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
```

Realize uma operação de failover para tornar a nova instância `sample-serverless-instance-1` o gravador do cluster.

```
aws docdb failover-db-cluster \
    --db-cluster-identifier sample-cluster \
    --target-db-instance-identifier sample-serverless-instance-1
```

A operação de failover leva alguns segundos para ser concluída, após a qual sample-serverless-instance-1 se tornará o gravador do cluster. Verifique isso com a entrada a seguir:

```
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             |
+--------------------------------+--------------------------------
+------------------+----------------+
```

Para finalizar, exclua as instâncias provisionadas originais:

```
aws docdb delete-db-instance \
    --db-instance-identifier sample-provisioned-instance-1

aws docdb delete-db-instance \
    --db-instance-identifier sample-provisioned-instance-2
```

Migração do MongoDB para o DocumentDB Sem Servidor

É possível migrar seus bancos de dados do MongoDB para o DocumentDB Sem Servidor, da mesma forma que com o Amazon DocumentDB provisionado. Para obter mais informações, consulte Migração e atualização do Amazon DocumentDB.

Gerenciamento do Amazon DocumentDB Sem Servidor

Visualização e modificação da configuração do intervalo de capacidade de escalabilidade de um cluster

O argumento `ServerlessV2ScalingConfiguration` especifica a faixa de capacidade de escalabilidade de suas instâncias do DocumentDB Sem Servidor. Ele consiste nos valores mínimo e máximo de unidade de capacidade do DocumentDB (DCU) que se aplicam a todas as instâncias do DocumentDB Sem Servidor no cluster.

- **`MinCapacity`**: a capacidade mínima de escalabilidade de qualquer instância com tecnologia sem servidor do DocumentDB no cluster.

- **`MaxCapacity`**: a capacidade máxima de escalabilidade de qualquer instância com tecnologia sem servidor do DocumentDB no cluster.

> **Note**
>
> As modificações na configuração de escalabilidade a seguir exigem a reinicialização da instância para refletir os novos valores `MinCapacity` e `MaxCapacity`:
>
> - Quaisquer alterações no valor `MaxCapacity`
> - Alteração do valor de `MinCapacity` para 1,0 ou inferior a partir de um valor maior
> - Alteração do valor de `MinCapacity` para um maior que 1,0 a partir de um valor menor

Para obter mais informações sobre a configuração de escalabilidade e como selecionar os limites adequados de capacidade de escalabilidade, consulte Configuração de escalabilidade do Amazon DocumentDB Sem Servidor.

Using the Console de gerenciamento da AWS

O exemplo de Console de gerenciamento da AWS configuração a seguir mostra como visualizar e editar as definições de configuração de escalabilidade de um cluster sem servidor DocumentDB.

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon DocumentDB.
2. No painel de navegação, escolha Clusters.

   > **Tip**
   >
   > Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

   A tabela Clusters será exibida.
3. Na tabela Clusters, marque a caixa de seleção do cluster para o qual você deseja modificar a capacidade de escalabilidade.

4. Escolha Ações e, em seguida, Modificar.

   A caixa de diálogo Modificar cluster será exibida.

5. Localize a seção Configuração de capacidade de tecnologia sem servidor e defina a configuração de escalabilidade (Intervalo de capacidade) com base na descrição da capacidade na caixa de diálogo.

   Para obter mais informações sobre escalabilidade e intervalos de capacidade, consulte Configuração de escalabilidade do Amazon DocumentDB Sem Servidor.

6. Escolha Continue.
7. Em Scheduling of modifications (Programação de modificações), selecione Apply immediately (Aplicar imediatamente).
8. Escolha Modificar Cluster.
9. Depois que a modificação for concluída, cada instância com tecnologia sem servidor deverá ser reinicializada. Para minimizar a indisponibilidade do gravador, execute a sequência de operações a seguir:

   a. Reinicialize cada instância de leitor com tecnologia sem servidor.

      i. Selecione a instância do leitor, escolha Ações e, em seguida, Reinicializar.

      ii. Aguarde até que o status da instância retorne para Disponível.

   b. Execute uma operação de failover em uma instância com tecnologia sem servidor reinicializada.

      i. Selecione o cluster, escolha Ações e, em seguida, Failover.

      ii. Aguarde a conclusão da operação de failover.

   c. Reinicialize a instância com tecnologia sem servidor restante.

      i. Selecione a instância restante, escolha Ações e, em seguida, Reinicializar.

      ii. Aguarde até que o status da instância retorne para Disponível.

Using the AWS CLI

O exemplo AWS CLI de configuração a seguir mostra a configuração de escala atual.

No exemplo a seguir, substitua cada um *`user input placeholder`* por suas próprias informações ou parâmetros de configuração.

A configuração atual de escalabilidade do cluster pode ser visualizada usando o `describe-db-clusters` AWS CLI comando:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[0].ServerlessV2ScalingConfiguration'
```

Veja a seguir a saída desse comando:

```
{
    "MinCapacity": 0.5,
    "MaxCapacity": 16.0
}
```

A configuração de escalabilidade do cluster pode ser modificada usando o comando `modify-db-cluster`:

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --serverless-v2-scaling-configuration MinCapacity=0.5,MaxCapacity=32
```

Após a conclusão, cada instância com tecnologia sem servidor deverá ser reinicializada. Para minimizar a indisponibilidade do gravador, podemos executar a sequência de operações a seguir:

```
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
```

Visualização e modificação do nível de promoção de leitores com tecnologia sem servidor

No caso de clusters que contenham várias instâncias do DocumentDB Sem Servidor ou uma mistura de instâncias com tecnologia sem servidor e provisionadas, preste atenção à configuração do nível de promoção de cada instância com tecnologia sem servidor. Essa configuração controla mais aspectos do comportamento das instâncias com tecnologia sem servidor do que das instâncias provisionadas.

No caso de instâncias provisionadas, a escolha do nível de 0 a 15 determina apenas a ordem na qual o Amazon DocumentDB escolhe qual instância de leitor promover a gravador durante uma operação de failover. Contudo, para instâncias com tecnologia sem servidor, o número do nível também determina se a escala da instância será aumentada verticalmente para corresponder à capacidade da instância do gravador ou se será escalada independentemente, com base somente na própria workload. As instâncias de leitor com tecnologia sem servidor no nível 0 ou 1 são mantidas em uma capacidade mínima pelo menos tão alta quanto a instância do gravador. Dessa forma, elas estarão prontas para assumir o controle da instância do gravador em caso de failover. Se a instância do gravador for uma instância provisionada, o Amazon DocumentDB estimará a capacidade do DocumentDB Sem Servidor equivalente. Ele usa essa estimativa como a capacidade mínima para a instância do leitor com tecnologia sem servidor.

As instâncias de leitor do DocumentDB Sem Servidor nos níveis 2 a 15 não têm a mesma restrição em sua capacidade mínima. Quando estiverem ociosas, elas poderão ter a escala reduzida na vertical para o valor mínimo da unidade de capacidade do DocumentDB (DCU) especificado no intervalo de capacidade do cluster.

Using the Console de gerenciamento da AWS

O exemplo de Console de gerenciamento da AWS configuração a seguir mostra como visualizar e modificar as configurações do nível promocional de um leitor de instâncias sem servidor DocumentDB.

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon DocumentDB.

Os níveis de promoção de cada instância são exibidos na coluna Nível de promoção, no Console de gerenciamento da AWS.

2. No painel de navegação, escolha Clusters.

> **Tip**
>
> Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

A tabela Clusters será exibida.

3. Na tabela Clusters, marque a caixa de seleção da instância para a qual você deseja modificar o nível de promoção.
4. Escolha Ações e, em seguida, Modificar.

A caixa de diálogo Modificar instância será exibida.

5. Localize a seção Failover e defina Nível de promoção para o nível desejado.
6. Escolha Continue.
7. Em Scheduling of modifications (Programação de modificações), selecione Apply immediately (Aplicar imediatamente).
8. Escolha Modify instance (Modificar instância).

Using the AWS CLI

O exemplo AWS CLI de configuração a seguir mostra a configuração de escala atual.

No exemplo a seguir, substitua cada um *`user input placeholder`* por suas próprias informações ou parâmetros de configuração.

Os níveis de promoção de todas as instâncias em um cluster podem ser visualizados usando o `describe-db-clusters` AWS CLI comando:

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

```
    --output table
```

Veja a seguir a saída desse comando:

```
-------------------------------------------------------------------------------------------------
|                                          DescribeDBClusters
                  |
+---------------------------------+---------------------------------
+------------------+----------------+
|  DBClusterParameterGroupStatus |      DBInstanceIdentifier       | IsClusterWriter
  | PromotionTier  |
+---------------------------------+---------------------------------
+------------------+----------------+
|  in-sync                       |  sample-serverless-instance-2   |  False
  |  1             |
|  in-sync                       |  sample-serverless-instance-1   |  True
  |  1             |
+---------------------------------+---------------------------------
+------------------+----------------+
```

O nível de promoção de uma instância específica pode ser modificado usando o comando `modify-db-instance`:

```
aws docdb modify-db-instance \
    --db-instance-identifier sample-serverless-instance-2 \
    --promotion-tier 3
```

Limites das instâncias do Amazon DocumentDB Sem Servidor

Para instâncias do DocumentDB Sem Servidor, os limites por instância a seguir dependem da capacidade de escalabilidade atual da instância:

- Memória da instância (GiB)
- Conexões (todas)
- Limite do cursor
- Abrir transações
- Conexões (ativas)

As tabelas a seguir descrevem como os limites por instância das instâncias com tecnologia sem servidor têm a escala reduzida horizontalmente em relação à capacidade de escalabilidade atual da instância.

Observe que os valores dos limites são diferentes, dependendo se a configuração de escalabilidade `MinCapacity` do cluster foi definida como maior que dois (2). Se `MinCapacity` for definida como menor ou igual a 2, os valores dos limites para limite do cursor, transações abertas e conexões (ativas) serão restritos a um valor máximo menor. (A razão técnica para a restrição é apoiar a redução da escala na vertical para um estado “ocioso”. Para obter mais informações, consulte Estado ocioso (0,5 DCUs).

limites de instância db.serverless se ≤ 1 MinCapacity

Capacidade atual (DCU)	Memória da instância (GiB)	Conexões (todas)	Limite do cursor	Abrir transações	Conexões (ativas)
0,5	1	250	6	12	39
1	2	500	12	24	79
2	4	1000	24	48	173
4	8	2000	48	96	416
8	16	4000	96	192	1071
16	32	8000	132	264	1550
32	64	16000	132	264	1550
64	128	32000	132	264	1550
128	256	60000	132	264	1550
256	512	60000	132	264	1550

limites de instância db.serverless se >1 MinCapacity

Capacidade atual (DCU)	Memória da instância (GiB)	Conexões (todas)	Limite do cursor	Abrir transações	Conexões (ativas)
1.5	3	750	18	36	124
2	4	1000	24	48	173
4	8	2000	48	96	416
8	16	4000	96	192	1071
16	32	8000	192	384	2709
32	64	16000	384	768	4500
64	128	32000	768	1536	4500
128	256	60000	1536	3072	4500
256	512	60000	3072	6144	4500

Você pode monitorar e alertar sobre os limites por instância usando as CloudWatch métricas a seguir. Para obter mais informações sobre as CloudWatch métricas do Amazon DocumentDB, consulteMonitorando o Amazon DocumentDB com CloudWatch.

Recurso	CloudWatch métrica limite	CloudWatch métrica de uso (máximo de 1 min.)	CloudWatch métrica de uso
Memória da instância	-	-	FreeableMemory
Conexões (todas)	DatabaseConnectionsLimit	DatabaseConnectionsMax	DatabaseConnections
Cursores	DatabaseCursorsLimit	DatabaseCursorsMax	DatabaseCursors
Transações	TransactionsOpenLimit	TransactionsOpenMax	TransactionsOpen

Configuração de escalabilidade do Amazon DocumentDB Sem Servidor

Tópicos

- Escolha do intervalo de capacidade de escalabilidade para um cluster do DocumentDB Sem Servidor
- Escolha da configuração MinCapacity para um cluster do DocumentDB Sem Servidor
- Escolha da configuração MaxCapacity para um cluster do DocumentDB Sem Servidor
- Evitar erros de falta de memória
- Por que minha instância com tecnologia sem servidor não está reduzindo a escala verticalmente?

Escolha do intervalo de capacidade de escalabilidade para um cluster do DocumentDB Sem Servidor

Antes de adicionar qualquer instância do DocumentDB Sem Servidor a um cluster do Amazon DocumentDB, o cluster também deve ter o parâmetro `ServerlessV2ScalingConfiguration` definido.

O parâmetro `ServerlessV2ScalingConfiguration` consiste em dois valores que definem a faixa de capacidade de escalabilidade da tecnologia sem servidor de qualquer instância com tecnologia sem servidor no cluster:

- **`MinCapacity`**: a capacidade mínima de escalabilidade de qualquer instância com tecnologia sem servidor do DocumentDB no cluster.
- **`MaxCapacity`**: a capacidade máxima de escalabilidade de qualquer instância com tecnologia sem servidor do DocumentDB no cluster.

Escolha da configuração **`MinCapacity`** para um cluster do DocumentDB Sem Servidor

É tentador escolher sempre 0,5 para `MinCapacity`. Esse valor permite que a instância reduza sua escala na vertical para a menor capacidade quando estiver completamente ociosa, permanecendo ativa. No entanto, dependendo de como você usa esse cluster e das outras configurações definidas,

uma capacidade mínima diferente pode ser o mais eficaz. Considere os seguintes fatores ao escolher a configuração de capacidade mínima:

- A taxa de escalabilidade para uma instância do DocumentDB Sem Servidor depende de sua capacidade atual. Quanto maior a capacidade atual, com maior rapidez será possível aumentar a escala dela na vertical. Se você precisar aumentar a escala da instância na vertical com rapidez até uma capacidade muito alta, considere definir a capacidade mínima como um valor em que o intervalo de escalabilidade atenda às suas necessidades.
- Se você normalmente modifica a classe de suas instâncias prevendo uma workload especialmente alta ou baixa, é possível usar essa experiência para realizar uma estimativa aproximada do intervalo de capacidade do DocumentDB Sem Servidor equivalente. Para determinar o tamanho da memória de um tipo de instância provisionada do Amazon DocumentDB, consulte Cotas de instâncias.

  Por exemplo, suponha que você use a classe de instância `db.r6g.xlarge` quando seu cluster tem uma workload baixa. Essa classe de instância tem 32 GiB de memória. Portanto, é possível especificar uma `MinCapacity` de 16 para configurar uma instância com tecnologia sem servidor que possa reduzir a escala verticalmente para aproximadamente a mesma capacidade. Isso ocorre porque cada DCU corresponde a aproximadamente 2 GiB de memória. É possível especificar um valor um pouco menor para permitir que a instância seja reduzida ainda mais no caso de sua instância `db.r6g.xlarge` às vezes ser subutilizada.
- Se a aplicação funcionar de forma mais eficiente quando as instâncias tiverem uma certa quantidade de dados no cache de buffer, considere especificar uma configuração mínima de DCU em que a memória seja grande o suficiente para manter os dados acessados com frequência. Caso contrário, alguns dados serão removidos do cache de buffer quando a escala das instâncias com tecnologia sem servidor for reduzida verticalmente para um tamanho de memória menor. Então, quando a escala das instâncias for novamente aumentada verticalmente, as informações serão lidas de volta no cache do buffer ao longo do tempo. Se a quantidade necessária I/O para trazer os dados de volta ao cache do buffer for substancial, talvez seja mais eficaz escolher um valor mínimo de DCU mais alto. Para obter mais informações, consulte Dimensionamento de instância.
- Se suas instâncias do DocumentDB Sem Servidor são executadas na maior parte do tempo em uma determinada capacidade, considere especificar uma configuração de capacidade mínima inferior à linha de base, mas não muito mais baixa. As instâncias com tecnologia sem servidor podem estimar com mais eficiência quanto e com que rapidez a escala será aumentada verticalmente quando a capacidade atual não for drasticamente menor do que a capacidade necessária.

- Se a workload provisionada tiver requisitos de memória muito altos para classes de instâncias pequenas, como T3 ou T4g, escolha uma configuração mínima de DCU que forneça memória comparável a uma instância R5 ou R6g.
- Use o mínimo a seguir `MinCapacity` para os recursos especificados (sujeitos a alterações):
  - Insights de performance: duas DCUs
  - Clusters globais — 8 DCUs (aplica-se somente ao primário Região da AWS)
- No Amazon DocumentDB, a replicação ocorre na camada de armazenamento, portanto, a capacidade do leitor não afeta diretamente a replicação. No entanto, para instâncias de leitor do DocumentDB Sem Servidor que escalam de forma independente, certifique-se de que a capacidade mínima seja suficiente para lidar com workloads durante períodos intensivos de gravação, a fim de evitar a latência da consulta. Se instâncias de leitor nos níveis de promoção de 2 a 15 apresentarem problemas de performance, considere aumentar a capacidade mínima do cluster. Para obter detalhes sobre como determinar se as instâncias do leitor são escaladas junto com o gravador ou de forma independente, consulte Visualização e modificação do nível de promoção de leitores com tecnologia sem servidor.

  Se você tem um cluster com instâncias de leitor do DocumentDB Sem Servidor, os leitores não escalarão junto com a instância de banco de dados de gravador quando o nível de promoção dos leitores não for 0 ou 1. Nesse caso, definir uma capacidade mínima baixa pode ocasionar atraso excessivo de replicação. Isso ocorre porque os leitores podem não ter capacidade suficiente para aplicar alterações do gravador quando o banco de dados está ocupado. Defina a capacidade mínima como um valor que represente uma quantidade comparável de memória e CPU à instância do gravador.
- O tempo necessário para uma instância do DocumentDB Sem Servidor escalar de sua capacidade mínima para sua capacidade máxima depende da diferença entre seus valores mínimo e máximo de DCU. Quando a capacidade atual da instância for grande, o DocumentDB Sem Servidor aumenta a escala verticalmente em incrementos maiores do que quando a instância parte de uma pequena capacidade. Portanto, se você especificar uma capacidade máxima relativamente grande e a instância passar a maior parte do tempo próxima dessa capacidade, considere aumentar a configuração mínima de DCU. Dessa forma, a escala de uma instância ociosa pode ser aumentada verticalmente novamente até a capacidade máxima com maior rapidez.
- Alguns limites de instância são determinados pela capacidade atual da instâncias com tecnologia sem servidor, como limite de conexões, limite de cursor e limite de transações abertas. Se a capacidade atual da instância for pequena, os limites também serão igualmente pequenos. Se esses limites forem um problema quando sua instância sem servidor tiver a escala reduzida verticalmente para seu valor de `MinCapacity`, considere aumentar `MinCapacity` para um valor

maior. Para obter mais informações, consulte Limites das instâncias do Amazon DocumentDB Sem Servidor.

- Além disso, certos limites de instância são restritos a um valor máximo menor se a `MinCapacity` for definida como menor ou igual a 1,0 DCUs, como limite de conexões ativas, limite de cursor e limite de transações abertas. Se esses limites máximos forem insuficientes para sua carga de trabalho, use um `MinCapacity` valor de pelo menos 1,5 DCUs. Para obter mais informações, consulte Limites das instâncias do Amazon DocumentDB Sem Servidor.

Para obter instruções sobre como modificar a configuração de escalabilidade de um cluster, consulte Gerenciamento do Amazon DocumentDB Sem Servidor.

Escolha da configuração **`MaxCapacity`** para um cluster do DocumentDB Sem Servidor

É tentador sempre escolher algum valor alto para configuração de capacidade do DocumentDB Sem Servidor. Uma capacidade máxima grande permite que a escala da instância seja aumentada verticalmente ao máximo quando estiver executando uma workload intensa. Um valor baixo evita a possibilidade de cobranças inesperadas. Dependendo de como você usa esse cluster e das outras configurações definidas, o valor mais efetivo pode ser maior ou menor do que o imaginado originalmente. Considere os seguintes fatores ao escolher a configuração de capacidade máxima:

- A capacidade máxima deve ser pelo menos tão alta quanto a capacidade mínima. É possível definir a capacidade mínima e máxima como valores idênticos. No entanto, nesse caso, a escala da capacidade nunca é aumentada nem reduzida na vertical. Assim, usar valores idênticos para a capacidade mínima e máxima não é apropriado além das situações de teste.
- A capacidade máxima deve ser pelo menos de 1,0 DCU e no máximo 256 DCUs.
- Monitore a escalabilidade e o uso de recursos de suas instâncias sem servidor. Se sua instância sem servidor estiver frequentemente escalando até a capacidade máxima e atingindo restrições de recursos (por exemplo, quando a `DCUUtilization` métrica está em 100,0), selecione um valor maior. `MaxCapacity` Para obter mais informações, consulte Monitoramento do Amazon DocumentDB Sem Servidor.
- Se você normalmente modifica a classe de suas instâncias provisionadas prevendo uma workload especialmente alta ou baixa, é possível usar essa experiência para realizar uma estimativa do intervalo de capacidade do DocumentDB Sem Servidor equivalente. Para determinar o tamanho da memória de instâncias provisionadas do Amazon DocumentDB, consulte Cotas de instâncias.

Por exemplo, suponha que você use a classe de instância `db.r6g.4xlarge` quando o cluster tem uma workload alta. Essa classe de instância tem 128 GiB de memória. Portanto, é possível especificar uma configuração máxima de DCU de 64 para configurar uma instância com tecnologia sem servidor cuja escala possa ser aumentada verticalmente para aproximadamente a mesma capacidade. Isso ocorre porque cada DCU corresponde a aproximadamente 2 GiB de memória. É possível especificar um valor um pouco maior para aumentar ainda mais a escala da instância veticalmente, caso sua instância `db.r6g.4xlarge` eventualmente não tenha capacidade suficiente para lidar com a workload de forma eficaz.

- Se você tiver um limite orçamentário para o uso do banco de dados, escolha um valor que permaneça dentro desse limite, mesmo que todas as suas instâncias com tecnologia sem servidor sejam executadas na capacidade máxima o tempo todo. Lembre-se de que quando houver n instâncias com tecnologia sem servidor em seu cluster, em tese, a capacidade de tecnologia sem servidor que o cluster pode consumir a qualquer momento é n vezes a configuração máxima de DCU para o cluster. (A quantidade real consumida pode ser menor, por exemplo, se alguns leitores forem escalados independentemente do gravador.)
- Se você fizer uso de instâncias do leitor com tecnologia sem servidor para descarregar parte da workload somente de leitura da instância do gravador, talvez você possa escolher uma configuração de capacidade máxima mais baixa. Esse procedimento é realizado para indicar que nem todas as instâncias de leitor precisam ser escaladas para um valor tão alto como seria necessário se o cluster contivesse apenas uma única instância.
- Suponha que você queira evitar uso excessivo devido a parâmetros de banco de dados mal configurados ou consultas ineficientes em sua aplicação. Nesse caso, você pode evitar o uso excessivo acidental escolhendo uma configuração de capacidade máxima menor que a mais alta absoluta que seja possível definir.
- Se os picos decorrentes da atividade real do usuário forem raros, mas acontecerem, você pode levar essas ocasiões em consideração ao escolher a configuração de capacidade máxima. Se a prioridade for a aplicação continuar a funcionar com performance e escalabilidade totais, especifique uma configuração de capacidade máxima maior do que a observada no uso normal. Se for aceitável a aplicação ser executada com throughput reduzido durante picos extremos de atividade, escolha uma configuração de capacidade máxima um pouco menor. Escolha uma configuração que ainda tenha memória e recursos de CPU suficientes para manter a aplicação em execução.
- Se você ativar configurações em seu cluster que aumentem o uso de memória para cada instância, leve essa memória em consideração ao decidir sobre o valor máximo de DCU. Essas configurações incluem aquelas para Performance Insights e clusters globais. Verifique se o valor

máximo de DCU permite que a escala das instâncias com tecnologia sem servidor seja aumentada verticalmente o suficiente para lidar com a workload quando esses recursos estiverem sendo usados. Para obter informações sobre a solução de problemas causados pela combinação de uma configuração de DCU máxima baixa e recursos do Amazon DocumentDB que ocasionam sobrecarga de memória, consulte Evitar erros de falta de memória (adiante).

- Use o mínimo a seguir `MaxCapacity` para os recursos especificados (sujeitos a alterações):
  - Criação de instância com tecnologia sem servidor em um cluster com um grande volume de dados, 2 DCUs (isso inclui a criação de instâncias com tecnologia sem servidor como parte de uma restauração de cluster).
- Alguns limites de instância são determinados pela capacidade atual da instância, como limite de conexões, limite de cursor e limite de transações abertas. Ao escolher o `MaxCapacity` valor para sua carga de trabalho, lembre-se desses limites de instância para evitar ficar preso por um desses limites. Para obter mais informações, consulte Limites das instâncias do Amazon DocumentDB Sem Servidor.

Para obter instruções sobre como modificar a configuração de escalabilidade de um cluster, consulte Gerenciamento do Amazon DocumentDB Sem Servidor.

Evitar erros de falta de memória

Se uma de suas instâncias do DocumentDB Sem Servidor atinge de forma consistente o limite de sua capacidade máxima, o Amazon DocumentDB indica essa condição definindo o status da instância como incompatible-parameters. Enquanto a instância apresentar o status incompatible-parameters, algumas operações ficarão bloqueadas. Por exemplo, não é possível atualizar a versão do mecanismo. Para obter mais informações sobre o status de uma instância do Amazon DocumentDB, consulteMonitoramento do status de uma instância do Amazon DocumentDB.

Normalmente, sua instância entra nesse status quando ela é reinicializada com frequência devido a erros de falta de memória. O Amazon DocumentDB registra um evento quando esse tipo de reinicialização acontece. Para visualizar eventos de recursos, consulteVisualizando eventos do Amazon DocumentDB. Um uso de memória excepcionalmente alto pode ocorrer devido à sobrecarga causada pela ativação de configurações como os Insights de performance. A causa também pode ser uma workload pesada em sua instância ou o gerenciamento dos metadados associados a um grande número de objetos de esquema.

Se a pressão da memória se tornar menor para que a instância não atinja sua capacidade máxima com muita frequência, o Amazon DocumentDB alterará automaticamente o status da instância de volta para disponível.

Para se recuperar dessa condição, você pode realizar algumas ou todas as seguintes ações:

- Aumente o limite inferior de capacidade para as instâncias com tecnologia sem servidor alterando o valor mínimo da unidade de capacidade do DocumentDB (DCU) para o cluster. Isso evita problemas em que um banco de dados ocioso tem a escala reduzida na vertical para uma capacidade com menos memória do que o necessário para os recursos que estão ativados no cluster. Depois de alterar as configurações de DCU do cluster, reinicie a instância com tecnologia sem servidor. Com esse procedimento, você avalia se o Amazon DocumentDB consegue redefinir o status de volta como disponível.
- Aumente o limite superior de capacidade das instâncias com tecnologia sem servidor alterando o valor máximo de DCU para o cluster. Isso evita problemas em que não é possível aumentar a escala de um banco de dados ocupado na vertical até uma capacidade com memória suficiente para os recursos ativados no cluster e na workload do banco de dados. Depois de alterar as configurações de DCU do cluster, reinicie a instância com tecnologia sem servidor. Com esse procedimento, você avalia se o Amazon DocumentDB consegue redefinir o status de volta como disponível.
- Desative as configurações que exigem sobrecarga de memória. Por exemplo, digamos que você tenha um recurso, como os Insights de performance ativado, mas não o use. Se esse for o caso, é possível desativá-lo. Também é possível ajustar os valores de capacidade mínima e máxima do cluster para valores mais altos com o objetivo de levar em conta a memória utilizada por esses tipos de recursos. Para obter as diretrizes sobre como escolher configurações de capacidade mínima e máxima, consulte Escolha do intervalo de capacidade de escalabilidade para um cluster do DocumentDB Sem Servidor.
- Reduza a workload na instância. Por exemplo, é possível adicionar instâncias de leitor ao cluster para distribuir a carga de consultas somente de leitura por mais instâncias.

Por que minha instância com tecnologia sem servidor não está reduzindo a escala verticalmente?

Em alguns casos, o DocumentDB Sem Servidor não reduz a escala verticalmente até a capacidade mínima, mesmo sem carga no banco de dados. Isso pode acontecer por um dos seguintes motivos:

- Os Insights de performance podem aumentar o uso de recursos e impedir que a escala do banco de dados seja reduzida verticalmente para a capacidade mínima. Esses recursos incluem o seguinte:
- Se a escala de uma instância de leitor não estiver sendo reduzida verticalmente ao mínimo e permanecer com a mesma capacidade ou maior que a instância do gravador, verifique o nível de prioridade da instância do leitor. As instâncias de leitor do DocumentDB Sem Servidor nos níveis 0 ou 1 são mantidas em uma capacidade mínima pelo menos tão alta quanto a instância do gravador. Altere o nível de prioridade do leitor para 2 ou superior para que ele aumente e reduza a escala verticalmente independentemente do gravador. Para obter mais informações, consulte Escalabilidade do Amazon DocumentDB Sem Servidor.
- Workloads de banco de dados pesadas podem aumentar o uso de recursos.
- Grandes volumes de banco de dados podem aumentar o uso de recursos. O Amazon DocumentDB usa recursos de memória e CPU para o gerenciamento de clusters. O Amazon DocumentDB exige mais CPU e memória para gerenciar clusters com volumes maiores de banco de dados. Se a capacidade mínima do cluster for menor do que a mínima exigida para o gerenciamento do cluster, a escala do cluster não será reduzida verticalmente para a capacidade mínima.
- A atividade de manutenção em segundo plano pode aumentar periodicamente o uso de recursos.

Se o banco de dados ainda não diminuir para a capacidade mínima configurada, interrompa e reinicie o banco de dados para recuperar quaisquer fragmentos de memória que possam ter se acumulado ao longo do tempo.

> **Note**
>
> Parar e iniciar um banco de dados resulta em tempo de inatividade. Só faça isso quando necessário.

Monitoramento do Amazon DocumentDB Sem Servidor

Para saber mais sobre o monitoramento ni Amazon DocumentDB, consulte Monitoramento e registro no Amazon DocumentDB.

Tópicos

- Sem memória: status de parâmetros incompatíveis

- CloudWatch Métricas da Amazon para DocumentDB sem servidor
- Monitoramento da performance do DocumentDB Sem Servidor com Insights de performance

Sem memória: status de parâmetros incompatíveis

Se uma de suas instâncias de tecnologia sem servidor atinge de forma consistente o limite de sua capacidade máxima, o Amazon DocumentDB indica essa condição definindo o status da instância como incompatible-parameters. Para obter mais informações, consulte Evitar erros de falta de memória.

CloudWatch Métricas da Amazon para DocumentDB sem servidor

Para saber mais sobre como usar CloudWatch com o Amazon DocumentDB, consulteMonitorando o Amazon DocumentDB com CloudWatch.

Você pode visualizar suas instâncias sem servidor CloudWatch para monitorar a capacidade consumida por cada instância com a `ServerlessDatabaseCapacity` métrica. Você também pode monitorar todas as CloudWatch métricas padrão do DocumentDB, como `DatabaseConnections` consultas. Para ver a lista completa de CloudWatch métricas que você pode monitorar para o Amazon DocumentDB, consulteMonitorando o Amazon DocumentDB com CloudWatch. É importante monitorar as seguintes métricas em CloudWatch nível de instância para que você entenda como suas instâncias sem servidor do DocumentDB estão aumentando ou diminuindo. Todas essas métricas são calculadas a cada segundo. Dessa forma, é possível monitorar o status atual das instâncias com tecnologia sem servidor. É possível definir alarmes para notificá-lo se alguma instância com tecnologia sem servidor se aproximar de um limite de métricas relacionadas à capacidade. É possível determinar se as configurações de capacidade mínima e máxima são apropriadas ou se você precisa ajustá-las. É possível determinar onde concentrar seus esforços para otimizar a eficiência de seu banco de dados.

- **`ServerlessDatabaseCapacity`**: como métrica específica da instância, ela relata o número de DCUs representadas pela capacidade atual da instância. Como métrica específica do cluster, ela representa a média dos valores de `ServerlessDatabaseCapacity` de todas as instâncias do DocumentDB Sem Servidor no cluster.
- **`DCUUtilization.`**: esta métrica é nova no DocumentDB Sem Servidor. Esse valor é representado como um percentual. Ela é calculada como o valor da métrica `ServerlessDatabaseCapacity` dividida pelo valor máximo de DCU do cluster. Considere as seguintes diretrizes para interpretar essa métrica e agir:

- Se essa métrica se aproximar de um valor de `100.0`, a escala da instância foi aumentada verticalmente ao nível mais elevado possível. Considere aumentar a configuração máxima de DCU para o cluster. Dessa forma, as instâncias do gravador e do leitor poderão ser escaladas para uma capacidade maior.
- Suponha que uma workload somente de leitura faça com que uma instância do leitor se aproxime de um `DCUUtilization` de `100.0`, enquanto a instância do gravador não está próxima de sua capacidade máxima. Nesse caso, considere adicionar mais instâncias de leitor ao cluster. Dessa forma, é possível distribuir a parte somente de leitura da workload por mais instâncias, reduzindo a carga em cada instância de leitor.
- Suponha que você esteja executando uma aplicação de produção, em que a performance e a escalabilidade sejam as principais considerações. Nesse caso, é possível definir o valor máximo de DCU para o cluster como um número elevado. Seu objetivo é que a métrica `DCUUtilization` permaneça sempre abaixo de 100,0. Com um alto valor máximo de DCU, é possível ter certeza de que há espaço suficiente caso haja picos inesperados na atividade do banco de dados. Você só será cobrado pela capacidade do banco de dados que for realmente consumida.

- **`CPUUtilization`**: esta métrica é interpretada de modo diferente no DocumentDB Sem Servidor em relação às instâncias provisionadas. No caso do DocumentDB Sem Servidor, esse valor é um percentual calculado como a quantidade de CPU sendo usada atualmente dividida pela capacidade da CPU disponível abaixo do valor máximo de DCU do cluster. O Amazon DocumentDB monitora esse valor automaticamente e aumenta a escala verticalmente da sua instância com tecnologia sem servidor quando ela usa consistentemente uma alta proporção de sua capacidade de CPU.

  Se essa métrica se aproximar de um valor de `100.0`, a instância atingiu sua capacidade máxima de CPU. Considere aumentar a configuração máxima de DCU para o cluster. Se essa métrica se aproximar de um valor de `100.0` em uma instância de leitor, considere adicionar mais instâncias de leitor ao cluster. Dessa forma, é possível distribuir a parte somente de leitura da workload por mais instâncias, reduzindo a carga em cada instância de leitor.
- **`FreeableMemory`**: este valor representa a quantidade de memória não utilizada que está disponível quando a instância do DocumentDB Sem Servidor é escalada para sua capacidade máxima. Para cada DCU em que a capacidade atual esteja abaixo da capacidade máxima, esse valor aumenta em aproximadamente 2 GiB. Assim, essa métrica não se aproxima de zero até que a escala da instância seja aumentada verticalmente ao nível mais alto possível.

Se essa métrica se aproximar de um valor de zero, a escala da instância terá sido aumentada verticalmente ao nível mais alto possível e estará se aproximando do limite de sua memória disponível. Considere aumentar a configuração máxima de DCU para o cluster. Se essa métrica se aproximar de um valor de zero em uma instância de leitor, considere adicionar mais instâncias de leitor ao cluster. Dessa forma, é possível distribuir a parte somente de leitura da workload por mais instâncias, reduzindo o uso de memória em cada instância de leitor.

- **`TempStorageIops`**: o número de IOPS realizadas no armazenamento local anexado à instância. Ele inclui as IOPS para leituras e gravações. Essa métrica representa uma contagem e é medida uma vez por segundo. Essa métrica é nova no DocumentDB Sem Servidor. Para obter detalhes, consulte Monitorando o Amazon DocumentDB com CloudWatch.
- **`TempStorageThroughput`**: a quantidade de dados transferidos do armazenamento local associado à instância. Essa métrica é apresentada em bytes e é medida uma vez por segundo. Essa métrica é nova no DocumentDB Sem Servidor. Para obter detalhes, consulte Monitorando o Amazon DocumentDB com CloudWatch.

Normalmente, a maioria dos aumentos de escala verticalmente das instâncias do DocumentDB Sem Servidor é ocasionada por uso de memória e atividade da CPU. As métricas `TempStorageIops` e `TempStorageThroughput` podem ajudar você a diagnosticar os casos raros em que a atividade de rede para transferências entre sua instância e dispositivos de armazenamento local é responsável por aumentos inesperados de capacidade. Para monitorar outras atividades de rede, você pode usar estas métricas existentes:

- `NetworkReceiveThroughput`
- `NetworkThroughput`
- `NetworkTransmitThroughput`
- `StorageNetworkReceiveThroughput`
- `StorageNetworkThroughput`
- `StorageNetworkTransmitThroughput`

Como as CloudWatch métricas sem servidor do DocumentDB se aplicam à sua AWS fatura

As cobranças sem servidor do DocumentDB em sua AWS fatura são calculadas com base na mesma `ServerlessDatabaseCapacity` métrica que você pode monitorar. O mecanismo de

cobrança pode diferir da CloudWatch média computada dessa métrica nos casos em que você usa a capacidade sem servidor do DocumentDB por apenas parte de uma hora. Também pode ser diferente se os problemas do sistema tornarem a CloudWatch métrica indisponível por breves períodos. Portanto, você pode ver um valor ligeiramente diferente de DCU-hours em sua fatura do que se você mesmo calculasse o número a partir do valor `ServerlessDatabaseCapacity` médio.

Exemplos de comandos do Amazon CloudWatch CLI para métricas sem servidor do DocumentDB

Os AWS CLI exemplos a seguir demonstram como você pode monitorar as CloudWatch métricas mais importantes relacionadas ao DocumentDB sem servidor. Em cada caso, substitua a string `Value=` pelo parâmetro `--dimensions` com o identificador de sua própria instância do DocumentDB Sem Servidor.

O exemplo de Linux a seguir exibe os valores de capacidade mínima, máxima e média de uma instância, medidos a cada dez minutos em uma hora. Os comandos date do Linux especificam as horas de início e término em relação à data e hora atuais. A função `sort_by` no parâmetro `--query` classifica os resultados cronologicamente com base no 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
```

Os exemplos de Linux a seguir demonstram o monitoramento da capacidade de uma instância em um cluster. Eles medem a utilização mínima, máxima e média da capacidade de uma instância. As medições são realizadas uma vez por hora durante um período de três horas. Esses exemplos usam a métrica `DCUUtilization`, que representa um percentual do limite superior em DCUs, em vez de `ServerlessDatabaseCapacity`, que representa um número fixo de DCUs. Dessa forma, não é necessário conhecer os números reais para os valores mínimo e máximo de DCU no intervalo de capacidade. É possível ver porcentagens que variam de 0 a 100.

```
aws cloudwatch get-metric-statistics \
```

```
    --metric-name "DCUUtilization" \
    --start-time "$(date -d '3 hours ago')" \
    --end-time "$(date -d 'now')" \
    --period 3600 \
    --namespace "AWS/DocDB" \
    --statistics Minimum Maximum Average \
    --dimensions Name=DBInstanceIdentifier,Value=my_instance \
    --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' \
    --output table
```

O exemplo do Linux a seguir realiza medições semelhantes às anteriores. Nesse caso, as medidas são para a métrica `CPUUtilization`. As medições são realizadas a cada dez minutos durante um período de uma hora. Os números representam o percentual de CPU disponível usado, com base nos recursos de CPU disponíveis para a configuração de capacidade máxima para a instância.

```
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
```

O exemplo do Linux a seguir realiza medições semelhantes às anteriores. Nesse caso, as medidas são para a métrica `FreeableMemory`. As medições são realizadas a cada dez minutos durante um período de uma 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
```

Monitoramento da performance do DocumentDB Sem Servidor com Insights de performance

É possível usar os Insights de performance para monitorar a performance de instâncias do DocumentDB Sem Servidor. Para saber os procedimentos do Performance Insights, consulte Monitoramento com o Performance Insights.

Os novos contadores dos Insights de performance a seguir se aplicam a instâncias do DocumentDB Sem Servidor:

- **`os.general.serverlessDBCapacity`**: a capacidade atual da instância, em DCUs. O valor corresponde à `ServerlessDatabaseCapacity` CloudWatch métrica da instância.
- **`os.general.dcuUtilization`** — o percentual de capacidade atual da capacidade máxima configurada. O valor corresponde à `DCUUtilization` CloudWatch métrica da instância.
- **`os.general.maxConfiguredDcu`**: a capacidade máxima que você configurou para essa instância do DocumentDB Sem Servidor. É medida em DCUs.
- **`os.general.minConfiguredDcu`**: a capacidade mínima que você configurou para essa instância do DocumentDB Sem Servidor. É medida em DCUs.

Para ver a lista completa de contadores do Performance Insights, consulte Métricas de contadores do Performance Insights.

Quando valores de vCPU são mostrados para uma instância do DocumentDB Sem Servidor nos Insights de performance, esses valores representam estimativas com base no valor de DCU para a instância. No intervalo padrão de um minuto, todos os valores de vCPU fracionários são arredondados para o número inteiro mais próximo. Para intervalos mais longos, o valor de vCPU mostrado é a média dos valores inteiros de vCPU para cada minuto.

Usando clusters elásticos do Amazon DocumentDB

Os clusters elásticos do Amazon DocumentDB suportam cargas de trabalho com milhões reads/writes por segundo e petabytes de capacidade de armazenamento. Os clusters elásticos também simplificam a forma como os desenvolvedores interagem com o Amazon DocumentDB, eliminando a necessidade de escolher, gerenciar ou atualizar instâncias.

> **Note**
>
> Os clusters elásticos usam um protocolo de conexão compatível com o MongoDB 5.0. Eles não são compatíveis com o mecanismo Amazon DocumentDB versão 8.0 e não estão disponíveis nas regiões da China ou AWS GovCloud.

Os clusters elásticos do Amazon DocumentDB foram criados para:

- Fornecer uma solução para clientes que procuram um banco de dados que ofereça escala praticamente ilimitada com recursos avançados de consulta e compatibilidade com a API MongoDB.
- Oferecer aos clientes limites de conexão mais altos e reduza o tempo de inatividade causado pela aplicação de patches.
- Continuar investindo em uma arquitetura nativa de nuvem, elástica e líder de classe para workloads JSON.

Tópicos

- Casos de uso do cluster elástico
- Vantagens dos clusters elásticos
- Disponibilidade de regiões e versões do cluster elástico
- Limitações
- Clusters elásticos do Amazon DocumentDB: como funcionam
- Como iniciar com clusters elásticos do Amazon DocumentDB
- Práticas recomendadas para clusters elásticos do Amazon DocumentDB
- Gerenciamento de clusters elásticos do Amazon DocumentDB
- Criptografia de dados em repouso para clusters elásticos do Amazon DocumentDB

- Service-linked funções em clusters elásticos

Casos de uso do cluster elástico

Os bancos de dados de documentos são úteis para workloads que exigem um esquema flexível para desenvolvimento rápido e iterativo. Por exemplo, casos de uso do Amazon DocumentDB, consulte Casos de uso de um banco de dados.

A seguir estão alguns exemplos de casos de uso para os quais os clusters elásticos podem fornecer vantagens significativas:

Perfis de usuário

Como os bancos de dados de documentos têm um esquema flexível, eles podem armazenar documentos com atributos e valores de dados diferentes em escala. Os clusters elásticos são uma solução prática para perfis online nos quais usuários diferentes fornecem tipos de informações diferentes. Suponha que suas aplicações ofereçam suporte a centenas de milhões de perfis de usuário. É possível usar clusters elásticos para dar suporte a essas aplicações porque eles podem ser ampliados e reduzidos para dar suporte a milhões de gravações e leituras nesses perfis de usuário. Você também pode reduzir a escala verticalmente fora dos horários de pico para reduzir custos.

Gerenciamento de conteúdo e registros históricos

Para gerenciar o conteúdo com eficiência, é necessário coletar e agregar o conteúdo de várias fontes e, em seguida, enviá-lo ao cliente. Devido ao esquema flexível, os bancos de dados de documentos são perfeitos para coletar e armazenar qualquer tipo de dados. É possível usá-los para criar e incorporar novos tipos de conteúdo, incluindo conteúdo gerado pelo usuário, como imagens, comentários e vídeos. Com o tempo, seu banco de dados pode exigir mais armazenamento. Com clusters elásticos, é possível distribuir seus dados em mais volumes de armazenamento, permitindo armazenar petabytes de dados em um único cluster.

Vantagens dos clusters elásticos

AWS integração de serviços

Os clusters elásticos do Amazon DocumentDB se integram a outros AWS serviços da mesma forma que o Amazon DocumentDB faz:

- Migração — Você pode usar o AWS Database Migration Service (DMS) para migrar do MongoDB e de outros bancos de dados relacionais para clusters elásticos do Amazon DocumentDB.
- Monitoramento — Você pode monitorar a integridade e o desempenho do seu cluster elástico usando a Amazon CloudWatch.
- Segurança — Você pode configurar a autenticação e a autorização por meio do AWS Identity and Access Management (IAM) para gerenciar seus recursos de cluster elástico, como criar e atualizar um cluster elástico, e usar o Amazon VPC para VPC-only conexões seguras. Não há suporte para login de autenticação do IAM para clusters elásticos.
- Gerenciamento de dados — Você pode usar AWS Glue para importar e exportar dados from/to outros AWS serviços, como Amazon S3, Amazon Redshift e Amazon OpenSearch Service.

Disponibilidade de regiões e versões do cluster elástico

Disponibilidade regional para clusters elásticos

A tabela a seguir mostra Regiões da AWS onde os clusters elásticos do Amazon DocumentDB estão atualmente disponíveis e o endpoint para cada região.

Nome da região	Região	Zonas de disponibilidade
Leste dos EUA (Norte da Virgínia)	`us-east-1`	5
Leste dos EUA (Ohio)	`us-east-2`	3
Oeste dos EUA (Oregon)	`us-west-2`	3
Ásia-Pacífico (Hong Kong)	`ap-east-1`	3
Ásia-Pacífico (Mumbai)	`ap-south-1`	3
Ásia-Pacífico (Seul)	`ap-northeast-2`	3
Ásia-Pacífico (Singapura)	`ap-southeast-1`	3
Ásia-Pacífico (Sydney)	`ap-southeast-2`	3
Ásia-Pacífico (Tóquio)	`ap-northeast-1`	3

Nome da região	Região	Zonas de disponibilidade
Canadá (Central)	`ca-central-1`	3
América do Sul (São Paulo)	`sa-east-1`	3
Europa (Frankfurt)	`eu-central-1`	3
Europa (Irlanda)	`eu-west-1`	3
Europa (Londres)	`eu-west-2`	3
Europa (Milão)	`eu-south-1`	3
Europa (Paris)	`eu-west-3`	3

Disponibilidade da versão

Os clusters elásticos oferecem suporte ao protocolo de conexão compatível com MongoDB 5.0. Para saber as diferenças entre clusters baseados em instâncias do Amazon DocumentDB 4.0 e clusters elásticos, consulte Diferenças funcionais entre o Amazon DocumentDB 4.0 e clusters elásticos. Observação: os clusters elásticos não são compatíveis com o Amazon DocumentDB 8.0

Limitações

Gerenciamento de clusters elásticos

Não há suporte para os recursos e capacidades de gerenciamento de cluster a seguir nesta versão:

- Porta personalizada (há suporte somente para 27017)
- Capacidade de usar perfis e usuários do IAM para se autenticar no banco de dados
- Capacidade de criar clusters globais
- Eventos do Amazon DocumentDB existentes e como se inscrever neles
- Fragmentação de alcance
- Fragmentar a coleção existente
- Multi-field chave fragmentada

- Alterar chave de fragmento
- Point-in-time restaurar
- Clonagem
- Insights de Performance

> **Note**
>
> Para obter informações sobre cotas elásticas de clusters, consulteCotas do Amazon DocumentDB.

Operações de gravação e de consulta

Não há suporte para os comandos e recursos de operação de consulta e gravação a seguir nesta versão:

- Comandos DDL durante operações de escalabilidade
- Profiler
- Grupos de parâmetros
- AWS Config
- AWS Backup

Gerenciamento de coleções e índices

Não há suporte para os recursos de gerenciamento de coleções e índices a seguir nesta versão:

- Índices exclusivos
- Índices parciais
- Índice de texto
- Índices de vetores
- Compactação de documentos

Administração e diagnóstico

Não há suporte para os comandos e recursos de administração e diagnóstico a seguir nesta versão:

- AWS Secrets Manager
- Role-based-access-control (RBAC) funções personalizadas.
- Ao conectar, a preocupação de gravação de 0 não é suportada.
- Alteração de sub-redes pertencentes a uma VPC que não está atualmente atribuída a um cluster elástico existente.

Opt-in features

Não há suporte para os recursos opcionais do Amazon DocumentDB a seguir nesta versão:

- Transações ACID
- DDL/DML auditoria
- Change streams
- Comandos de sessão

Clusters elásticos do Amazon DocumentDB: como funcionam

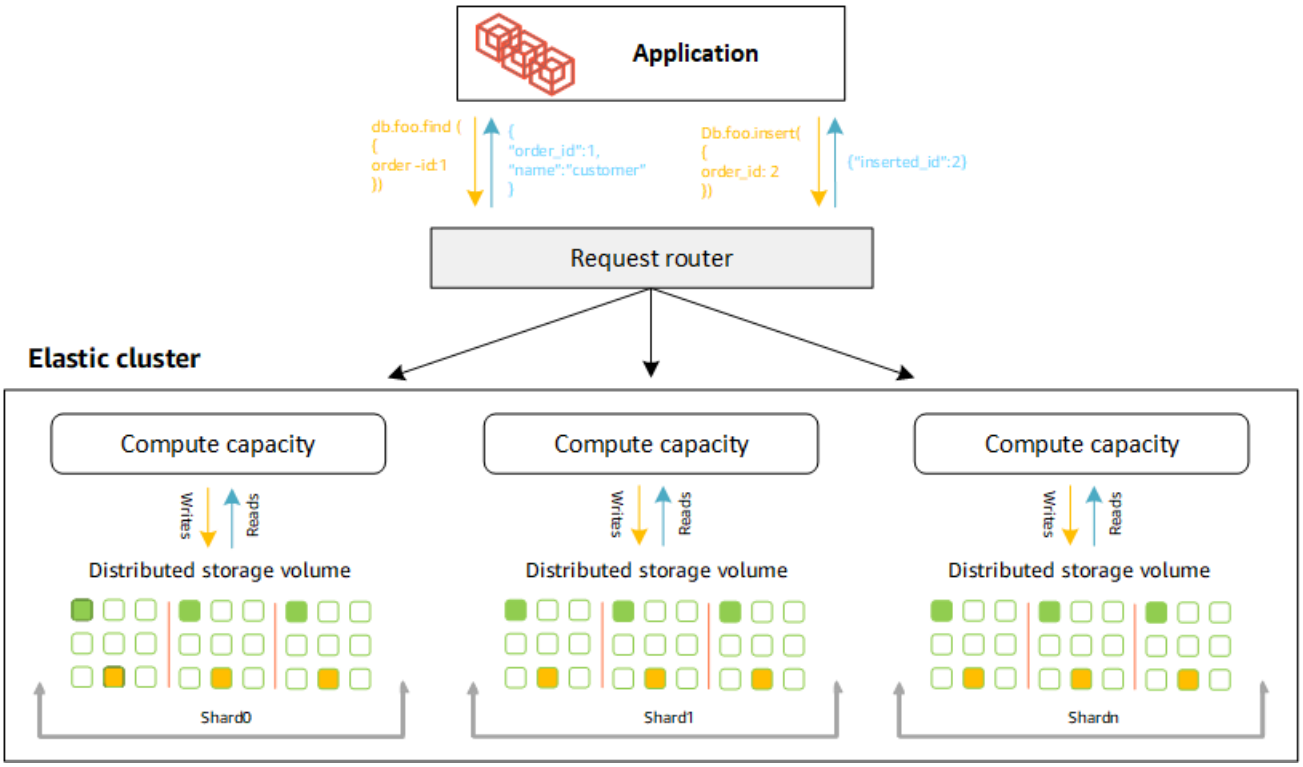
Os tópicos desta seção fornecem informações sobre os mecanismos e funções que alimentam os clusteres elásticos do Amazon DocumentDB.

Tópicos

- Fragmentação de clusters elásticos do Amazon DocumentDB
- Migração de clusters elásticos
- Escalabilidade dos clusters elásticos
- Confiabilidade dos clusters elásticos
- Armazenamento e disponibilidade de clusters elásticos
- Diferenças funcionais entre o Amazon DocumentDB 4.0 e clusters elásticos

Fragmentação de clusters elásticos do Amazon DocumentDB

Os clusters elásticos do Amazon DocumentDB usam fragmentação baseada em hash para particionar dados em um sistema de armazenamento distribuído. A fragmentação, também conhecida como particionamento, divide grandes conjuntos de dados em pequenos conjuntos de dados em vários nós, permitindo aumentar a escala horizontalmente do seu banco de dados além dos limites de escala vertical. Os clusters elásticos usam a separação, ou “desacoplamento”, de computação e armazenamento no Amazon DocumentDB, permitindo que você escale independentemente um do outro. Em vez de reparticionar as coleções movendo pequenos pedaços de dados entre os nós de computação, os clusters elásticos copiam os dados de forma eficiente dentro do sistema de armazenamento distribuído.



Definições de fragmentos

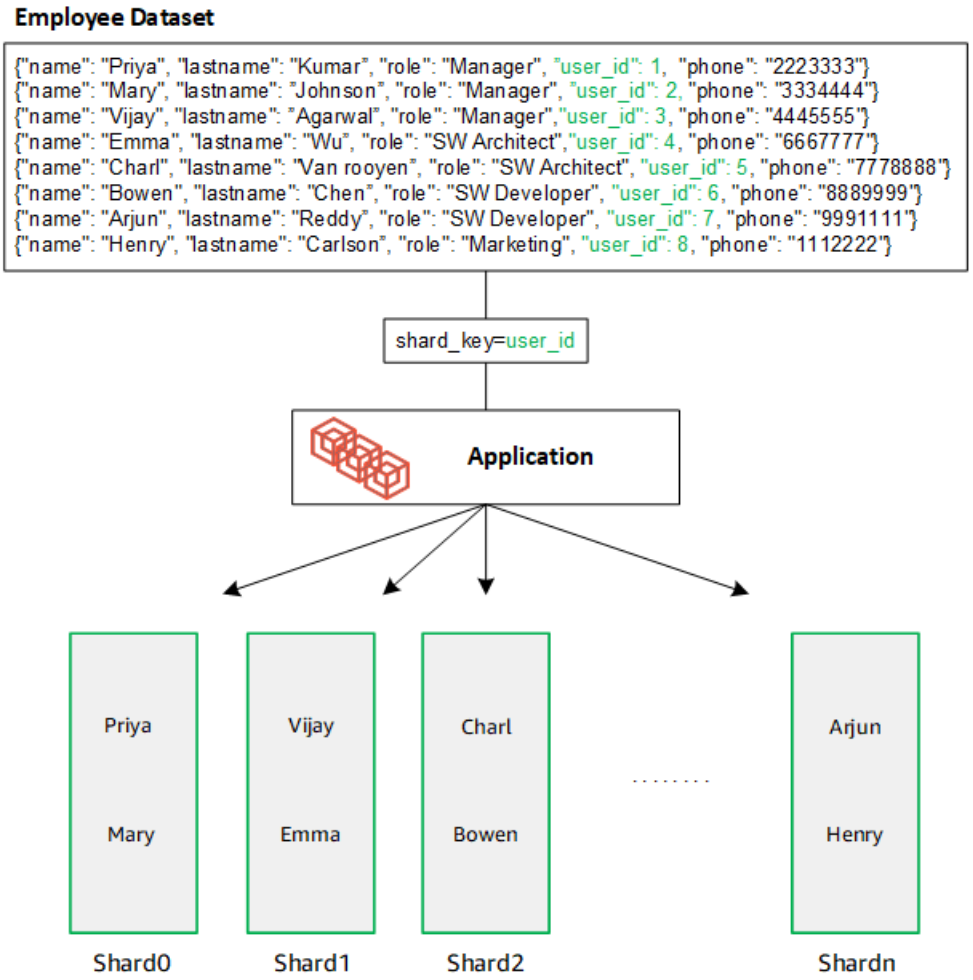
Definições da nomenclatura de fragmentos:

- Fragmento — Um fragmento fornece computação para um cluster elástico. Ele terá uma única instância de gravador e de 0 a 15 réplicas de leitura. Por padrão, um fragmento terá duas instâncias: um gravador e uma única réplica de leitura. É possível configurar no máximo 32 fragmentos e cada instância de fragmento pode ter no máximo 64 vCPUs.

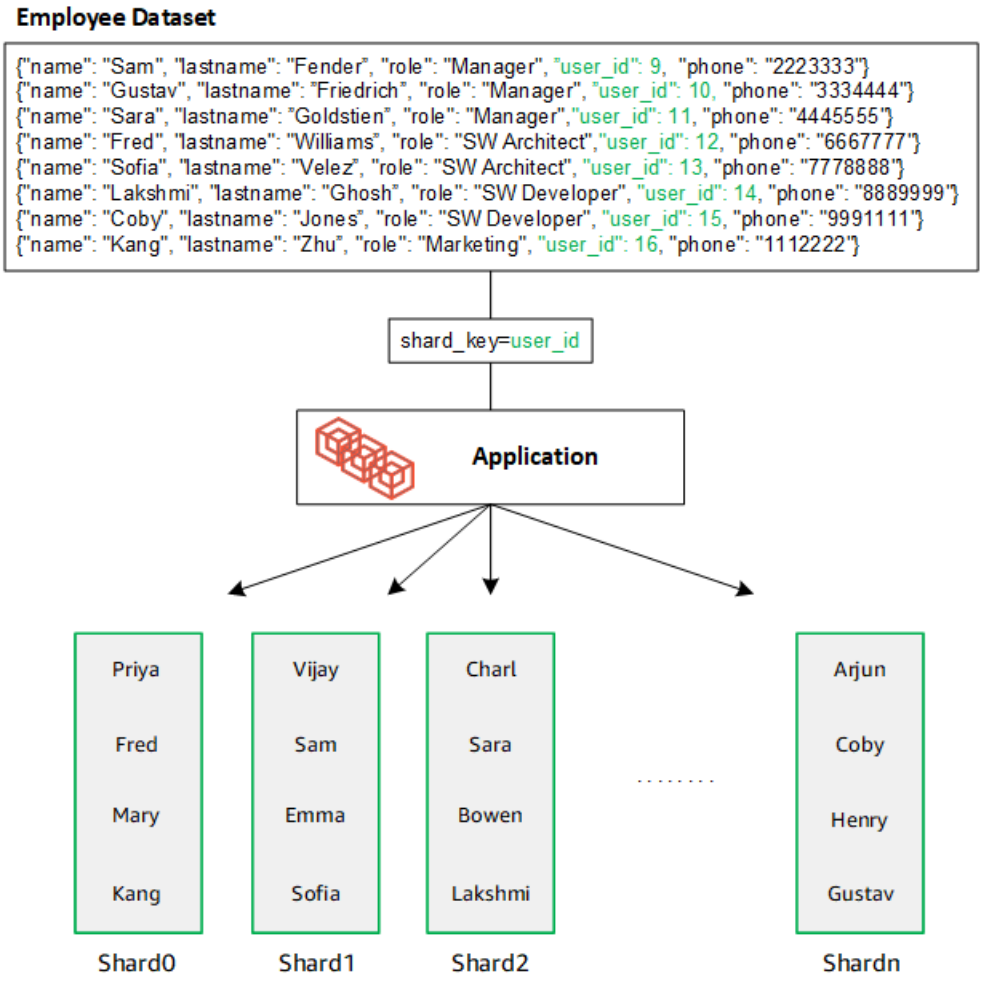
- Chave de fragmento — Uma chave de fragmento é um campo obrigatório em seus documentos JSON em coleções fragmentadas que os clusters elásticos usam para distribuir tráfego de leitura e gravação para o fragmento correspondente.
- Coleção fragmentada: uma coleção fragmentada é uma coleção cujos dados são distribuídos em um cluster elástico em partições de dados.
- Partição — Uma partição é uma parte lógica dos dados fragmentados. Quando você cria uma coleção fragmentada, os dados são organizados em partições dentro de cada fragmento automaticamente com base na chave do fragmento. Cada fragmento tem várias partições.

Distribuir dados entre fragmentos configurados

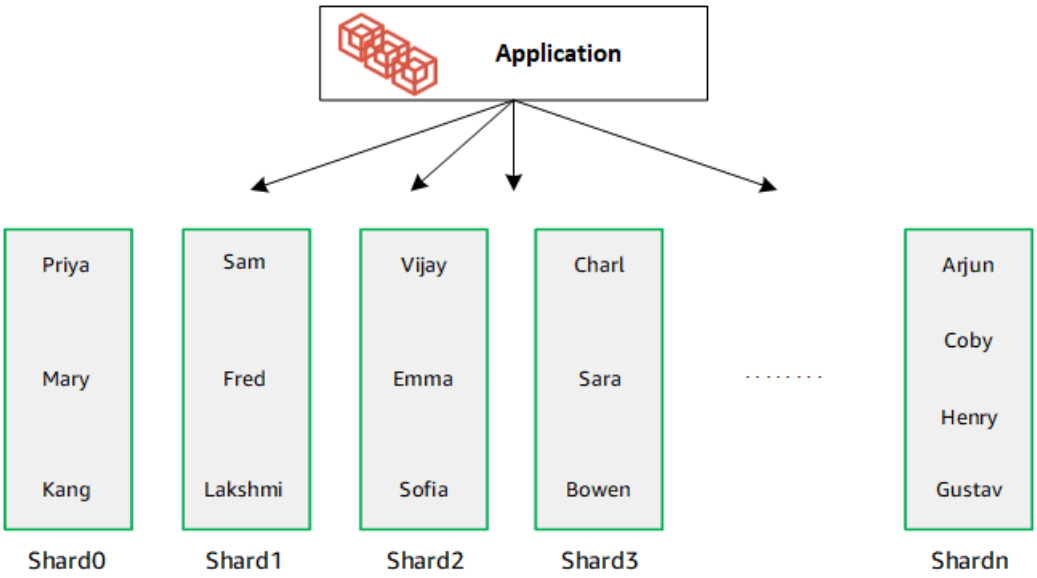
Crie uma chave de fragmento que tenha muitos valores exclusivos. Uma boa chave de fragmento particionará uniformemente seus dados entre os fragmentos subjacentes, oferecendo à sua workload a melhor throughput e desempenho. O exemplo a seguir são dados de nome de funcionário que usam uma chave de fragmento chamada “user_id”:



O DocumentDB usa fragmentação de hash para particionar seus dados em fragmentos subjacentes. Dados adicionais são inseridos e distribuídos da mesma forma:



Quando você aumentar a escala de seu banco de dados horizontalmente, adicionando fragmentos, o Amazon DocumentDB redistribui automaticamente os dados:



Migração de clusters elásticos

O Amazon DocumentDB oferece suporte à migração de dados fragmentados do MongoDB para clusters elásticos. Há suporte para métodos de migração offline, online e híbrida. Para obter mais informações, consulte Migração e atualização do Amazon DocumentDB.

Escalabilidade dos clusters elásticos

Os clusters elásticos do Amazon DocumentDB oferecem a capacidade de aumentar o número de fragmentos (aumentar a escala horizontalmente) em seu cluster elástico e o número de vCPUs aplicados a cada fragmento (aumentar a escala verticalmente). Você também pode reduzir o número de fragmentos e a capacidade computacional (vCPUs) conforme necessário.

Para obter as melhores práticas de escalabilidade, consulte Escalar clusters elásticos.

> **Note**
>
> Cluster-level o dimensionamento também está disponível. Para obter mais informações, consulte Escalar clusters do Amazon DocumentDB.

Confiabilidade dos clusters elásticos

O Amazon DocumentDB foi projetado para ser confiável, durável e tolerante a falhas. Para melhorar a disponibilidade, os clusters elásticos implantam dois nós por fragmento colocados em diferentes

zonas de disponibilidade. O Amazon DocumentDB inclui vários recursos automáticos que o tornam uma solução de banco de dados confiável. Para obter mais informações, consulte Confiabilidade do Amazon DocumentDB.

Armazenamento e disponibilidade de clusters elásticos

Os dados do Amazon DocumentDB são armazenados em um volume de cluster, que é um único volume virtual único que usa unidades de estado sólido (SSDs). Um volume de cluster consiste em seis cópias de seus dados, que são replicadas automaticamente em várias zonas de disponibilidade em uma única AWS região. Essa replicação ajuda a garantir que seus dados sejam resilientes, com menor possibilidade de perda de dados. Isso também ajuda a garantir que o cluster esteja mais disponível durante um failover, pois as cópias dos dados já existem em outras zonas de disponibilidade. Para obter mais detalhes sobre armazenamento, alta disponibilidade e replicação, consulte Amazon DocumentDB: como funciona.

Diferenças funcionais entre o Amazon DocumentDB 4.0 e clusters elásticos

As seguintes diferenças funcionais existem entre o Amazon DocumentDB 4.0 e os clusters elásticos.

- Os resultados de `top` e `collStats` são particionados por fragmentos. Para coleções fragmentadas, os dados são distribuídos entre várias partições e `collStats` informa `collScans` agregados a partir das partições.
- As estatísticas de `top` e `collStats` para coleções fragmentadas são redefinidas quando a contagem de fragmentos do cluster é alterada.
- A função integrada de backup agora é compatível com `serverStatus`. Ação: desenvolvedores e aplicações com função de backup podem coletar estatísticas sobre o estado do cluster Amazon DocumentDB.
- O campo `SecondaryDelaySecs` substitui `slaveDelay` na saída `replSetGetConfig`.
- O comando `hello` substitui `isMaster` - `hello` retorna um documento que descreve a função do cluster elástico.
- O operador `$elemMatch` em clusters elásticos só corresponde aos documentos no primeiro nível de aninhamento de uma matriz. No Amazon DocumentDB 4.0, o operador percorre todos os níveis antes de devolver os documentos correspondentes. Por exemplo:

```
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 ] } ] ] }
```

- A projeção “$” no Amazon DocumentDB 4.0 retorna todos os documentos com todos os campos. Com clusters elásticos, o comando `find` com uma projeção “$” retorna documentos que correspondem ao parâmetro de consulta contendo somente o campo que corresponde à projeção “$”.
- Em clusters elásticos, os comandos `find` com parâmetros de consulta `$regex` e `$options` retornam um erro: “Não é possível definir opções em $regex e $options”.

- Com clusters elásticos, `$indexOfCP` agora retorna “-1" quando:
  - a substring não foi encontrada no `string expression`, ou
  - `start` é um número maior que `end`, ou
  - `start` é um número maior que o comprimento do byte da string.

  No Amazon DocumentDB 4.0, `$indexOfCP` retorna “0" quando a posição `start` é um número maior que `end` ou o comprimento do byte da string.
- Com clusters elásticos, as operações de projeção em `_id fields`, por exemplo, `{"_id.nestedField" : 1}`, retornam documentos que incluem apenas o campo projetado. Já

no Amazon DocumentDB 4.0, os comandos de projeção de campo aninhados não filtram nenhum documento.

Como iniciar com clusters elásticos do Amazon DocumentDB

Esta seção de introdução explica como é possível criar e consultar seu primeiro cluster elástico.

Há muitas formas de se conectar e começar a usar o Amazon DocumentDB. O procedimento a seguir é a maneira mais rápida, simples e fácil para os usuários começarem a usar nosso poderoso banco de dados de documentos. Este guia usa AWS CloudShell para se conectar e consultar seu cluster do Amazon DocumentDB diretamente a partir do Console de gerenciamento da AWS. Novos clientes qualificados para o nível AWS gratuito podem usar o Amazon DocumentDB CloudShell gratuitamente. Se seu AWS CloudShell ambiente ou cluster elástico do Amazon DocumentDB fizer uso de recursos além do nível gratuito, você pagará as AWS taxas normais por esses recursos. Este guia ajudará você a começar a usar o Amazon DocumentDB em menos de 5 minutos.

Tópicos

- Pré-requisitos
- Etapa 1: Criar um cluster do ElastiCache
- Etapa 2: conectar a seu cluster elástico
- Etapa 3: fragmentar sua coleção; inserir e consultar dados
- Etapa 4: explorar

Pré-requisitos

Antes de criar o primeiro cluster do Amazon DocumentDB, você deve fazer o seguinte:

Crie uma conta (AWS) da Amazon Web Services

Antes de começar a usar o Amazon DocumentDB, você deve ter uma conta da Amazon Web Services (AWS). A AWS conta é gratuita. Você paga apenas pelos serviços e recursos usados.

Se você não tiver um Conta da AWS, conclua as etapas a seguir para criar um.

Para se inscrever em um Conta da AWS

1. Aberto https://signin.aws.amazon.com/signup? request_type=registrar.

2. Siga as instruções online.

   Parte do procedimento de inscrição envolve receber uma chamada telefônica ou uma mensagem de texto e inserir um código de verificação pelo teclado do telefone.

   Quando você se inscreve em um Conta da AWS, um Usuário raiz da conta da AWS é criado. O usuário-raiz tem acesso a todos os Serviços da AWS e recursos na conta. Como prática recomendada de segurança, atribua o acesso administrativo a um usuário e use somente o usuário-raiz para executar tarefas que exigem acesso de usuário-raiz.

Configure as permissões necessárias AWS Identity and Access Management (IAM).

O acesso para gerenciar recursos do Amazon DocumentDB, como clusters, instâncias e grupos de parâmetros de cluster, exige credenciais que AWS possam ser usadas para autenticar suas solicitações. Para obter mais informações, consulte Gerenciamento de identidade e Gerenciamento de acesso para o Amazon DocumentDB.

1. Na barra de pesquisa do Console de gerenciamento da AWS, insira **`IAM`** e selecione IAM no menu suspenso.
2. Depois de chegar ao console do IAM, selecione Usuários no painel de navegação.
3. Selecione o nome de usuário.
4. Escolha Adicionar permissões.
5. Selecione Anexar políticas diretamente.
6. Digite `AmazonDocDBElasticFullAccess` na barra de pesquisa e selecione-a quando ela aparecer nos resultados da pesquisa.
7. Escolha Próximo.
8. Escolha Adicionar permissões.

> **Note**
>
> Sua AWS conta inclui uma VPC padrão em cada região. Se você optar por usar uma Amazon VPC, conclua as etapas no tópico Criação de uma Amazon VPC no Guia do usuário da Amazon VPC.

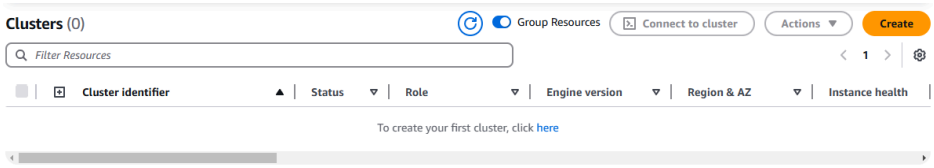
Etapa 1: Criar um cluster do ElastiCache

Nesta seção, explicamos como criar um novo cluster elástico usando Console de gerenciamento da AWS ou AWS CLI com as instruções a seguir.

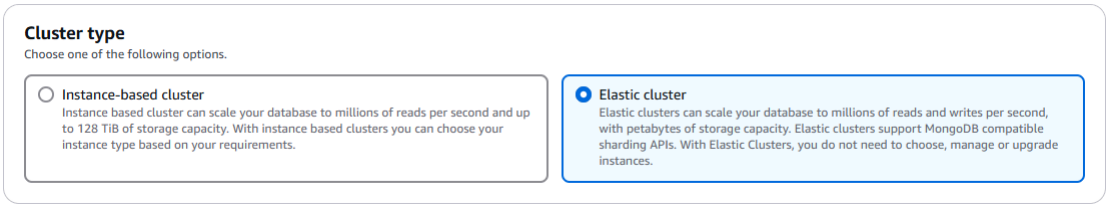
Using the Console de gerenciamento da AWS

Para criar uma configuração de cluster elástico usando o Console de gerenciamento da AWS:

1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No Console de gerenciamento do Amazon DocumentDB, em Clusters, escolha Criar.



3. Na página Criar cluster do Amazon DocumentDB, na seção Tipo de cluster, escolha Cluster elástico.



4. Na seção Configuração, configure o seguinte:

   a. No campo Nome do cluster, insira um identificador de cluster exclusivo (seguindo os requisitos de nomenclatura abaixo do campo).

   b. No campo Contagem de fragmentos, insira o número de fragmentos que você deseja em seu cluster. O número máximo de fragmentos por cluster é 32.

   > **Note**
   >
   > Dois nós serão implantados para cada fragmento. Ambos os nós terão a mesma capacidade de fragmentos.

   c. No campo Contagem de instâncias de fragmento, escolha o número de instâncias de réplica que você deseja associar a cada fragmento. O número máximo de instâncias de fragmento é 16, em incrementos de 1. Todas as instâncias de réplica têm a mesma

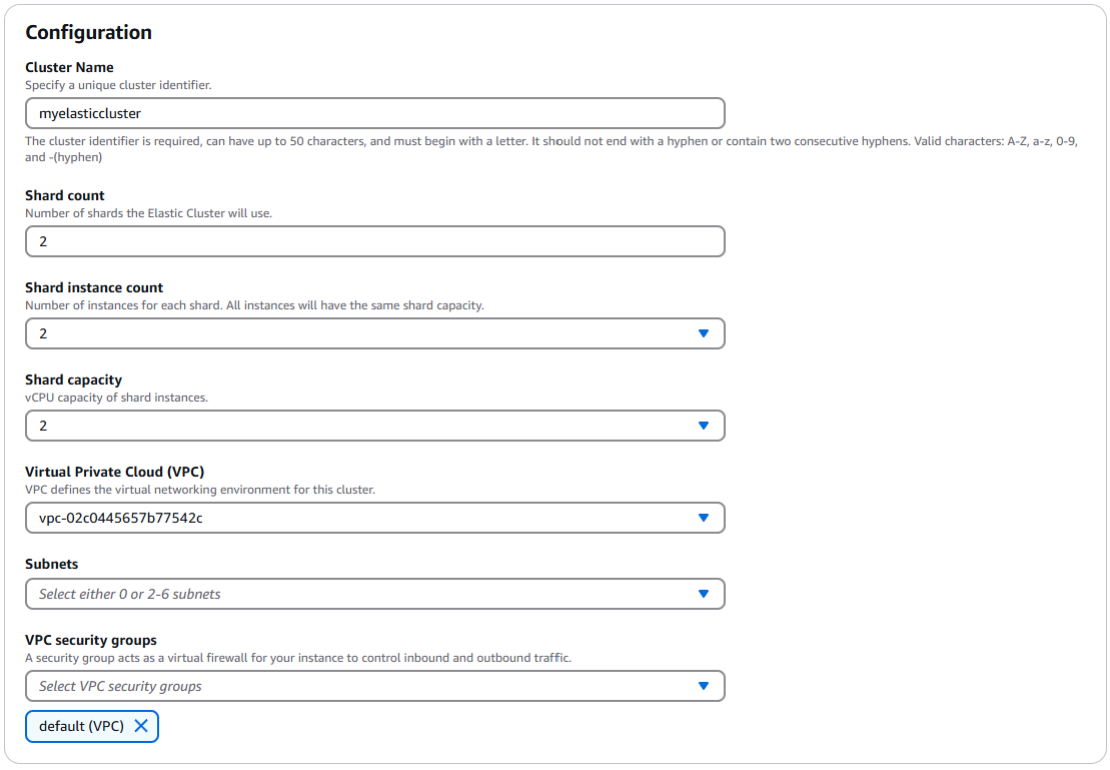
capacidade de fragmentos, conforme definido no campo a seguir. Para fins de teste, o valor padrão de 2 deve ser suficiente.

> **Note**
>
> O número de instâncias de réplica se aplica a todos os fragmentos no cluster elástico. O valor 1 de contagem de instâncias de fragmento significa que há uma instância gravadora, e todas as instâncias adicionais são réplicas que podem ser usadas para leituras e para melhorar a disponibilidade. Para fins de teste, o valor padrão de 2 deve ser suficiente.

d. No campo Capacidade de fragmentos, escolha o número de CPUs virtuais (vCPUs) que você deseja associar a cada instância de fragmento. O número máximo de vCPUs por instância de fragmento é 64. Os valores permitidos são 2, 4, 8, 16, 32, 64. Para fins de teste, o valor padrão de 2 deve ser suficiente.

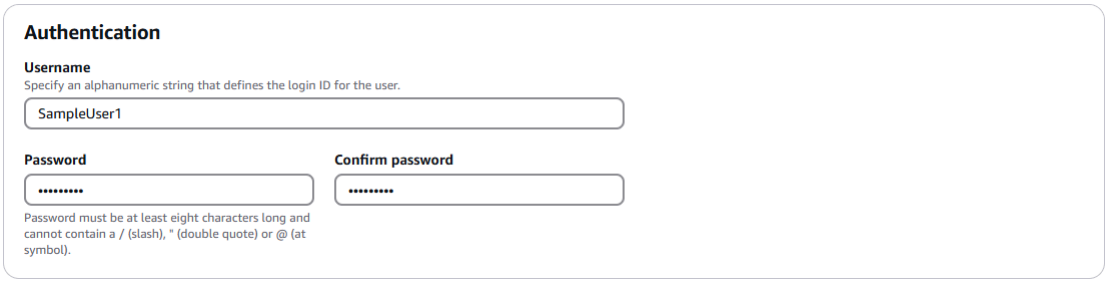
e. No campo Nuvem Privada Virtual (VPC), escolha uma VPC na lista suspensa.

f. Em Sub-redes e Grupos de segurança de VPC, é possível usar os padrões ou selecionar três sub-redes de sua escolha e até três Grupos de segurança de VPC (o mínimo é um).



5. Na seção Autenticação, insira uma string que identifique o nome de login do usuário principal no campo Nome de usuário.

   No campo Senha, insira uma senha exclusiva que esteja em conformidade com as instruções e, em seguida, confirme-a.



6. Na seção Criptografia, mantenha a configuração padrão (Chave padrão).

   Opcionalmente, você pode inserir um AWS KMS key ARN criado por você. Para obter mais informações, consulte Criptografia de dados em repouso para clusters elásticos do Amazon DocumentDB.

> **Important**
>
> A criptografia deve ser habilitada para clusters elásticos.

7. Na seção Backup, edite os campos de acordo com os requisitos do seu backup. Para fins de teste, é possível manter as configurações padrão.



   a. Período de retenção de backup— na lista, selecione o número de dias para manter os backups automáticos desse cluster antes de excluí-los.

   b. Janela de backup— defina o tempo e duração diários durante os quais o Amazon DocumentDB fará backups desse cluster.

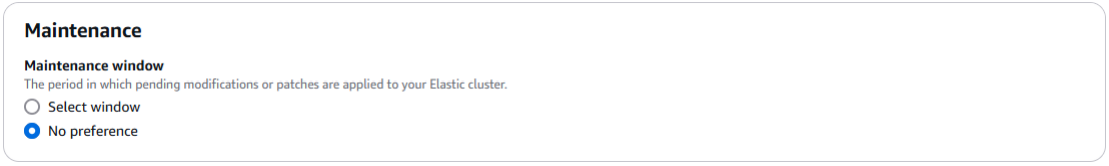
      i. Escolha Selecionar janela se quiser configurar a hora e a duração em que os backups são criados.

Hora de início: na primeira lista, escolha a hora de início (UTC) para seus backups automáticos. Na segunda lista, escolha o minuto da hora em que você deseja que os backups automáticos sejam iniciados.

Duração— na lista, selecione a quantidade de horas a serem alocadas na criação de backups automáticos.

ii. Escolha Sem preferência se quiser que o Amazon DocumentDB escolha a hora e a duração em que os backups são criados.

8. Na seção Manutenção, escolha o dia, a hora e a duração em que as modificações ou os patches serão aplicados ao seu cluster. Para fins de teste, é possível manter as configurações padrão.



9. Selecione Criar cluster.

Agora, seu cluster elástico está sendo provisionado. Esse processo pode levar até alguns minutos. É possível se conectar ao seu cluster quando o status do cluster elástico for exibido como Disponível na lista Clusters.

Using the AWS CLI

Para criar um cluster elástico usando o AWS CLI, use a `create-cluster` operação com os seguintes parâmetros:

- --cluster-name—Obrigatório. O nome atual do cluster elástico de escala, conforme inserido durante a criação ou a última modificação.
- --shard-capacity—Obrigatório. O número de vCPUs atribuído a cada fragmento. O máximo é 64. Os valores permitidos são 2, 4, 8, 16, 32, 64.
- --shard-count—Obrigatório. O número de fragmentos atribuídos ao cluster. O máximo é 32.
- --shard-instance-count—Opcional. O número de instâncias de réplica que se aplicam a todos os fragmentos desse cluster. O máximo é 16.
- --admin-user-name—Obrigatório. O nome de usuário associado ao usuário administrativo.
- --admin-user-password—Obrigatório. A senha associada ao usuário administrativo.

- --auth-type—Obrigatório. O tipo de autenticação usado para determinar onde buscar a senha usada para acessar o cluster elástico. Os tipos válidos são `PLAIN_TEXT` ou `SECRET_ARN`.
- --vpc-security-group-ids—Opcional. Configure uma lista de grupos de segurança da VPC do EC2 a serem associados a esse cluster.
- --preferred-maintenance-window—Opcional. O período semanal durante o qual pode ocorrer a manutenção do sistema, em Tempo Universal Coordenado (UTC).

  O formato é: `ddd:hh24:mi-ddd:hh24:mi`. Dias válidos (ddd): Mon, Tue, Wed, Thu, Fri, Sat, Sun

  O padrão é uma janela de 30 minutos selecionada aleatoriamente a partir de um bloco de tempo de 8 horas para cada região da Amazon Web Services, ocorrendo em um dia da semana aleatório.

  Janela mínima de 30 minutos.
- --kms-key-id—Opcional. O identificador da chave do KMS para um cluster criptografado.

  O identificador da chave KMS é o nome de recurso da Amazon (ARN) para a chave de AWS KMS criptografia. Se você estiver criando um cluster com a mesma conta da Amazon Web Services que tem a chave de criptografia KMS usada para criptografar o novo cluster, use o apelido da chave do KMS em vez de o ARN da chave de criptografia do KMS.

  Se uma chave de criptografia não for especificada em KmsKeyId e se o `StorageEncrypted` parâmetro for verdadeiro, o Amazon DocumentDB usará sua chave de criptografia padrão.
- --preferred-backup-window—Opcional. O intervalo de tempo diário preferencial, durante o qual os backups automatizados são criados. O padrão é uma janela de 30 minutos selecionada aleatoriamente em um bloco de 8 horas para cada uma. Região da AWS
- --backup-retention-period— opcional. O número de dias durante os quais os backups automatizados são retidos. O valor padrão é 1.
- --storage-encrypted—Opcional. Configura se o cluster é criptografado ou não.

  `--no-storage-encrypted` Especifica se o cluster é criptografado.
- --subnet-ids—Opcional. Configure IDs de sub-rede de rede.

No exemplo a seguir, substitua cada *`user input placeholder`* por suas próprias informações.

> **Note**
>
> Os exemplos a seguir incluem a criação de uma chave KMS específica. Para usar a chave KMS padrão, não inclua o parâmetro `--kms-key-id`.

Para Linux, macOS ou 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
```

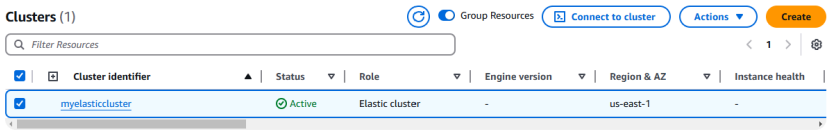
Para Windows:

```
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
```

Etapa 2: conectar a seu cluster elástico

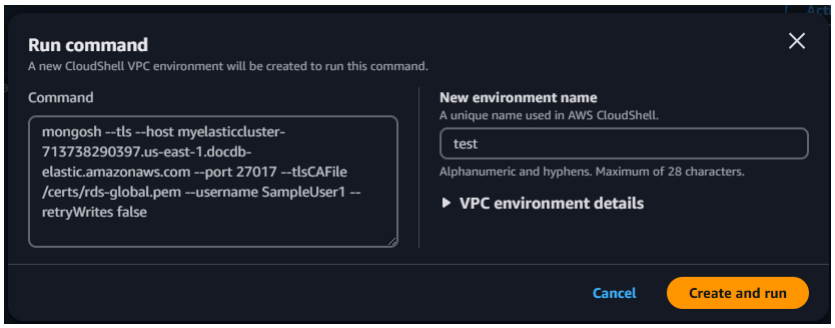
Conecte-se ao seu cluster elástico Amazon DocumentDB usando AWS CloudShell.

1. No console de gerenciamento do Amazon DocumentDB, em Clusters, localize o cluster elástico que você criou. Escolha o cluster clicando na caixa de seleção ao lado dele.



2. Escolha Conectar ao cluster (que fica ao lado do menu suspenso Ações). Esse botão será habilitado somente depois que você clicar na caixa de seleção ao lado do seu cluster e o status do cluster for exibido como Disponível. A tela de comando CloudShell Executar é exibida.

3. No campo Nome do novo ambiente, insira um nome exclusivo, como "teste" e escolha Criar e executar. Os detalhes do ambiente da VPC são configurados automaticamente para o banco de dados do Amazon DocumentDB.



4. Quando solicitado, insira a senha que você criou na Etapa 1: criação de um cluster elástico Amazon DocumentDB (subetapa 5).



Depois de você inserir sua senha e seu prompt se tornar `direct: mongos] <env-name>>`, a conexão ao seu cluster do Amazon DocumentDB será concluída com êxito.

> **Note**
>
> Para obter mais informações sobre solução de problemas, consulte Solução de problemas do Amazon DocumentDB.

Etapa 3: fragmentar sua coleção; inserir e consultar dados

Clusters elásticos adicionam suporte para fragmentação no Amazon DocumentDB. Agora que você está conectado ao seu cluster, é possível fragmentar o cluster, inserir dados e executar algumas consultas.

1. Para fragmentar uma coleção, digite o seguinte:

```
sh.shardCollection("db.Employee1" , { "Employeeid" : "hashed" })
```

2. Para inserir um único documento, digite o seguinte:

```
db.Employee1.insertOne({"Employeeid":1, "Name":"Joe", "LastName": "Bruin", "level":
 1 })
```

   A seguinte saída é exibida:

```
WriteResult({ "nInserted" : 1 })
```

3. Para ler o documento que você escreveu, insira o comando `findOne()` (ele retorna apenas um único documento):

```
db.Employee1.findOne()
```

   A seguinte saída é exibida:

```
{
"_id" : ObjectId("61f344e0594fe1a1685a8151"),
"EmployeeID" : 1,
"Name" : "Joe",
"LastName" : "Bruin",
"level" : 1
}
```

4. Para realizar mais algumas consultas, considere um caso de uso de perfil de jogo. Primeiro, insira algumas entradas em uma coleção intitulada "Funcionário". Insira o seguinte:

```
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 }
])
```

   A seguinte saída é exibida:

```
{ acknowledged: true,
     insertedIds: {
        '0': ObjectId('679d02cd6b5a0581be78bcbd'),
        '1': ObjectId('679d02cd6b5a0581be78bcbe'),
        '2': ObjectId('679d02cd6b5a0581be78bcbf'),
        '3': ObjectId('679d02cd6b5a0581be78bcc0')
    }
}
```

5. Para retornar todos os documentos na coleção de perfis, insira o comando `find`():

```
db.Employee.find()
```

   Os dados inseridos na etapa 4 são exibidos.

6. Para consultar um único documento, inclua um filtro (por exemplo: "Katie"). Insira o seguinte:

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

   A seguinte saída é exibida:

```
[

   {

     _id: ObjectId('679d02cd6b5a0581be78bcc0'),

     Employeeid: 4,
```

```
     name: 'Katie',

     lastname: 'Schaper',

     level: 3

   }

]
```

7. Para encontrar um perfil e modificá-lo, digite o comando `findAndModify`. Neste exemplo, o funcionário "Matt" recebe um nível mais alto de "14":

```
db.Employee.findAndModify({
    query: { "Employeeid" : 1, "name" : "Matt"},
    update: { "Employeeid" : 1, "name" : "Matt", "lastname" : "Winkle", "level" :
 14 }
})
```

   A seguinte saída é exibida (observe que o nível ainda não foi alterado):

```
{

   _id: ObjectId('679d02cd6b5a0581be78bcbd'),

   Employeeid: 1,

   name: 'Matt',

   lastname: 'Winkle',

   level: 12

}
```

8. Para verificar o aumento do nível, insira a seguinte consulta:

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

   A seguinte saída é exibida:

```
[
```

```
    {

    _id: ObjectId('679d02cd6b5a0581be78bcbd'),

    Employeeid: 1,

    name: 'Matt',

    lastname: 'Winkle',

    level: 14

    }
]
```

Etapa 4: explorar

Parabéns! Você concluiu com êxito o procedimento Conceitos básicos dos clusters elásticos do Amazon DocumentDB.

E depois? Saiba como aproveitar ao máximo esse banco de dados com alguns de seus atributos populares:

- Práticas recomendadas para clusters elásticos do Amazon DocumentDB
- Gerenciamento de clusters elásticos do Amazon DocumentDB

> **Note**
>
> O cluster elástico que você criou a partir desse procedimento introdutório continuará gerando custos, a menos que seja excluído. Para obter instruções, consulte Como excluir um cluster elástico.

Práticas recomendadas para clusters elásticos do Amazon DocumentDB

Conheça as práticas recomendadas para trabalhar com clusters elásticos do Amazon DocumentDB. Todas as práticas recomendadas para clusters Amazon DocumentDB baseados em instâncias

também se aplicam a clusters elásticos. Essa seção é continuamente atualizada conforme novas melhores práticas são identificadas.

Tópicos

- Selecionar chaves de fragmento
- Gerenciamento de conexões
- Coleções não fragmentadas
- Escalar clusters elásticos
- Monitoramento de clusters elásticos

Selecionar chaves de fragmento

A lista a seguir descreve as diretrizes para criar chaves de fragmento.

- Use uma chave de hash distribuída uniformemente para distribuir seus dados em todos os fragmentos do seu cluster (evite teclas de atalho).
- Use sua chave de fragmento em todas as solicitações read/update /delete para evitar consultas dispersas.
- Evite chaves fragmentadas aninhadas ao fazer operações read/update /delete.
- Ao fazer operações em lote, defina `ordered` como falso para que todos os fragmentos possam ser executados paralelamente e melhorar as latências.

Gerenciamento de conexões

A lista a seguir descreve as diretrizes para gerenciar as conexões com seu banco de dados.

- Monitore suas contagens de conexões e a frequência com que novas conexões são abertas e fechadas.
- Distribua suas conexões em todas as sub-redes na configuração da sua aplicação. Se seu cluster estiver configurado em várias sub-redes, mas você utilizar apenas um subconjunto das sub-redes, é possível sofrer um gargalo em suas conexões máximas.

Coleções não fragmentadas

A seguir, é descrita uma diretriz para coleções não fragmentadas.

- Ao trabalhar com coleções não fragmentadas no momento da distribuição da carga, tente manter coleções não fragmentadas altamente utilizadas em bancos de dados diferentes. Os clusters elásticos do Amazon DocumentDB colocam bancos de dados em diferentes fragmentos e co-localizam coleções não fragmentadas para o mesmo banco de dados no mesmo fragmento.

Escalar clusters elásticos

A lista abaixo descreve as diretrizes para escalar seus clusters elásticos.

- As operações de escalamento podem causar um breve período de erros intermitentes no banco de dados e na rede. Quando possível, evite escalar durante os horários de pico. Tente escalar durante as janelas de manutenção.
- É preferível aumentar e diminuir a capacidade do fragmento (alterando a contagem de vCPU por fragmento) para aumentar a computação em vez de aumentar ou diminuir a contagem de fragmentos, pois é mais rápido e tem uma duração menor de erros intermitentes do banco de dados e rede.
- Ao antecipar o crescimento, prefira aumentar a contagem de fragmentos em vez de aumentar a capacidade dos mesmos. Assim você pode escalar seu cluster aumentando a capacidade de fragmentação para cenários em que você precisa escalar rapidamente.
- Monitore suas políticas de repetição do lado do cliente e tente novamente com recuo exponencial e instabilidade para evitar sobrecarregar seu banco de dados ao receber erros enquanto estiver escalando.
- Alterar o número de fragmentos pode levar algum tempo, pois os dados precisam ser movidos com segurança de um fragmento para outro. Para minimizar o risco de erros, não interrompa nem faça outras alterações no estado do cluster enquanto a alteração estiver em andamento.

Monitoramento de clusters elásticos

A lista abaixo descreve as diretrizes para monitorar seus clusters elásticos.

- Acompanhe a proporção entre o pico e a média de suas métricas por fragmento para determinar se você está gerando tráfego irregular (tem um ponto de acesso). key/hot As métricas-chave para monitorar os índices de pico em relação à média são:
  - `PrimaryInstanceCPUUtilization`
    - Pode ser monitorado em nível por fragmento.

  - Em nível do cluster, você pode monitorar a média de inclinação de p99.
- `PrimaryInstanceFreeableMemory`
  - Pode ser monitorado em nível por fragmento.
  - Em nível do cluster, você pode monitorar a média de inclinação de p99.
- `DatabaseCursorsMax`
  - Deve ser monitorado no nível por fragmento para determinar a inclinação.
- `Documents-Inserted/Updated/Returned/Deleted`
  - Deve ser monitorado no nível por fragmento para determinar a inclinação.

Gerenciamento de clusters elásticos do Amazon DocumentDB

Para gerenciar um cluster elástico do Amazon DocumentDB, é necessário ter uma política do IAM com as permissões de ambiente de gerenciamento apropriadas do Amazon DocumentDB. Essas permissões permitem criar, modificar e excluir clusters. A DocumentDBFullAccess política da Amazon fornece todas as permissões necessárias para administrar um cluster elástico do Amazon DocumentDB.

Os tópicos a seguir mostram como realizar várias tarefas ao trabalhar com clusters elásticos do Amazon DocumentDB.

Tópicos

- Como modificar configurações de clusters elásticos
- Como monitorar um cluster elástico
- Como excluir um cluster elástico
- Como gerenciar snapshots de cluster elástico
- Interromper e iniciar um cluster elástico do Amazon DocumentDB
- Manutenção dos clusters elásticos do Amazon DocumentDB

Como modificar configurações de clusters elásticos

Nesta seção, explicamos como modificar o cluster elástico usando o Console de gerenciamento da AWS ou AWS CLI com as instruções a seguir.

O principal uso da modificação do cluster é escalar fragmentos aumentando ou diminuindo a capacidade computacional de fragmentos do número and/or de fragmentos.

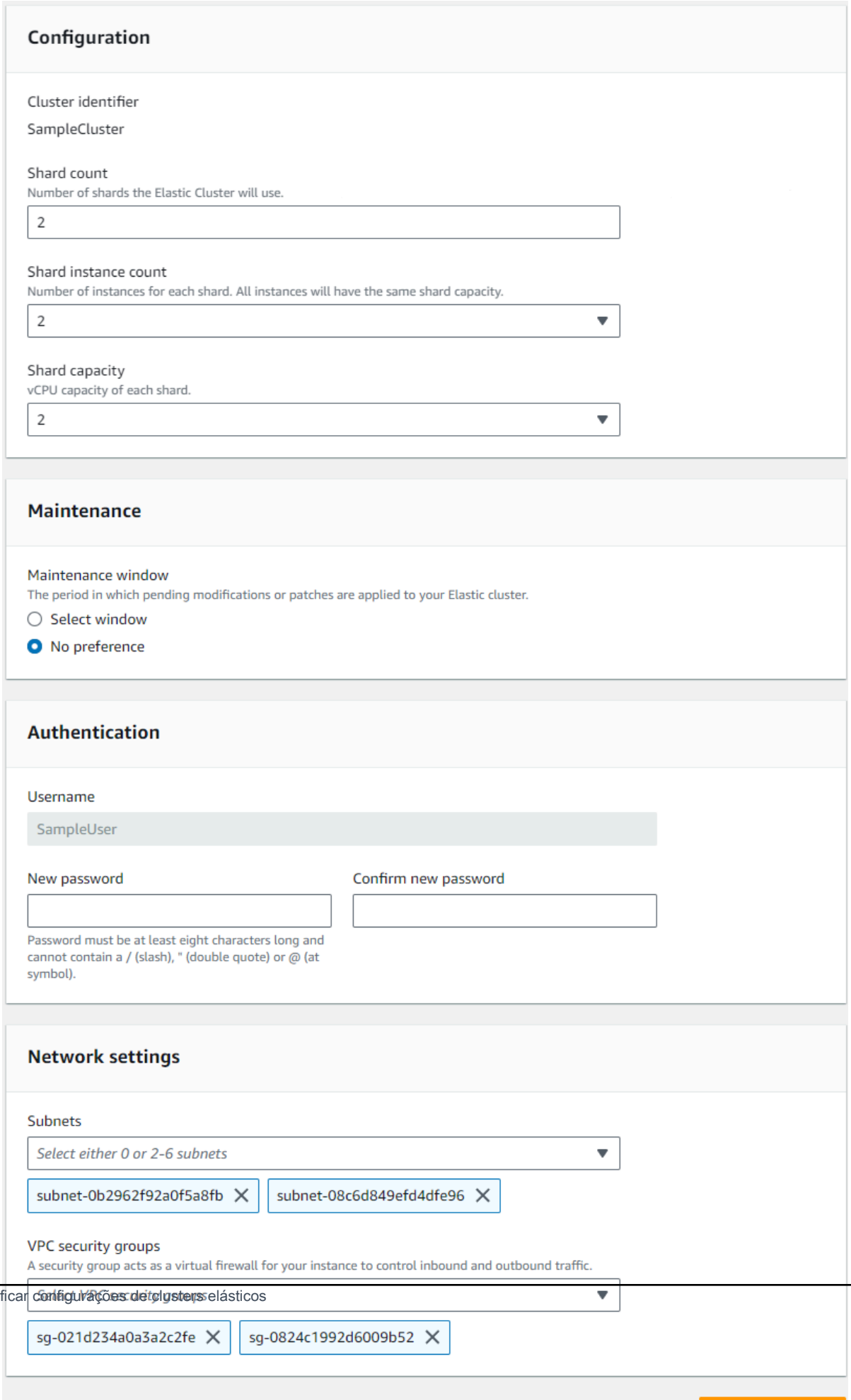
Using the Console de gerenciamento da AWS

Para modificar uma configuração de cluster elástico usando o Console de gerenciamento da AWS:

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon DocumentDB.
2. No painel de navegação, escolha Clusters.

> **Tip**
>
> Se você não visualizar o painel de navegação à esquerda da tela, selecione o ícone do menu no canto superior esquerdo da página.

3. Na lista de clusters, escolha o nome do cluster que deseja modificar na coluna Identificador do Cluster.
4. Escolha Modificar.
5. Edite os campos que você deseja alterar e selecione Modificar cluster .


Configuration
Cluster identifier
SampleCluster
Shard count
Number of shards the Elastic Cluster will use.
2
Shard instance count
Number of instances for each shard. All instances will have the same shard capacity.
2
Shard capacity
vCPU capacity of each shard.
2
Maintenance
Maintenance window
The period in which pending modifications or patches are applied to your Elastic cluster.
Select window
No preference
Authentication
Username
SampleUser
New password
Confirm new password
Password must be at least eight characters long and cannot contain a / (slash), " (double quote) or @ (at symbol).
Network settings
Subnets
Select either 0 or 2-6 subnets
subnet-0b2962f92a0f5a8fb
subnet-08c6d849efd4dfe96
VPC security groups
A security group acts as a virtual firewall for your instance to control inbound and outbound traffic.
sg-021d234a0a3a2c2fe
sg-0824c1992d6009b52

> **Note**
>
> Como alternativa, é possível acessar a caixa de diálogo Modificar cluster acessando a página Clusters, marcando a caixa ao lado do cluster e escolhendo Ações e, em seguida, Modificar.

Using the AWS CLI

Para modificar uma configuração de cluster elástico usando o AWS CLI, use a `update-cluster` operação com os seguintes parâmetros:

- **`--cluster-arn`**—Obrigatório. O identificador ARN do cluster que você deseja excluir.
- **`--shard-capacity`**—Opcional. O número de vCPUs atribuído a cada fragmento. O máximo é 64. Os valores permitidos são 2, 4, 8, 16, 32, 64.
- **`--shard-count`**—Opcional. O número de fragmentos atribuídos ao cluster. O máximo é 32.
- **`--shard-instance`**-count: opcional. O número de instâncias de réplica que se aplicam a todos os fragmentos desse cluster. O máximo é 16.
- **`--auth-type`**—Opcional. O tipo de autenticação usado para determinar onde buscar a senha usada para acessar o cluster elástico. Os tipos válidos são `PLAIN_TEXT` ou `SECRET_ARN`.
- **`--admin-user-password`**—Opcional. A senha associada ao usuário administrativo.
- **`--vpc-security-group-ids`**—Opcional. Configure uma lista de grupos de segurança do Amazon EC2 e da Nuvem privada virtual (VPC) para associar a esse cluster.
- **`--preferred-maintenance-window`**—Opcional. O período semanal durante o qual pode ocorrer a manutenção do sistema, em Tempo Universal Coordenado (UTC)

  O formato é: `ddd:hh24:mi-ddd:hh24:mi`. Dias válidos (ddd): Mon, Tue, Wed, Thu, Fri, Sat, Sun

  O padrão é uma janela de 30 minutos selecionada aleatoriamente a partir de um bloco de tempo de 8 horas para cada região da Amazon Web Services, ocorrendo em um dia da semana aleatório.

  Janela mínima de 30 minutos.
- **`--subnet-ids`**—Opcional. Configure IDs de sub-rede de rede.

No exemplo a seguir, substitua cada *user input placeholder* por suas próprias informações.

Para Linux, macOS ou 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 monitorar o status do cluster elástico após sua modificação, consulte Como monitorar um cluster elástico.

Como monitorar um cluster elástico

Nesta seção, explicamos como monitorar seu cluster elástico usando o Console de gerenciamento da AWS ou AWS CLI com as instruções a seguir.

Using the Console de gerenciamento da AWS

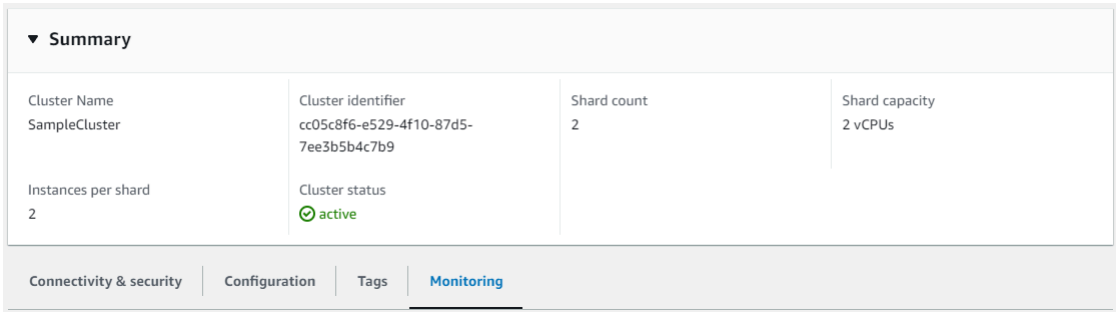
Para monitorar uma configuração de cluster elástico usando Console de gerenciamento da AWS:

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon DocumentDB.

2. No painel de navegação, escolha Clusters.

> **Tip**
>
> Se você não visualizar o painel de navegação à esquerda da tela, selecione o ícone do menu no canto superior esquerdo da página.

3. Escolha o nome do cluster que você deseja monitorar na coluna Identificador do cluster.
4. Escolha a guia Monitoramento.



Vários gráficos da Amazon CloudWatch são exibidos para as seguintes categorias de monitoramento:

- Utilização de recursos
- Throughput
- Operações
- Sistema

Você também pode acessar a Amazon CloudWatch por meio do Console de gerenciamento da AWS para configurar seu próprio ambiente de monitoramento para seus clusters elásticos.

Using the AWS CLI

Para monitorar uma configuração específica de cluster elástico usando o AWS CLI, use a `get-cluster` operação com os seguintes parâmetros:

- **`--cluster-arn`**—Obrigatório. O identificador ARN do cluster para o qual você deseja obter informações.

No exemplo a seguir, substitua cada *`user input placeholder`* por suas próprias informações.

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante à seguinte:

```
"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 obter mais informações, consulte `DescribeClusterSnapshot` na Referência da API de gerenciamento de recursos do Amazon DocumentDB.

Para visualizar os detalhes de todos os clusters elásticos usando o AWS CLI, use a `list-clusters` operação com os seguintes parâmetros:

- **`--next-token`**—Opcional. Se o número de saída de itens (`--max-results`) for menor do que o número total de itens retornados pelas chamadas de API subjacentes, a saída incluirá

um `NextToken` que pode ser passado para um comando subsequente para recuperar o próximo conjunto de itens.

- **`--max-results`**—Opcional. O número total de itens para retornar na saída do comando. Se existirem mais registros do que o valor `max-results` especificado, um token de paginação (`next-token`) será incluído na resposta para que os resultados restantes possam ser recuperados.
  - Padrão: 100
  - Mínimo 20, máximo 100

No exemplo a seguir, substitua cada *`user input placeholder`* por suas próprias informações.

Para Linux, macOS ou 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
```

A saída dessa operação é semelhante à seguinte:

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

Como excluir um cluster elástico

Nesta seção, explicamos como excluir um cluster elástico usando o Console de gerenciamento da AWS ou AWS CLI com as instruções a seguir.

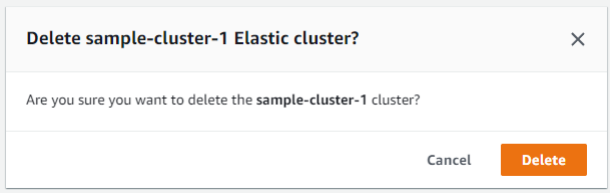
Using the Console de gerenciamento da AWS

Para excluir uma configuração de cluster elástico usando o Console de gerenciamento da AWS:

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon DocumentDB.
2. No painel de navegação, escolha Clusters.

> **Tip**
>
> Se você não visualizar o painel de navegação à esquerda da tela, selecione o ícone do menu no canto superior esquerdo da página.

3. Na tabela da lista de cluster, marque a caixa de seleção à esquerda do nome do cluster que você deseja excluir e escolha Ações. No menu suspenso, escolha Excluir.
4. Na caixa de diálogo Excluir cluster elástico "nome-do-cluster", escolha Excluir.



A exclusão do cluster demora alguns minutos. Para monitorar o status do cluster, consulte Monitoramento do status de um cluster do Amazon DocumentDB.

Using the AWS CLI

Para excluir um cluster elástico usando o AWS CLI, use a `delete-cluster` operação com os seguintes parâmetros:

- **`--cluster-arn`**—Obrigatório. O identificador ARN do cluster que deseja excluir.
- **`--no-skip-final-backup`**—Opcional. Se você desejar um backup final, inclua esse parâmetro com um nome para o backup final. É necessário incluir `--final-backup-identifier` ou `--skip-final-backup`.
- **`--skip-final-backup`**—Opcional. Use esse parâmetro somente se você não quiser obter um snapshot final antes de excluir seu cluster. A configuração padrão é obter uma captura de tela final.

Os exemplos de AWS CLI código a seguir excluem um cluster com um ARN de arn:aws:docdb:us-west- 2:123456789012:sharded-cluster:sample-cluster com um backup final.

No exemplo a seguir, substitua cada um *`user input placeholder`* por suas próprias informações.

Para Linux, macOS ou 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
```

Os exemplos de AWS CLI código a seguir excluem um cluster com um ARN de arn:aws:docdb:us-west- 2:123456789012:sharded-cluster:sample-cluster sem fazer um backup final.

No exemplo a seguir, substitua cada *user input placeholder* por suas próprias informações.

Para Linux, macOS ou 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 ^
```

A saída da operação `delete-cluster` é o cluster que você está excluindo.

A exclusão do cluster demora alguns minutos. Para monitorar o status do cluster, consulte Monitoramento do status de um cluster do Amazon DocumentDB.

Como gerenciar snapshots de cluster elástico

Os snapshots manuais podem ser obtidos após a criação de um cluster elástico. Os backups automatizados são criados no momento em que o snapshot do cluster elástico é criado.

> **Note**
>
> O cluster deve estar no estado `Available` para que um snapshot automático seja obtido.

Esta seção explica como é possível criar, visualizar, restaurar e excluir snapshots de clusters elásticos.

Os tópicos a seguir mostram como realizar várias tarefas ao trabalhar com snapshots de clusters elásticos do Amazon DocumentDB.

Tópicos

- Como criar um snapshot manual de cluster
- Como visualizar um snapshot de cluster elástico
- Como restaurar um cluster usando um snapshot
- Copiar um snapshot de cluster elástico
- Como excluir um snapshot de cluster elástico
- Gerenciar um backup automático de snapshot de cluster elástico

Como criar um snapshot manual de cluster

Nesta seção, explicamos como criar um snapshot manual de cluster elástico usando Console de gerenciamento da AWS ou AWS CLI com as instruções a seguir.

Using the Console de gerenciamento da AWS

Para criar um snapshot manual de cluster elástico usando o Console de gerenciamento da AWS:

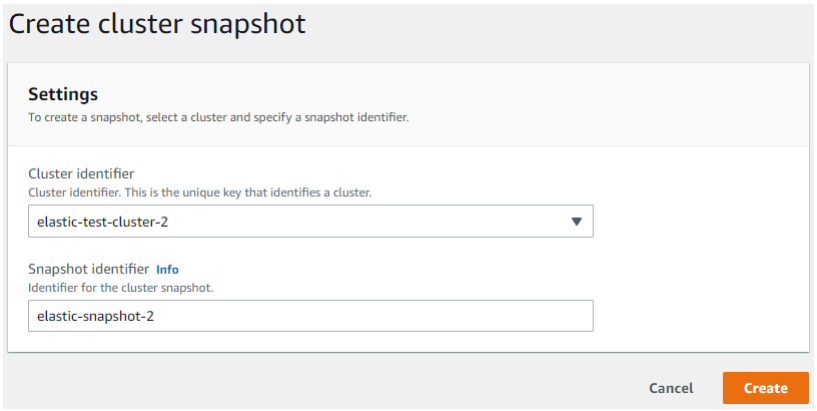
1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon DocumentDB.
2. No painel de navegação, escolha Snapshots.

   > **Tip**
   >
   > Se você não visualizar o painel de navegação à esquerda da tela, selecione o ícone do menu no canto superior esquerdo da página.

3. Na página Snapshots, selecione Criar.
4. Na página Criar snapshot do cluster, no campo Identificador do cluster, escolha seu cluster elástico na lista suspensa.

   No campo Identificador de snapshot, insira um identificador exclusivo para seu cluster elástico.

   Escolha Criar.



> **Note**
>
> Como alternativa, é possível acessar a caixa de diálogo Criar snapshot de cluster acessando a página Clusters, marcando a caixa ao lado do seu cluster e escolhendo Ações e, em seguida, Tirar snapshot.

Agora, seu snapshot de cluster elástico está sendo provisionado. Esse processo pode levar até alguns minutos. É possível visualizar e restaurar a partir do seu snapshot quando os status `Available` for exibido na lista Snapshots.

Using the AWS CLI

Para criar um snapshot manual de cluster elástico usando o AWS CLI, use a `create-cluster-snapshot` operação com os seguintes parâmetros:

- **`--snapshot-name`**—Obrigatório. O nome do snapshot a partir do qual será criado um novo cluster.
- **`--cluster-arn`**—Obrigatório. O identificador ARN do cluster do qual você deseja criar um snapshot.

No exemplo a seguir, substitua cada *`user input placeholder`* por suas próprias informações.

Para Linux, macOS ou 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
```

Como visualizar um snapshot de cluster elástico

Nesta seção, explicamos como visualizar informações de snapshots de clusters elásticos usando Console de gerenciamento da AWS ou AWS CLI com as instruções a seguir.

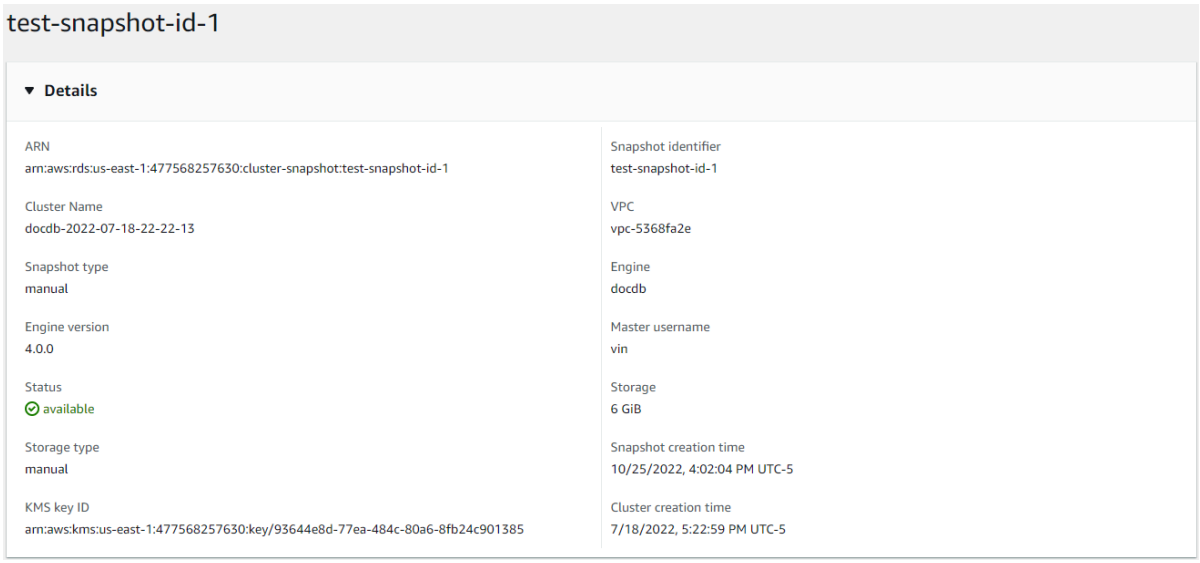
Using the Console de gerenciamento da AWS

Para visualizar informações sobre um snapshot de cluster elástico específico usando o Console de gerenciamento da AWS:

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon DocumentDB.
2. No painel de navegação, escolha Snapshots.

   > **Tip**
   >
   > Se você não visualizar o painel de navegação à esquerda da tela, selecione o ícone do menu no canto superior esquerdo da página.

3. Na página Snapshots, escolha seu snapshot na lista clicando no nome na coluna Identificador do snapshot.
4. Visualize as informações do seu snapshot em Detalhes.



Using the AWS CLI

Para visualizar informações sobre um snapshot de cluster elástico específico usando o AWS CLI, use a `get-cluster-snapshot` operação com os seguintes parâmetros:

- **`--snapshot-arn`**—Obrigatório. O identificador ARN do snapshot do qual você deseja informações.

No exemplo a seguir, substitua cada *`user input placeholder`* por suas próprias informações.

Para Linux, macOS ou Unix:

```
aws docdb-elastic get-cluster-snapshot \
    --snapshot-arn sampleResourceName
```

Para Windows:

```
aws docdb-elastic get-cluster-snapshot ^
    --snapshot-arn sampleResourceName
```

Para visualizar informações sobre um snapshot de cluster elástico específico usando o AWS CLI, use a `get-cluster-snapshot` operação com os seguintes parâmetros:

- **`--snapshot-arn`**—Obrigatório. O identificador ARN do snapshot do qual você deseja informações.

No exemplo a seguir, substitua cada *`user input placeholder`* por suas próprias informações.

Para Linux, macOS ou Unix:

```
aws docdb-elastic get-cluster-snapshot \
    --snapshot-arn sampleResourceName
```

Para Windows:

```
aws docdb-elastic get-cluster-snapshot ^
    --snapshot-arn sampleResourceName
```

Para visualizar informações sobre todos os snapshots de cluster elásticos usando o AWS CLI, use a `list-cluster-snapshots` operação com os seguintes parâmetros:

- **`--snapshot-type`**—Opcional. O tipo de snapshots de cluster a ser retornado. É possível especificar um dos seguintes valores:
  - `automated`- Retorne todos os snapshots de cluster que o Amazon DocumentDB criou automaticamente para sua AWS conta.
  - `manual`- Retorne todos os instantâneos de cluster que você criou manualmente para sua AWS conta.
  - `shared`- Retorne todos os instantâneos manuais do cluster que foram compartilhados em sua AWS conta.
  - `public` - retorna todos os snapshots do cluster que foram marcados como públicos.
- **`--next-token`**—Opcional. Um token de paginação opcional fornecido por uma solicitação anterior. Se esse parâmetro for especificado, a resposta incluirá apenas os registros além do marcador, até o valor especificado por `max-results`.
- **`--max-results`**—Opcional. O número máximo de registros a serem incluídos na resposta. Se existirem mais resultados do que o valor `max-results` especificado, um token de paginação (`next-token`) será incluído na resposta para que os resultados restantes possam ser recuperados.

- Padrão: 100
- Mínimo 20, máximo 100

No exemplo a seguir, substitua cada *user input placeholder* por suas próprias informações.

Para Linux, macOS ou 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
```

Como restaurar um cluster usando um snapshot

Nesta seção, explicamos como restaurar um cluster elástico a partir de um snapshot, usando Console de gerenciamento da AWS ou AWS CLI com as instruções a seguir.

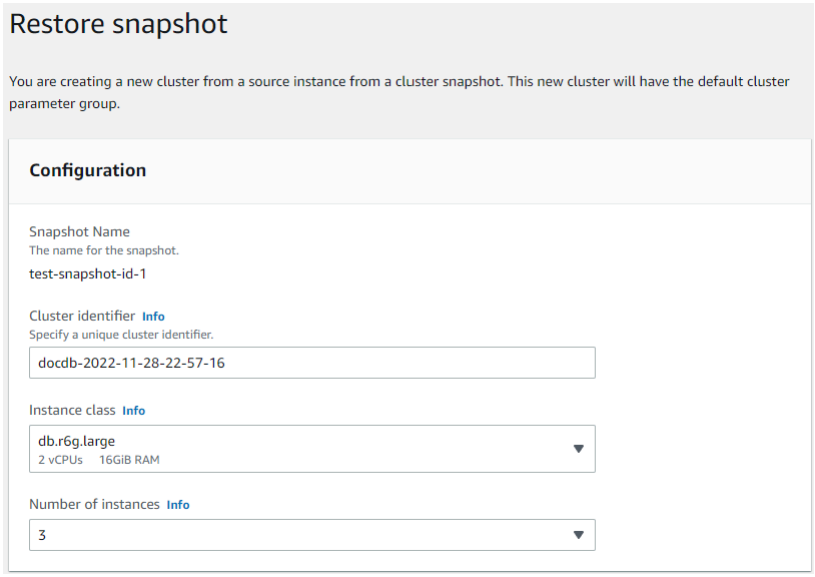
Using the Console de gerenciamento da AWS

Para restaurar um cluster elástico de um snapshot de cluster usando a Console de gerenciamento da AWS:

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon DocumentDB.
2. No painel de navegação, escolha Snapshots.

> **Tip**
>
> Se você não visualizar o painel de navegação à esquerda da tela, selecione o ícone do menu no canto superior esquerdo da página.

3. Escolha o botão à esquerda do snapshot que você quer usar para restaurar um cluster, na coluna Identificador de snapshot.
4. Em Ações, escolha Restaurar.



5. Na página Restaurar snapshot, insira um nome para o novo cluster no campo Identificador do cluster.

> **Note**
>
> Para qualquer restauração manual de snapshots, você deve criar um novo cluster.

6. No campo Nuvem Privada Virtual (VPC), escolha uma VPC da lista suspensa.
7. Em Sub-redes e Grupos de segurança de VPC, é possível usar os padrões ou selecionar três sub-redes de sua escolha e até três grupos de segurança de VPC (no mínimo um).
8. Se você estiver satisfeito com a configuração de cluster, escolha Restore cluster (Restaurar cluster) e aguarde enquanto o cluster é restaurado.

Using the AWS CLI

Para restaurar um cluster elástico a partir de um snapshot usando o AWS CLI, use a `restore-cluster-from-snapshot` operação com os seguintes parâmetros:

- **`--cluster-name`**—Obrigatório. O nome atual do cluster elástico, conforme inserido durante a criação ou a última modificação.

- **`--snapshot-arn`**—Obrigatório. O identificador ARN do snapshot que está sendo usado para restaurar o cluster.
- **`--vpc-security-group-ids`**—Opcional. Uma lista de grupos de segurança do Amazon EC2 e da Nuvem privada virtual (VPC) para associar a esse cluster.
- **`--kms-key-id`**—Opcional. O identificador da chave do KMS para um cluster criptografado.

  O identificador da chave KMS é o nome de recurso da Amazon (ARN) para a chave de AWS KMS criptografia. Se você estiver criando um cluster com a mesma conta da Amazon Web Services que tem a chave de criptografia KMS usada para criptografar o novo cluster, use o apelido da chave do KMS em vez de o ARN da chave de criptografia do KMS.

  Se uma chave de criptografia não for especificada em KmsKeyId e se o `StorageEncrypted` parâmetro for verdadeiro, o Amazon DocumentDB usará sua chave de criptografia padrão.
- **`--subnet-ids`**—Opcional. IDs de sub-rede da rede.

No exemplo a seguir, substitua cada um *`user input placeholder`* por suas próprias informações.

Para Linux, macOS ou 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
```

Copiar um snapshot de cluster elástico

No Amazon DocumentDB, você pode copiar snapshots manuais e automáticos de clusters elásticos na mesma região e na mesma conta. Nesta seção, explicamos como copiar um snapshot de cluster elástico usando o Console de gerenciamento da AWS ou AWS CLI.

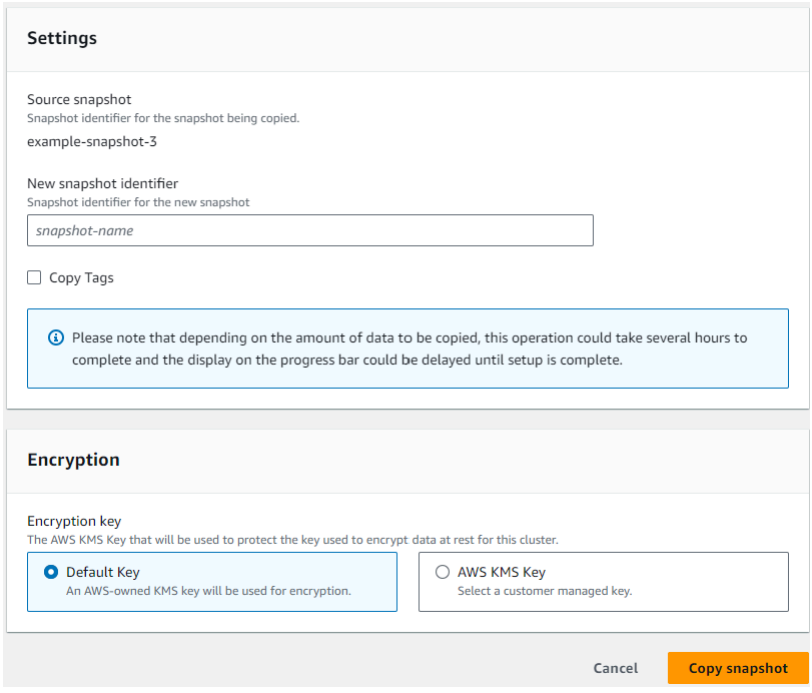
Using the Console de gerenciamento da AWS

Para copiar um snapshot de cluster elástico usando o Console de gerenciamento da AWS:

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon DocumentDB.
2. No painel de navegação, escolha Snapshots.

> **Tip**
>
> Se você não visualizar o painel de navegação à esquerda da tela, selecione o ícone do menu no canto superior esquerdo da página.

3. Escolha o botão à esquerda do snapshot que você quer copiar na coluna Identificador de snapshot.
4. Escolha Ações e Copiar.



5. Em Identificador do novo snapshot, insira o nome do novo snapshot.

6. Em Copiar tags, marque a caixa se quiser copiar todas as tags do snapshot do cluster elástico de origem no snapshot do cluster elástico de destino.
7. Para Criptografia, escolha uma chave AWS KMS padrão ou uma chave KMS de sua escolha. A segunda opção permite selecionar uma chave KMS existente que você já criou ou permite criar uma nova.
8. Escolha Copiar snapshot ao concluir.

Using the AWS CLI

Para copiar um snapshot de cluster elástico usando o AWS CLI, use a `copy-cluster-snapshot` operação com os seguintes parâmetros:

- **`--source-db-cluster-snapshot-identifier`**—Obrigatório. O identificador do snapshot do cluster elástico existente a ser copiado. O snapshot do cluster elástico deve existir e estar no estado disponível. Esse parâmetro não diferencia maiúsculas de minúsculas.
- **`--target-db-cluster-snapshot-identifier`**—Obrigatório. O identificador do novo snapshot de cluster elástico a ser criado a partir do snapshot de cluster existente. Esse parâmetro não diferencia maiúsculas de minúsculas.

  Restrições de nome do snapshot de destino:
  - Não pode ser o nome de um snapshot existente.
  - O comprimento é de [1 a 63] letras, números ou hífens.
  - O primeiro caractere deve ser uma letra.
  - Não pode terminar com um hífen ou conter dois hifens consecutivos.

No exemplo a seguir, substitua cada *`user input placeholder`* por suas próprias informações.

Para Linux, macOS ou 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
```

Como excluir um snapshot de cluster elástico

Nesta seção, explicamos como excluir um snapshot de cluster elástico usando o Console de gerenciamento da AWS ou AWS CLI.

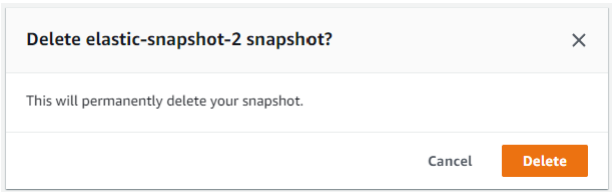
Using the Console de gerenciamento da AWS

Para restaurar um cluster elástico de um snapshot de cluster usando a Console de gerenciamento da AWS:

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon DocumentDB.
2. No painel de navegação, escolha Snapshots.

> **Tip**
>
> Se você não visualizar o painel de navegação à esquerda da tela, selecione o ícone do menu no canto superior esquerdo da página.

3. Escolha o botão à esquerda do snapshot que você quer usar para restaurar um cluster, na coluna Identificador de snapshot.
4. Escolha Ações e Excluir.



5. Na caixa de diálogo Excluir snapshot “nome-do-snapshot”, escolha Excluir.

Using the AWS CLI

Para excluir um snapshot de cluster elástico usando o AWS CLI, use a `delete-cluster-snapshot` operação com os seguintes parâmetros:

- **`--snapshot-arn`**—Obrigatório. O identificador ARN do snapshot que está sendo usado para restaurar o cluster.

No exemplo a seguir, substitua cada um *`user input placeholder`* por suas próprias informações.

Para Linux, macOS ou Unix:

```
aws docdb-elastic delete-cluster-snapshot \
     --snapshot-arn sampleResourceName
```

Para Windows:

```
aws docdb-elastic delete-cluster-snapshot ^
     --snapshot-arn sampleResourceName
```

Gerenciar um backup automático de snapshot de cluster elástico

O Amazon DocumentDB tira snapshots diários de seus clusters elásticos. É possível especificar a janela de backup preferencial e o período de retenção de backup em uma configuração de snapshot de cluster elástico nova ou existente. Nesta seção, explicamos como definir parâmetros de backup automático em um snapshot de cluster elástico usando o Console de gerenciamento da AWS ou AWS CLI.

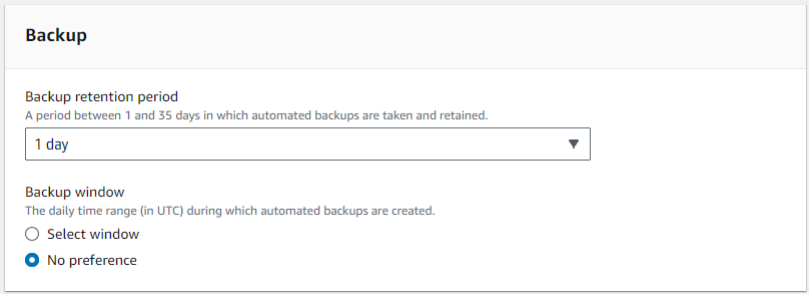
Using the Console de gerenciamento da AWS

Para definir um backup automático para um novo snapshot de cluster elástico usando o Console de gerenciamento da AWS:

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon DocumentDB.
2. No painel de navegação, escolha Clusters.

> **Tip**
>
> Se você não visualizar o painel de navegação à esquerda da tela, selecione o ícone do menu no canto superior esquerdo da página.

3. Escolha o botão à esquerda do cluster no qual você quer alterar as configurações de backup, na coluna Identificador de cluster.
4. Selecione Ações e Modificar.
5. Na seção Backup, edite os campos de acordo com os requisitos do seu backup.



   a. Período de retenção de backup— na lista, selecione o número de dias para manter os backups automáticos desse cluster antes de excluí-los.
   b. Janela de backup— defina o tempo e duração diários durante os quais o Amazon DocumentDB fará backups desse cluster.

      i. Escolha Selecionar janela se quiser configurar a hora e a duração em que os backups são criados.

         Hora de início: na primeira lista, escolha a hora de início (UTC) para seus backups automáticos. Na segunda lista, escolha o minuto da hora em que você deseja que os backups automáticos sejam iniciados.

         Duração— na lista, selecione a quantidade de horas a serem alocadas na criação de backups automáticos.

      ii. Escolha Sem preferência se quiser que o Amazon DocumentDB escolha a hora e a duração em que os backups são criados.
6. Escolha Modificar cluster quando estiver concluído.

Using the AWS CLI

Para definir um backup automático para um novo snapshot de cluster elástico usando o AWS CLI, use a `create-cluster-snapshot` operação com os seguintes parâmetros:

- **`--preferred-backup-window`**—Opcional. O intervalo de tempo diário preferencial, durante o qual os backups automatizados são criados. O padrão é uma janela de 30 minutos selecionada aleatoriamente a partir de um bloco de 8 horas para cada uma. Região da AWS

Restrições:

- Deve estar no formato `hh24:mi-hh24:mi`.
- Deve estar expresso no Tempo Universal Coordenado (UTC).
- Não pode entrar em conflito com a janela de manutenção preferencial.
- Deve ser, pelo menos, 30 minutos.

- **`--backup-retention-period`**— opcional. O número de dias durante os quais os backups automatizados são retidos. O valor padrão é 1.

  Restrições:

  - É preciso especificar o valor mínimo de 1.
  - Varia de 1 a 35.

> **Note**
>
> Os backups automatizados são feitos somente quando o cluster está no estado “ativo”.

> **Note**
>
> Você também pode modificar os parâmetros `preferred-backup-window` e `backup-retention-period` de um cluster elástico existente usando o comando `aws docdb-elastic update-cluster`.

No exemplo a seguir, substitua cada *`user input placeholder`* por suas próprias informações.

O `create-cluster` exemplo a seguir cria o cluster elástico Amazon DocumentDB *`sample-cluster`* com o período de retenção para backups automáticos de *`7`* dias e uma janela de backup preferencial de*`18:00-18:30 UTC`*.

Para Linux, macOS ou 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
```

Interromper e iniciar um cluster elástico do Amazon DocumentDB

Interromper e iniciar clusters elásticos do Amazon DocumentDB pode ajudar a gerenciar os custos dos ambientes de teste e desenvolvimento. Em vez de criar e excluir clusters elásticos toda vez que você usar o Amazon DocumentDB, você pode interromper temporariamente seu cluster quando não for necessário. Depois, é possível iniciá-lo novamente, ao retomar os testes.

Tópicos

- Visão geral de como interromper e iniciar um cluster elástico
- Operações que você pode realizar em um cluster elástico interrompido

Visão geral de como interromper e iniciar um cluster elástico

Durante os períodos nos quais não precisar de um cluster elástico do Amazon DocumentDB, você pode interromper o cluster. Depois, você pode iniciar o cluster novamente a qualquer momento, sempre que precisar usá-lo. Iniciar e interromper simplifica os processos de configuração e destruição dos clusters elásticos usados em desenvolvimento, teste ou atividades afins que não exijam disponibilidade contínua. Você pode parar e iniciar um cluster elástico usando o Console de gerenciamento da AWS ou o AWS CLI com uma única ação.

Enquanto o cluster elástico estiver interrompido, o volume de armazenamento do cluster permanecerá inalterado. Serão cobrados somente o armazenamento, as captura de telas manuais e o armazenamento do backup automatizado dentro da janela de retenção especificada. O Amazon DocumentDB iniciará automaticamente seu cluster elástico depois de sete dias para garantir que ele não perca nenhuma atualização de manutenção necessária. Quando o cluster reiniciar após sete dias, você começará a ser cobrado pelo uso do cluster elástico novamente. Enquanto o cluster está interrompido, não é possível consultar o volume de armazenamento porque a consulta requer que o cluster esteja no estado disponível.

Quando um cluster elástico do Amazon DocumentDB é interrompido, não pode ser modificado de maneira alguma. Isso inclui sua exclusão.

Using the Console de gerenciamento da AWS

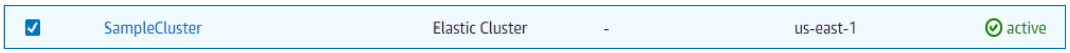
O procedimento a seguir mostra como interromper um cluster elástico com no estado disponível ou iniciar um cluster elástico interrompido.

Para interromper ou iniciar um cluster elástico do Amazon DocumentDB

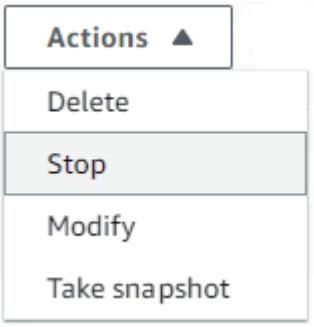
1. Faça login no e abra Console de gerenciamento da AWS o console do Amazon DocumentDB em https://console.aws.amazon.com/docdb.
2. No painel de navegação, escolha Clusters.

   > **Tip**
   >
   > Caso não visualize o painel de navegação à esquerda da tela, selecione o ícone do menu (≡) no canto superior esquerdo da página.

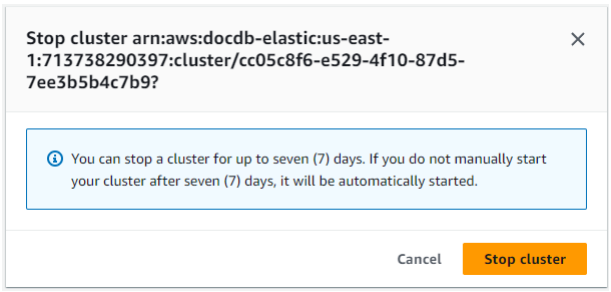
3. Na lista de clusters, escolha o botão à esquerda do nome do cluster que você deseja interromper ou iniciar.

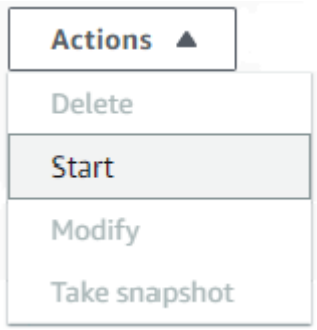
4. Escolha Ações e selecione a ação que deseja executar no cluster.
   - Se você quiser interromper o cluster e ele estiver disponível:
     a. Escolha Parar.



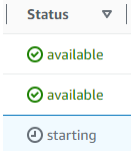
b. Na caixa de diálogo de confirmação, confirme se deseja interromper o cluster elástico escolhendo Interromper cluster ou, para manter o cluster em execução, escolha Cancelar.



- Se quiser iniciar o cluster e o cluster estiver interrompido, escolha Iniciar.



5. Monitorar o status do cluster elástico. Se você iniciou o cluster, poderá retomar seu uso quando ele estiver disponível. Para obter mais informações, consulte Determinar o status de um cluster.



Using the AWS CLI

Os exemplos de código a seguir mostram como interromper um cluster elástico no estado ativo ou disponível ou iniciar um cluster elástico interrompido.

Para interromper um cluster elástico usando o AWS CLI, use a `stop-cluster` operação. Para iniciar um cluster interrompido, use a operação `start-cluster`. Ambas as operações usam o parâmetro `--cluster-arn`.

Parâmetro

- **`--cluster-arn`**—Obrigatório. O identificador ARN do cluster elástico que deseja interromper ou iniciar.

Example— Para interromper um cluster elástico usando o AWS CLI

No exemplo a seguir, substitua cada *`user input placeholder`* por suas próprias informações.

O código a seguir interrompe o cluster elástico com o ARN `arn:aws:docdb-elastic:us-east-1:477568257630:cluster/b9f1d489-6c3e-4764-bb42-da62ceb7bda2`.

> **Note**
>
> O cluster elástico deve estar no estado ativo ou disponível.

Para Linux, macOS ou 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 um cluster elástico usando o AWS CLI

No exemplo a seguir, substitua cada *user input placeholder* por suas próprias informações.

O código a seguir inicia o cluster elástico com o ARN `arn:aws:docdb-elastic:us-east-1:477568257630:cluster/b9f1d489-6c3e-4764-bb42-da62ceb7bda2`.

> **Note**
>
> O cluster elástico deve estar interrompido no momento.

Para Linux, macOS ou 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
```

Operações que você pode realizar em um cluster elástico interrompido

Não é possível modificar a configuração de um cluster elástico do Amazon DocumentDB enquanto o cluster estiver interrompido. Será necessário iniciar o cluster antes de realizar uma dessas ações administrativas.

O Amazon DocumentDB aplicará as manutenções programadas em seu cluster elástico interrompido somente depois que for reiniciado. Após sete dias, o Amazon DocumentDB inicia automaticamente um cluster elástico interrompido, para que não fique muito atrasado em seu status de manutenção. Quando o cluster elástico for reiniciado, você começará a ser cobrado pelos fragmentos no cluster novamente.

Enquanto um cluster elástico estiver interrompido, o Amazon DocumentDB não executa quaisquer backups automatizados, nem estende o período de retenção de backup.

Manutenção dos clusters elásticos do Amazon DocumentDB

Tópicos

- Visualização de ações pendentes de manutenção de clusters elásticos
- Atualizações do mecanismo de cluster elástico
- Atualizações do sistema operacional do cluster elástico

Periodicamente, o Amazon DocumentDB realiza a manutenção em seus recursos de clusters elásticos. Geralmente, a manutenção envolve atualizações do mecanismo de banco de dados (manutenção do cluster elástico) ou do sistema operacional (SO) subjacente ao cluster elástico (manutenções de SO). As atualizações do mecanismo de banco de dados são patches obrigatórios e incluem correções de segurança, correções de bugs e aprimoramentos no mecanismo de banco de dados. Embora a maioria dos patches do sistema operacional sejam opcionais, se você não os aplicar por algum tempo, o patch poderá ser necessário e aplicado automaticamente para manter sua postura de segurança. Aplique atualizações do sistema operacional aos seus clusters elásticos do Amazon DocumentDB assim que estiverem disponíveis.

Os patches do mecanismo de banco de dados exigem que você coloque os clusters elásticos do Amazon DocumentDB offline por um curto período. Uma vez disponíveis, esses patches são automaticamente programados para serem aplicados durante uma próxima janela de manutenção programada do seu cluster elástico do Amazon DocumentDB.

Os clusters elásticos têm suas próprias janelas de manutenção respectivas. As modificações no cluster elástico e na instância que você optou por não aplicar imediatamente são aplicadas durante a janela de manutenção. Por padrão, quando você cria um cluster elástico, o Amazon DocumentDB atribui uma janela de manutenção ao seu cluster elástico. É possível escolher a janela de manutenção ao criar um cluster elástico. Você também pode modificar as janelas de manutenção a qualquer momento para atender às suas programações ou práticas comerciais. Em geral, é recomendável escolher janelas de manutenção que minimizem o impacto da manutenção em sua aplicação (por exemplo, em noites ou fins de semana).

Visualização de ações pendentes de manutenção de clusters elásticos

É possível ver se uma atualização de manutenção está disponível para o seu cluster elástico usando a AWS CLI.

Se uma atualização estiver disponível, será possível executar uma das seguintes ações:

- Adiar uma ação de manutenção atualmente programada para a próxima janela de manutenção (somente para patches do sistema operacional).
- Aplicar as ações de manutenção imediatamente.
- Agendar as ações de manutenção para serem iniciadas durante a próxima janela de manutenção.
- Agende as ações de manutenção para serem iniciadas durante sua próxima janela de manutenção.

A janela de manutenção determina quando as operações pendentes começam, mas não limita o tempo total de execução dessas operações.

Use a AWS CLI operação a seguir para determinar quais ações de manutenção estão pendentes. Liste todas as ações de manutenção pendentes:

```
aws docdb-elastic list-pending-maintenance-actions
```

A saída dessa operação é semelhante ao seguinte (formato JSON):

```
{
'ResourcePendingMaintenanceActions': [
    {
        'ResourceArn': 'string-arn',
        'PendingMaintenanceActionDetails': [
            {
                'Action': 'ENGINE_UPDATE',
                'AutoAppliedAfterDate': 'string',
                'ForcedApplyDate': 'string',
                'OptInStatus': 'string',
                'CurrentApplyDate': 'string',
                'Description': 'string'
            },
        ]
    },
],
'NextToken': 'string'
}
```

Obtenha uma ação de manutenção pendente (se houver) em um determinado `resourceArn`:

```
aws docdb-elastic get-pending-maintenance-action --resource-arn string-arn
```

A saída dessa operação é semelhante ao seguinte (formato JSON).

```
{
    'ResourcePendingMaintenanceAction': {
        'ResourceArn': 'string-arn',
        'PendingMaintenanceActionDetails': [
            {
                'Action': 'ENGINE_UPDATE',
                'AutoAppliedAfterDate': 'string',
                'ForcedApplyDate': 'string',
                'OptInStatus': 'string',
                'CurrentApplyDate': 'string',
                'Description': 'string'
            }
        ]
    }
}
```

Parâmetros:

- `ResourceArn`: o nome do recurso da Amazon (ARN) do Amazon DocumentDB do recurso ao qual a ação de manutenção pendente se aplica.
- `Action`: o tipo de ação de manutenção pendente sendo aplicada ao recurso.

  Valores válidos:

  - `ENGINE_UPDATE`
  - `ENGINE_UPGRADE`
  - `SECURITY_UPDATE`
  - `OS_UPDATE`
  - `MASTER_USER_PASSWORD_UPDATE`
- `AutoAppliedAfterDate`: primeira janela de manutenção após essa data. `NEXT_MAINTENANCE` `OPT_IN` é ignorado nesse caso.
- `ForcedApplyDate`: aplicada independentemente da janela de manutenção. `IMMEDIATE` `OPT_IN` é ignorado nesse caso.
- `OptInStatus` - um valor que especifica o tipo de solicitação de inclusão ou desfaz uma solicitação de inclusão. Uma solicitação de inclusão do tipo `IMMEDIATE` não pode ser desfeita.

  Valores válidos:

- `IMMEDIATE` – aplique a ação de manutenção imediatamente.
- `NEXT_MAINTENANCE` - Aplique a ação de manutenção durante a próxima janela de manutenção do recurso.
- `APPLY_ON`: aplica a ação de manutenção na data de aplicação especificada, independentemente da próxima janela de manutenção do recurso.
- `UNDO_OPT_IN`: cancela quaisquer solicitações de inclusão de `NEXT_MAINTENANCE` ou `APPLY_ON` existentes.

- `CurrentApplyDate`: exibido se o tipo de inclusão for `APPLY_ON`.
- `Description`: uma descrição da opção para a ação de manutenção.

Atualizações do mecanismo de cluster elástico

Com o Amazon DocumentDB, você pode escolher quando aplicar operações de manutenção. É possível decidir quando o Amazon DocumentDB deve aplicar as atualizações usando a AWS CLI.

Aplicação de ações de manutenção pendentes:

```
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]
```

Parâmetros:

- **`--resource-arn`**: o nome do recurso da Amazon (ARN) do Amazon DocumentDB do recurso ao qual a ação de manutenção pendente se aplica.
- **`--apply-action`** - a ação de manutenção pendente a ser aplicada a esse recurso.

  Valores válidos:

  - `ENGINE_UPDATE`
  - `ENGINE_UPGRADE`
  - `SECURITY_UPDATE`
  - `OS_UPDATE`
  - `MASTER_USER_PASSWORD_UPDATE`

- **`--opt-in-type`** - um valor que especifica o tipo de solicitação de inclusão ou desfaz uma solicitação de inclusão. Uma solicitação de inclusão do tipo `IMMEDIATE` não pode ser desfeita.

  Valores válidos:

  - `IMMEDIATE` – aplique a ação de manutenção imediatamente.
  - `NEXT_MAINTENANCE` - Aplique a ação de manutenção durante a próxima janela de manutenção do recurso.
  - `APPLY_ON`: aplica a ação de manutenção na data de aplicação especificada, independentemente da próxima janela de manutenção do recurso.
  - `UNDO_OPT_IN`: cancela quaisquer solicitações de inclusão de `NEXT_MAINTENANCE` ou `APPLY_ON` existentes.
- **`[--apply-on]`**: obrigatório se o tipo de inclusão for `APPLY_ON`. Formato: `yyyy/MM/dd HH:mm-yyyy/MM/dd HH:mm` (Essa opção usa o horário UTC. A hora de início pode ser a qualquer momento no futuro, de no mínimo 30 minutos e no máximo 14 dias, ou a force/apply data da ação pendente, o que ocorrer primeiro. A janela de horário do início ao fim pode ser de no mínimo 30 minutos a no máximo 8 horas.)

A saída dessa operação é semelhante ao seguinte (formato JSON):

```
{
 'ResourcePendingMaintenanceAction': {
        'ResourceArn': 'string-arn',
        'PendingMaintenanceActionDetails': [
            {
                'Action': 'SECURITY_UPDATE',
                'AutoAppliedAfterDate': 'string',
                'ForcedApplyDate': 'string',
                'OptInStatus': 'IMMEDIATE',
                'CurrentApplyDate': 'string',
                'Description': 'string'
            },
        ]
 }
}
```

Parâmetros:

- `ResourceArn`: o nome do recurso da Amazon (ARN) do Amazon DocumentDB do recurso ao qual a ação de manutenção pendente se aplica.
- `Action`: o tipo de ação de manutenção pendente sendo aplicada ao recurso.

  Valores válidos:

  - `ENGINE_UPDATE`
  - `ENGINE_UPGRADE`
  - `SECURITY_UPDATE`
  - `OS_UPDATE`
  - `MASTER_USER_PASSWORD_UPDATE`
- `AutoAppliedAfterDate`: primeira janela de manutenção após essa data. `NEXT_MAINTENANCE` `OPT_IN` é ignorado nesse caso.
- `ForcedApplyDate`: aplicada independentemente da janela de manutenção. `IMMEDIATE` `OPT_IN` é ignorado nesse caso.
- `OptInStatus` - um valor que especifica o tipo de solicitação de inclusão ou desfaz uma solicitação de inclusão. Uma solicitação de inclusão do tipo `IMMEDIATE` não pode ser desfeita.

  Valores válidos:

  - `IMMEDIATE` – aplique a ação de manutenção imediatamente.
  - `NEXT_MAINTENANCE` - Aplique a ação de manutenção durante a próxima janela de manutenção do recurso.
  - `APPLY_ON`: aplica a ação de manutenção na data de aplicação especificada, independentemente da próxima janela de manutenção do recurso.
  - `UNDO_OPT_IN`: cancela quaisquer solicitações de inclusão de `NEXT_MAINTENANCE` ou `APPLY_ON` existentes.
- `CurrentApplyDate`: exibido se o tipo de inclusão for `APPLY_ON`.
- `Description`: uma descrição da opção para a ação de manutenção.

Aplicar datas

Cada ação de manutenção tem uma respectiva data de aplicação que você pode encontrar ao descrever as ações de manutenção pendentes. Quando você lê o resultado das ações de manutenção pendentes do AWS CLI, três datas são listadas:

- `CurrentApplyDate` - Data em que a ação de manutenção será aplicada imediatamente ou durante a próxima janela de manutenção. Se a manutenção for opcional, esse valor poderá ser nulo.
- `ForcedApplyDate` - Data em que a manutenção será aplicada automaticamente, independente de sua janela de manutenção.
- `AutoAppliedAfterDate` - Data depois da qual a manutenção será aplicada durante a janela de manutenção do cluster.

User-created ações de manutenção

Como usuário do cluster elástico do Amazon DocumentDB, é possível iniciar atualizações para as configurações dos seus clusters.

Atualização da senha primária do cluster

```
aws docdb-elastic update-cluster
--cluster-arn string-arn
[--admin-user-password string]
[--auth-type string-enum]
[--apply-method string-enum]
[--apply-on string-date-range]
#... other parameters of the API that follow here are not relevant for this
 configuration
```

Parâmetros:

- **`--cluster-arn`**: o nome do recurso da Amazon (ARN) do Amazon DocumentDB do recurso ao qual a ação de manutenção pendente se aplicará.
- **`[--admin-user-password]`**: a senha associada ao usuário administrativo.
- **`[--auth-type]`**: o tipo de autenticação usado para determinar onde buscar a senha usada para acessar o cluster elástico. Os tipos válidos são `PLAIN_TEXT` e `SECRET_ARN`.
- **`[--apply-method]`**: um valor que especifica o tipo de método que está sendo aplicado. Os valores permitidos são `IMMEDIATE` e `APPLY_ON`. O padrão é `IMMEDIATE`.
- **`[--apply-on]`**: obrigatório se `apply-method` for `APPLY_ON`. Formato: `yyyy/MM/dd HH:mm-yyyy/MM/dd HH:mm` (Esta opção utiliza o horário UTC. O horário de início pode ser em qualquer momento no futuro, com um intervalo mínimo de 30 minutos e máximo de 14 dias. O intervalo de tempo entre o início e o término pode ter um mínimo de 30 minutos e um máximo de 8 horas.)

A saída dessa operação é semelhante ao seguinte (formato JSON):

```
{
 'ResourcePendingMaintenanceAction': {
        'ResourceArn': 'string-arn',
        'PendingMaintenanceActionDetails': [
            {
                'Action': 'MASTER_USER_PASSWORD_UPDATE',
                'OptInStatus': 'APPLY_ON',
                'CurrentApplyDate': 'string',
                'Description': 'string'
            },
        ]
 }
}
```

Gerenciar suas janelas de manutenção do Amazon DocumentDB

A janela de manutenção deve ser definida no horário de menor utilização e, portanto, talvez precise ser alterada de vez em quando. Seu cluster elástico estará indisponível durante esse tempo somente se as alterações do sistema (como uma alteração de operação de armazenamento em escala) estiverem sendo aplicadas e exigirem uma interrupção. Ele estará indisponível apenas pelo intervalo mínimo de tempo necessário para fazer as alterações necessárias.

O padrão é uma janela de 30 minutos selecionada aleatoriamente a partir de um bloco de tempo de 8 horas para cada região da Amazon Web Services, ocorrendo em um dia da semana aleatório.

Para alterar a janela de manutenção, consulte Como modificar configurações de clusters elásticos.

Atualizações do sistema operacional do cluster elástico

Os clusters elásticos do Amazon DocumentDB ocasionalmente exigem atualizações do sistema operacional. O Amazon DocumentDB faz upgrade do sistema operacional para uma versão mais recente para melhorar a performance do banco de dados e o procedimento de segurança geral dos clientes. As atualizações de sistema operacional não alteram a versão do mecanismo do cluster de um cluster elástico do Amazon DocumentDB.

A maioria das atualizações do sistema operacional para clusters elásticos do Amazon DocumentDB são opcionais e não têm uma data definida para serem aplicadas. Contudo, se você não aplicar essas atualizações por um tempo, elas poderão eventualmente se tornar necessárias e aplicadas automaticamente durante a janela de manutenção dos seus clusters. Isso ajuda a manter a postura

de segurança de seu banco de dados. Para evitar um tempo de inatividade inesperado, aplique atualizações do sistema operacional ao seu cluster elástico do Amazon DocumentDB assim que elas estiverem disponíveis e defina a janela de manutenção do cluster em um horário adequado às suas necessidades comerciais.

Criptografia de dados em repouso para clusters elásticos do Amazon DocumentDB

Os tópicos a seguir ajudam você a aprender, criar e monitorar chaves de AWS Key Management Service criptografia para clusters elásticos do Amazon DocumentDB:

Tópicos

- Como os clusters elásticos do Amazon DocumentDB usam concessões em AWS KMS
- Criar uma chave gerenciada pelo cliente
- Monitorar suas chaves de criptografia para clusters elásticos do Amazon DocumentDB
- Saiba mais

Os clusters elásticos do Amazon DocumentDB se integram automaticamente com AWS Key Management Service (AWS KMS) para gerenciamento de chaves e usam um método conhecido como criptografia de envelope para proteger seus dados. Para ter mais informações sobre a criptografia de envelope, consulte Criptografia de envelope no Guia do desenvolvedor do AWS Key Management Service .

An AWS KMS key é uma representação lógica de uma chave. A chave do KMS inclui metadados, como o ID da chave, a data de criação, a descrição e o estado da chave. A chave do KMS também contém o material de chave usado para criptografar e descriptografar dados. Para obter mais informações sobre as chaves do KMS, consulte AWS KMS keys no Guia do desenvolvedor do AWS Key Management Service .

Os clusters elásticos do Amazon DocumentDB oferecem suporte à criptografia com dois tipos de chaves:

- AWS chaves próprias — os clusters elásticos do Amazon DocumentDB usam essas chaves por padrão para criptografar automaticamente dados de identificação pessoal. Você não pode visualizar, gerenciar ou usar chaves AWS de propriedade, nem auditar seu uso. No entanto, não é necessário tomar nenhuma medida nem alterar qualquer programa para proteger as chaves que

criptografam seus dados. Para obter mais informações, consulte chaves de propriedade da AWS no Guia do desenvolvedor do AWS Key Management Service .

- Customer-managed chaves — Simétricas AWS KMS keys que você cria, possui e gerencia. Como você tem controle total dessa camada de criptografia, você pode realizar tarefas como:
  - Estabelecer e manter as políticas de chave
  - Estabelecer e manter subsídios e IAM policies
  - Habilitar e desabilitar políticas de chaves
  - Alternar os materiais de criptografia de chave
  - Adicionar etiquetas
  - Criar réplicas de chaves
  - Chaves de agendamento para exclusão

  Para obter mais informações, consulte Customer managed keys no Guia do desenvolvedor do AWS Key Management Service .

> **Important**
>
> Você deve usar uma chave do KMS de criptografia simétrica para criptografar seu cluster, pois o Amazon DocumentDB só oferece suporte a chaves KMS de criptografia simétrica. Não use uma chave KMS assimétrica para tentar criptografar os dados nos clusters elásticos do Amazon DocumentDB. Para obter mais informações, consulte Chaves assimétricas no AWS KMS no Guia do desenvolvedor do AWS Key Management Service .
> Se o Amazon DocumentDB não puder mais obter acesso à chave de criptografia de um cluster por exemplo, quando o acesso a uma chave for revogado o cluster criptografado entrará em um estado de terminal. Nesse caso, só é possível restaurar o cluster a partir de um backup. Para o Amazon DocumentDB, os backups estão sempre habilitados para 1 dia. Além disso, se você desativar a chave para um cluster criptografado do Amazon DocumentDB, acabará perdendo o acesso de leitura e gravação a esse cluster. Quando o Amazon DocumentDB encontra um cluster que é criptografado por uma chave à qual ele não tem acesso, ele coloca o cluster em um estado terminal. Nesse estado, o cluster deixa de estar disponível e o estado atual do banco de dados não pode ser recuperado. Para restaurar o cluster, você deve reativar o acesso à chave de criptografia para o Amazon DocumentDB e, depois, restaurar o cluster a partir de um backup.

> **Important**
>
> Não é possível alterar a chave KMS para um cluster criptografado depois de já tê-lo criado. Certifique-se de determinar seus requisitos de chave de criptografia antes de criar seu cluster elástico criptografado.

Como os clusters elásticos do Amazon DocumentDB usam concessões em AWS KMS

Os clusters elásticos do Amazon DocumentDB exigem uma concessão para usar sua chave gerenciada pelo cliente.

Quando você cria um cluster criptografado com uma chave gerenciada pelo cliente, os clusters elásticos do Amazon DocumentDB criam uma concessão em seu nome enviando uma `CreateGrant` solicitação para o. AWS KMS As concessões AWS KMS são usadas para dar aos clusters elásticos do Amazon DocumentDB acesso a uma chave KMS em uma conta de cliente.

Os clusters elásticos do Amazon DocumentDB exigem a concessão para usar sua chave gerenciada pelo cliente para as seguintes operações internas:

- Envie `DescribeKey` solicitações AWS KMS para verificar se o ID simétrico da chave gerenciada pelo cliente, inserido ao criar um cluster elástico, é válido.
- Envie `GenerateDataKey` solicitações AWS KMS para gerar chaves de dados criptografadas pela chave gerenciada pelo cliente.
- Envie `Decrypt` solicitações AWS KMS para descriptografar as chaves de dados criptografadas para que elas possam ser usadas para criptografar seus dados.
- É possível revogar o acesso à concessão, ou remover o acesso do serviço à chave gerenciada pelo cliente a qualquer momento. Se você fizer isso, os clusters elásticos do Amazon DocumentDB não poderão acessar nenhum dos dados criptografados com a chave gerenciada pelo cliente, o que afetará as operações que dependerem desses dados.

Criar uma chave gerenciada pelo cliente

Você pode criar uma chave simétrica gerenciada pelo cliente usando o Console de gerenciamento da AWS ou a AWS KMS API.

Criação das chaves simétricas gerenciadas pelo cliente

Siga as etapas para criar uma chave simétrica gerenciada pelo cliente no Guia do AWS Key Management Service desenvolvedor.

Política de chave

As políticas de chaves controlam o acesso à chave gerenciada pelo cliente. Cada chave gerenciada pelo cliente deve ter exatamente uma política de chaves, que contém declarações que determinam quem pode usar a chave e como pode usá-la. Ao criar a chave gerenciada pelo cliente, é possível especificar uma política de chaves. Para obter mais informações, consulte as informações de acesso à chave KMS localizadas na Visão geral do AWS Key Management Service no Guia do desenvolvedor do AWS Key Management Service .

Para usar sua chave gerenciada pelo cliente com recursos de cluster elástico do Amazon DocumentDB, as seguintes operações de API devem ser permitidas na política de chaves:

- `kms:CreateGrant`: Adiciona uma concessão a uma chave gerenciada pelo cliente. Concede acesso de controle a uma chave KMS especificada, que permite o acesso às operações de concessão que os clusters elásticos do Amazon DocumentDB exigem. Para obter mais informações sobre o uso de concessões, consulte Concessões no AWS KMS no Guia do desenvolvedor do AWS Key Management Service .
- `kms:DescribeKey`— Fornece detalhes da chave gerenciada pelo cliente para permitir que os clusters elásticos do Amazon DocumentDB validem a chave.
- `kms:Decrypt`— Permite que os clusters elásticos do Amazon DocumentDB usem a chave de dados criptografada armazenada para acessar dados criptografados.
- `kms:GenerateDataKey`— Permite que clusters elásticos do Amazon DocumentDB gerem uma chave de dados criptografada e a armazenem porque a chave de dados não é usada imediatamente para criptografar.

Para obter mais informações, consulte Permissões para AWS serviços nas principais políticas e Solução de problemas de acesso às chaves no Guia do AWS Key Management Service desenvolvedor.

Restringir o acesso à chave gerenciada pelo cliente por meio de políticas do IAM

Além das políticas de chaves KMS, você também pode restringir as permissões da chave KMS em uma política do IAM.

É possível tornar a política do IAM mais rígida de várias maneiras. Por exemplo, para permitir que a chave gerenciada pelo cliente só seja usada para solicitações provenientes do cluster elástico do Amazon DocumentDB, é possível utilizar a chave de condição `kms:ViaService` com o valor `docdb-elastic.<region-name>.amazonaws.com`.

Para obter mais informações, consulte Como permitir que usuários em outras contas usem uma chave do KMS no Guia do desenvolvedor do AWS Key Management Service .

Monitorar suas chaves de criptografia para clusters elásticos do Amazon DocumentDB

Ao usar uma chave gerenciada pelo cliente com seus recursos de cluster elástico do Amazon DocumentDB, você pode usar AWS CloudTrail o Amazon CloudWatch Logs para rastrear solicitações para as quais os clusters elásticos do Amazon DocumentDB enviam. AWS KMS

Os exemplos a seguir são AWS CloudTrail eventos para`CreateGrant`, `GenerateDataKeyWithoutPlainTextDecrypt`, e `DescribeKey` para monitorar AWS KMS key operações chamadas pelos clusters elásticos do Amazon DocumentDB para acessar dados criptografados pela chave gerenciada pelo 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"
}
```


Saiba mais

Os recursos a seguir fornecem mais informações sobre a criptografia de dados em pausa:

- Para obter mais informações sobre AWS KMS conceitos, consulte conceitos AWS Key Management Service básicos no Guia do AWS Key Management Service desenvolvedor.
- Para obter mais informações sobre AWS KMS segurança, consulte Práticas recomendadas de segurança AWS Key Management Service no Guia do AWS Key Management Service desenvolvedor.

Service-linked funções em clusters elásticos

Os clusters elásticos do Amazon DocumentDB usam funções vinculadas a serviços AWS Identity and Access Management (IAM). Uma função vinculada a serviços é um tipo exclusivo de função do IAM vinculada diretamente aos clusters elásticos do Amazon DocumentDB. Service-linked as funções são predefinidas pelos clusters elásticos do Amazon DocumentDB e incluem todas as permissões que o serviço exige para chamar outros AWS serviços em seu nome.

Uma função vinculada ao serviço facilita a utilização dos clusters elásticos do Amazon DocumentDB, já que não é preciso adicionar as permissões necessárias manualmente. Os clusters elásticos do Amazon DocumentDB definem as permissões dos perfis vinculados ao serviço e, a não ser que esteja definido de outra forma, somente os clusters elásticos do Amazon DocumentDB poderão assumir os perfis. As permissões definidas incluem a política de confiança e a política de permissões, que não pode ser anexada a nenhuma outra entidade do IAM. É possível excluir as funções somente depois de primeiro excluir seus recursos relacionados. Isso protege seus recursos dos clusters elásticos do Amazon DocumentDB, pois você não pode remover por engano as permissões necessárias para acessar os recursos.

Para obter informações sobre outros serviços que oferecem suporte a funções vinculadas a AWS serviços, consulte os serviços que funcionam com o IAM e procure os serviços marcados com Sim na coluna Service-Linked Função. Escolha um Sim com um link para visualizar a documentação do perfil vinculado para esse serviço.

Service-linked permissões de função para clusters elásticos

Os clusters elásticos do Amazon DocumentDB usam a função vinculada ao serviço chamada `AWS ServiceRoleForDocDB-Elastic` para permitir que os clusters elásticos do Amazon DocumentDB chamem AWS serviços em nome de seus clusters.

Essa função vinculada a serviços tem uma política de permissões anexada a ela, chamada `AmazonDocDB-ElasticServiceRolePolicy`, que concede permissões para operar na conta.

A política de permissões de função permite que os clusters elásticos do Amazon DocumentDB concluam as seguintes ações nos recursos especificados:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": [
                        "AWS/DocDB-Elastic"
                    ]
                }
            }
        }
    ]
}
```

> **Note**
>
> É necessário configurar as permissões para permitir que uma entidade do IAM (como um usuário, grupo ou perfil) crie, edite ou exclua uma função vinculada ao serviço. Se encontrar a seguinte mensagem de erro: “Não foi possível criar o recurso. Você se você tem permissão para criar o perfil vinculado ao serviço. Caso contrário, aguarde e tente novamente mais tarde.” , verifique se você tem as seguintes permissões ativadas:

```
{
"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 obter mais informações, consulte as permissões de Service-linked função no Guia do usuário do IAM.

Criação de uma função vinculada ao serviço para clusters elásticos do Amazon DocumentDB

Não é necessário criar manualmente um perfil vinculado ao serviço. Quando você cria uma instância de BD, o Amazon DocumentDB elastic clusters cria a função vinculada ao serviço para você.

Editar um perfil vinculado ao serviço para clusters elásticos do Amazon DocumentDB

Os clusters elásticos do Amazon DocumentDB não permitem que você edite a função vinculada ao serviço `AWS ServiceRoleForDocDB-Elastic`. Depois de criar uma função vinculada a um serviço, você não poderá alterar o nome da função porque várias entidades poderão fazer referência a ela. No entanto, será possível editar a descrição do perfil usando o IAM. Para saber mais, consulte Editar um perfil vinculado ao serviço no Guia do usuário do IAM.

Exclusão de uma função vinculada ao serviço para clusters elásticos do Amazon DocumentDB

Se você não precisar mais usar um recurso ou serviço que exija uma função vinculada ao serviço, exclua essa função. Dessa forma, você não terá uma entidade não utilizada que não seja ativamente monitorada ou mantida. No entanto, você deve excluir todos os clusters antes de excluir a função vinculada ao serviço.

Limpar uma função vinculada ao serviço

Antes de usar o IAM para excluir uma função vinculada a um serviço, você deve primeiro confirmar que a função não tem sessões ativas e remover todos os recursos usados por ela.

Para verificar se a função vinculada ao serviço tem uma sessão ativa no console do IAM:

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM.
2. No painel de navegação do console do IAM, escolha Funções. A seguir, selecione o nome (não a caixa de seleção) da função `AWS ServiceRoleForDocDB-Elastic`.
3. Na página Resumo do perfil escolhido, escolha a guia Consultor de acesso.

> **Note**
>
> Se não tiver certeza se os clusters elásticos do Amazon DocumentDB estão usando a função `AWS ServiceRoleForDocDB-Elastic`, é possível tentar excluir a função. Se o serviço estiver usando a função, a exclusão falhará e você poderá ver Regiões da AWS onde a função está sendo usada. Se a função está sendo usada, você deve aguardar a sessão final antes de excluir a função. Não é possível revogar a sessão de uma função vinculada a um serviço.
>
> Se quiser remover a função `AWS ServiceRoleForDocDB-Elastic`, você deve primeiro excluir todos os clusters.

Exclusão de todos os Clusters

Para excluir um cluster no console do Amazon DocumentDB:

1. Faça login Console de gerenciamento da AWS e abra o console do Amazon DocumentDB.
2. No painel de navegação, escolha Clusters.
3. Escolha o cluster que você deseja excluir.
4. Em Ações, escolha Excluir.
5. Se for exibido Criar snapshot final?, escolha Sim ou Não.
6. Se você escolher Sim na etapa anterior, em Nome do snapshot final, digite o nome do snapshot final.
7. Escolha Excluir.

> **Note**
>
> É possível usar o console, a CLI ou a API do IAM para excluir o perfil vinculado ao serviço `AWS ServiceRoleForDocDB-Elastic`. Para saber mais, consulte Excluir um perfil vinculado ao serviço no Guia do usuário do IAM.

Cotas do Amazon DocumentDB

Este tópico descreve as cotas de recursos e as restrições de nomenclatura do Amazon DocumentDB (com compatibilidade com o MongoDB).

Para alguns recursos de gerenciamento, o Amazon DocumentDB usa a tecnologia operacional que é compartilhada com o Amazon Relational Database Service (Amazon RDS) e o Amazon Neptune.

Tópicos

- Tipos de instâncias compatíveis
- Regiões aceitas
- Cotas regionais
- Cotas de agregação
- Cotas de cluster
- Cotas de instâncias
- Restrições de nomenclatura
- Restrições de TTL
- Cotas elásticas de clusters
- Cotas elásticas de fragmentos de cluster
- Cotas de CPU, memória, conexão e cursor de cluster elástico por fragmento

Tipos de instâncias compatíveis

O Amazon DocumentDB oferece suporte a instâncias sob demanda e aos seguintes tipos de instâncias:

- NVMe-backed:
  - Tipos de instância R6GD: `db.r6gd.xlarge`, `db.r6gd.2xlarge`, `db.r6gd.4xlarge`, `db.r6gd.8xlarge`, `db.r6gd.12xlarge`, `db.r6gd.16xlarge`.
- Otimizado para memória:
  - Tipos de instância R8G: `db.r8g.large`,`db.r8g.xlarge`,`db.r8g.2xlarge`,`db.r8g.4xlarge`,,`db.r8g.8xlarge`, `db.r8g.12xlargedb.r8g.16xlarge`,`db.r8g.24xlarge`. `db.r8g.48xlarge`

  - Tipos de instância R6G: `db.r6g.large`,`db.r6g.xlarge`,,`db.r6g.2xlarge`, `db.r6g.4xlargedb.r6g.8xlarge`,`db.r6g.12xlarge`. `db.r6g.16xlarge`
  - Tipos de instância R5: `db.r5.large`,`db.r5.xlarge`,`db.r5.2xlarge`,,`db.r5.4xlarge`, `db.r5.8xlargedb.r5.12xlarge`,`db.r5.16xlarge`.
  - Tipos de instância R4: `db.r4.large`, `db.r4.2xlarge`, `db.r4.4xlarge`, `db.r4.8xlarge`, `db.r4.16xlarge`.
- Performance expansível:
  - Tipos de instância T4G: `db.t4g.medium`.
  - Tipos de instância T3: `db.t3.medium`.

Para obter mais informações sobre os tipos de instância compatíveis e suas especificações, consulte Especificações de classe de instância.

Regiões aceitas

O Amazon DocumentDB está disponível no seguinte: Regiões da AWS

Nome da região	Região	Zonas de disponibilidade (computação)
Leste dos EUA (Ohio)	`us-east-2`	3
Leste dos EUA (Norte da Virgínia)	`us-east-1`	6
Oeste dos EUA (Oregon)	`us-west-2`	4
África (Cidade do Cabo)	`af-south-1`	3
América do Sul (São Paulo)	`sa-east-1`	3
Ásia-Pacífico (Hong Kong)	`ap-east-1`	3
Ásia-Pacífico (Hyderabad)	`ap-south-2`	3
Ásia-Pacífico (Malásia)	`ap-southeast-5`	3
Ásia-Pacífico (Mumbai)	`ap-south-1`	3
Ásia-Pacífico (Osaka)	`ap-northeast-3`	3
Ásia-Pacífico (Seul)	`ap-northeast-2`	4
Ásia-Pacífico (Singapura)	`ap-southeast-1`	3

Nome da região	Região	Zonas de disponibilidade (computação)
Ásia-Pacífico (Sydney)	ap-southeast-2	3
Ásia-Pacífico (Jacarta)	ap-southeast-3	3
Ásia-Pacífico (Melbourne)	ap-southeast-4	3
Ásia-Pacífico (Tailândia)	ap-southeast-7	3
Ásia-Pacífico (Tóquio)	ap-northeast-1	3
Canadá (Central)	ca-central-1	3
Oeste do Canadá (Calgary)	ca-west-1	3
Região China (Pequim)	cn-north-1	3
China (Ningxia)	cn-northwest-1	3
Europa (Frankfurt)	eu-central-1	3
Europa (Zurique)	eu-central-2	3
Europa (Irlanda)	eu-west-1	3
Europa (Londres)	eu-west-2	3
Europa (Milão)	eu-south-1	3
Europa (Paris)	eu-west-3	3
Europa (Espanha)	eu-south-2	3
Europa (Estocolmo)	eu-north-1	3

Nome da região	Região	Zonas de disponibilidade (computação)
México (Centro)	`mx-central-1`	3
Oriente Médio (Emirados Árabes Unidos)	`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

Cotas regionais

Para alguns recursos de gerenciamento, o Amazon DocumentDB usa a tecnologia operacional que é compartilhada com o Amazon Relational Database Service (Amazon RDS). A tabela a seguir contém cotas regionais que são compartilhadas entre o Amazon DocumentDB e o Amazon RDS.

> **Note**
>
> A tecnologia compartilhada do Amazon RDS descrita acima se aplica somente aos clusters baseados em instâncias do Amazon DocumentDB. Os clusters elásticos do Amazon DocumentDB não compartilham tecnologia com o Amazon RDS.

As cotas a seguir se aplicam aos clusters baseados em instâncias do Amazon DocumentDB e são por região. Conta da AWS

Recurso	AWS cota padrão
Clusters	40
Grupos de parâmetros de clusters	50

Recurso	AWS cota padrão
Clusters por grupo de parâmetros	100
Instâncias por grupo de parâmetros	300
Assinaturas de eventos	20
Instâncias	40
Snapshots de clusters manuais	100
Réplicas de leitura por cluster	15
Grupos de sub-rede	50
Sub-redes por grupo de sub-rede	20
Tags por recurso	50
Grupos de segurança da VPC por instância	5

As cotas a seguir se aplicam aos clusters elásticos do Amazon DocumentDB e são Conta da AWS por região.

Recurso	AWS cota padrão
Clusters elásticos	20
vCPU de clusters elásticos	1024
Snapshot de cluster elástico manual	20

É possível usar o Service Quotas para solicitar um aumento para uma cota, se a cota for ajustável. Algumas solicitações são resolvidas automaticamente, enquanto outras são enviadas para Suporte o. Você pode acompanhar o status de uma solicitação de aumento de cota enviada para o. Suporte As solicitações para aumentar as cotas de serviço não recebem suporte prioritário. Se você tiver uma solicitação urgente, entre em contato Suporte. Para obter mais informações sobre o Service Quotas, consulte O que é o Service Quotas?

Para solicitar um aumento de cota para o Amazon DocumentDB:

1. Abra o console do Service Quotas em https://console.aws.amazon.com/servicequotas e, se necessário, entre.
2. No painel de navegação, escolha Serviços da AWS .
3. Selecione Amazon DocumentDB (compatível com MongoDB) ou Cluster elástico do Amazon DocumentDB na lista ou digite um dos dois no campo de pesquisa.
4. Se a cota for ajustável, será possível selecionar seu botão de opção ou seu nome e escolher Solicitar aumento de cota no canto superior direito da página.
5. Em Change quota value (Alterar valor da cota), insira o novo valor. O novo valor deve ser maior que o valor atual.
6. Escolha Solicitar. Depois que a solicitação é resolvida, o Applied quota value (Valor da cota aplicada) para a cota é definido como o novo valor.
7. Para exibir quaisquer solicitações pendentes ou resolvidas recentemente, escolha Dashboard (Painel) no painel de navegação. Para solicitações pendentes, escolha o status da solicitação para abrir o recibo da solicitação. O status inicial de uma solicitação é `Pending`. Depois que o status mudar para`Quota requested`, você verá o número do caso com Suporte. Escolha o número do caso para abrir o tíquete de sua solicitação.

Cotas de agregação

A tabela a seguir descreve as cotas de agregação no Amazon DocumentDB.

Recurso	Quota
Número máximo compatível de estágios	500

Cotas de cluster

A tabela a seguir descreve as cotas de cluster baseadas em instâncias do Amazon DocumentDB.

Recurso	Quota
Tamanho do cluster (soma de todas as coleções e índices)	256 TiB para a versão 8.0 e superior do motor (128 TiB para versões anteriores do motor)
Tamanho da coleção (a soma de todas as coleções não pode exceder a cota do cluster) — não inclui o tamanho do índice	32 TiB
Coleções por cluster	100.000
Bancos de dados por cluster	100.000
Tamanho do banco de dados (a soma de todos os bancos de dados não pode exceder a cota do cluster)	256 TiB para a versão 8.0 e superior do motor (128 TiB para versões anteriores do motor)
Profundidade do aninhamento de documentos	200 níveis
Tamanho do documento	16 MiB
Tamanho da chave de índice	2.048 bytes
Índices por coleção	64

Recurso	Quota
Chaves em um índice composto	32
O número máximo de gravações em um único comando em lote	100.000
Número de usuários por cluster	1000

Cotas de instâncias

A tabela a seguir descreve as cotas do Amazon DocumentDB por instância.

Amazon DocumentDB 8.0

Tipo de instância	Memória da instância (GiB)	Conexões (todas)	Cota de cursor	Abrir transações	Conexões (ativas)
T3.medium	4	1000	30	50	102
T4G.medium	4	1000	30	50	102
R5.large	16	3400	450	200	1100
R5.xlarge	32	7000	450	400	2700
R5.2xlarge	64	14200	450	800	4500
R5.4xlarge	128	28400	760	1600	4500
R5.8xlarge	256	60000	1520	3200	4500
R5.12xlarge	384	60000	2280	4800	4500
R5.16xlarge	512	60000	3040	6400	4500
R6G.large*	16	3400	450	200	1100
R6G.xlarge*	32	7000	450	400	2700
R6G.2xlarge*	64	14200	450	800	4500
R6G.4xlarge*	128	28400	760	1600	4500
R6G.8xlarge*	256	60000	1520	3200	4500

Tipo de instância	Memória da instância (GiB)	Conexões (todas)	Cota de cursor	Abrir transações	Conexões (ativas)
R6G.12xlarge*	384	60000	2280	4800	4500
R6G.16xlarge*	512	60000	3040	6400	4500
R8G.large	16	3400	450	200	1100
R8G.xlarge	32	7000	450	400	2700
R8G.2xlarge	64	14200	450	800	4500
R8G.4xlarge	128	28400	760	1600	4500
R8G.8xlarge	256	60000	1520	3200	4500
R8G.12xlarge	384	60000	2280	4800	4500
R8G.16xlarge	512	60000	3040	6400	4500
R8G.24xlarge	768	60000	4560	9600	4500
R8G.48xlarge	1536	60000	9120	9600	4500

* incluindo R6GD

Amazon DocumentDB 5.0

Tipo de instância	Memória da instância (GiB)	Conexões (todas)	Cota de cursor	Abrir transações	Conexões (ativas)
T3.medium	4	1000	30	50	102
T4G.medium	4	1000	30	50	102
R5.large	16	3400	450	200	1100
R5.xlarge	32	7000	450	400	2700
R5.2xlarge	64	14200	450	800	4500
R5.4xlarge	128	28400	760	1600	4500
R5.8xlarge	256	60000	1520	3200	4500
R5.12xlarge	384	60000	2280	4800	4500
R5.16xlarge	512	60000	3040	6400	4500
R6G.large*	16	3400	450	200	1100
R6G.xlarge*	32	7000	450	400	2700
R6G.2xlarge*	64	14200	450	800	4500
R6G.4xlarge*	128	28400	760	1600	4500
R6G.8xlarge*	256	60000	1520	3200	4500
R6G.12xlarge*	384	60000	2280	4800	4500

Tipo de instância	Memória da instância (GiB)	Conexões (todas)	Cota de cursor	Abrir transações	Conexões (ativas)
R6G.16xlarge*	512	60000	3040	6400	4500
R8G.large	16	3400	450	200	1100
R8G.xlarge	32	7000	450	400	2700
R8G.2xlarge	64	14200	450	800	4500
R8G.4xlarge	128	28400	760	1600	4500
R8G.8xlarge	256	60000	1520	3200	4500
R8G.12xlarge	384	60000	2280	4800	4500
R8G.16xlarge	512	60000	3040	6400	4500
R8G.24xlarge	768	60000	4560	9600	4500
R8G.48xlarge	1536	60000	9120	9600	4500

* incluindo R6GD

Amazon DocumentDB 4.0

Tipo de instância	Memória da instância (GiB)	Conexões (todas)	Cota de cursor	Abrir transações	Conexões (ativas)
T3.medium	4	1000	30	50	102

Tipo de instância	Memória da instância (GiB)	Conexões (todas)	Cota de cursor	Abrir transações	Conexões (ativas)
T4G.medium	4	1000	30	50	102
R5.large	16	3400	450	200	1100
R5.xlarge	32	7000	450	400	2700
R5.2xlarge	64	14200	450	800	4500
R5.4xlarge	128	28400	760	1600	4500
R5.8xlarge	256	60000	1520	3200	4500
R5.12xlarge	384	60000	2280	4800	4500
R5.16xlarge	512	60000	3040	6400	4500
R6G.large*	16	3400	450	200	1100
R6G.xlarge*	32	7000	450	400	2700
R6G.2xlarge*	64	14200	450	800	4500
R6G.4xlarge*	128	28400	760	1600	4500
R6G.8xlarge*	256	60000	1520	3200	4500
R6G.12xlarge*	384	60000	2280	4800	4500
R6G.16xlarge*	512	60000	3040	6400	4500

* incluindo R6GD

Amazon DocumentDB 3.6

Tipo de instância	Memória da instância (GiB)	Conexões (todas)	Cota de cursor	Abrir transações	Conexões (ativas)
T3.medium	4	500	30	N/A	102
R4.large	15.25	1700	450	N/A	1100
R4.xlarge	30.5	3400	450	N/A	2700
R4.2xlarge	61	6800	450	N/A	4500
R4.4xlarge	122	13600	725	N/A	4500
R4.8xlarge	288	27200	1450	N/A	4500
R4.16xlarge	488	30000	2900	N/A	4500
R5.large	16	1700	450	N/A	1100
R5.xlarge	32	3500	450	N/A	2700
R5.2xlarge	64	7100	450	N/A	4500
R5.4xlarge	128	14200	760	N/A	4500
R5.8xlarge	256	28400	1520	N/A	4500
R5.12xlarge	384	30000	2280	N/A	4500
R5.16xlarge	512	30000	3040	N/A	4500

Você pode monitorar e alertar sobre os limites por instância usando as CloudWatch métricas a seguir. Para obter mais informações sobre as CloudWatch métricas do Amazon DocumentDB, consulteMonitorando o Amazon DocumentDB com CloudWatch.

Recurso	CloudWatch métrica de cota	CloudWatch métrica de uso (máximo de 1 min.)	CloudWatch métrica de uso
Memória da instância	-	-	FreeableMemory
Conexões (todas)	DatabaseConnectionsLimit	DatabaseConnectionsMax	DatabaseConnections
Cursores	DatabaseCursorsLimit	DatabaseCursorsMax	DatabaseCursors
Transações	TransactionsOpenLimit	TransactionsOpenMax	TransactionsOpen

Restrições de nomenclatura

A tabela a seguir descreve restrições de nomenclatura no Amazon DocumentDB.

Recurso	Cota padrão
Identificador de Cluster	• O comprimento é de [1-63] letras, números ou hifens. • O primeiro caractere deve ser uma letra. • Não pode terminar com um hífen ou conter dois hifens consecutivos. • Deve ser exclusivo para todos os clusters (no Amazon RDS, Amazon Neptune e Amazon DocumentDB) por AWS conta, por região.
Identificador da instância	• O comprimento é de [1-63] letras, números ou hifens • O primeiro caractere deve ser uma letra • Não podem terminar com um hífen ou conter dois hífens consecutivos • Deve ser exclusivo para todas as instâncias (no Amazon RDS, Amazon Neptune e Amazon DocumentDB) por AWS conta, por região.
Nome da coleção: `<col>`	O comprimento é de [1—255] caracteres (Amazon DocumentDB 5.0 e versões posteriores).

Recurso	Cota padrão
	O comprimento é de [1—57] caracteres (Amazon DocumentDB 4.0 e versões anteriores).
Nome do banco de dados: <db>	O comprimento é [1-63] caracteres.
Nome da coleção totalmente qualificado: <db>.<col>	O comprimento é de [3—255] caracteres (Amazon DocumentDB 5.0 e versões posteriores). O comprimento é de [3—120] caracteres (Amazon DocumentDB 4.0 e versões anteriores).
Nome do índice totalmente qualificado: <db>.<col>$<index>	O comprimento é de [6—511] caracteres (Amazon DocumentDB 5.0 e versões posteriores). O comprimento é de [6—127] caracteres (Amazon DocumentDB 4.0 e versões anteriores).
Nome do índice	O comprimento é de [1—255] caracteres (Amazon DocumentDB 5.0 e versões posteriores). O comprimento é de [3—63] caracteres para o combinado <col>$<index> (Amazon DocumentDB 4.0 e versões anteriores).

Recurso	Cota padrão

Password principal	• O comprimento é de [8–100] caracteres ASCII imprimíveis.
	• Pode usar quaisquer caracteres ASCII imprimíveis, exceto:
	  • **/** (barra)
	  • **"** (aspas duplas)
	  • **@** (arroba)

Nome de usuário principal	• O comprimento é de [1–63] caracteres alfanuméricos. • O primeiro caractere deve ser uma letra. • Não pode ser uma palavra reservada pelo mecanismo de banco de dados.
Nome do parameter group	• O tamanho é de [1–255] caracteres alfanuméricos. • O primeiro caractere deve ser uma letra. • Não pode terminar com um hífen ou conter dois hifens consecutivos.

Restrições de TTL

Exclusões de um índice de TTL não são garantidas dentro de um período específico e têm como base o melhor esforço. Fatores como utilização de recursos da instância, tamanho do documento e throughput geral podem afetar a sincronização de uma exclusão de TTL.

Cotas elásticas de clusters

A tabela a seguir descreve as cotas máximas para clusters elásticos do Amazon DocumentDB.

Recurso	Quota
Clusters elásticos por região	20
vCPU somada em todos os clusters elásticos por região	1024
Instantâneos manuais de cluster por região	20
Fragmentos por cluster	32
Armazenamento por cluster (quando os dados são distribuídos uniformemente por fragmento-chave)	4 PiB
Conexões com o cluster	O valor mais baixo de 300.000 ou o número de fragmentos x a cota de conexão associada à vCPU por fragmento
UnSharded tamanho da coleção	32 TiB
Tamanho da coleção fragmentada (quando os dados são distribuídos uniformemente por fragmento-chave)	1 PB
Bancos de dados por cluster	10.000
UnSharded coleções por cluster	100.000
Coleções fragmentadas por cluster	1000
Usuários por cluster	100
O número máximo de gravações em um único comando em lote	100.000
Índices por coleção	64
Profundidade do aninhamento de documentos	100 níveis

Recurso	Quota
Tamanho do documento	16 MB
Tamanho da chave de índice	2048 bytes
Chaves em um índice composto	32

Cotas elásticas de fragmentos de cluster

A tabela a seguir descreve as cotas máximas de fragmentos para clusters elásticos do Amazon DocumentDB.

Recurso	Quota
vCPU por instância de fragmento	64
Instâncias por frgamento	16
Armazenamento por fragmento	128 TiB
Armazenamento por coleção por fragmento	32 TiB

Cotas de CPU, memória, conexão e cursor de cluster elástico por fragmento

A tabela a seguir descreve os limites máximos de CPU, memória, conexão e cursor nos fragmentos do cluster elástico do Amazon DocumentDB.

vCPUs por fragmento	Memória da instância (GiB)	Cota de conexão	Cota de cursor
2	16	1700	450
4	32	3500	450
8	64	7100	450

vCPUs por fragmento	Memória da instância (GiB)	Cota de conexão	Cota de cursor
16	128	14200	760
32	256	28400	1520
48	384	30000	2280
64	512	30000	3040

Solução de problemas do Amazon DocumentDB

As seções a seguir fornecem informações sobre como solucionar problemas que você pode encontrar ao usar o Amazon DocumentDB (compatível com MongoDB).

Tópicos

- Solução de problemas de conectividade
- Solução de problemas de índices
- Solução de problemas de performance e utilização de recursos
- Coleta de resíduos no Amazon DocumentDB
- Códigos de erro

Solução de problemas de conectividade

Está tendo problemas para se conectar? Aqui estão alguns cenários comuns e como resolvê-los.

Tópicos

- Não é possível conectar-se a um endpoint do Amazon DocumentDB
- Testar uma conexão com uma instância do Amazon DocumentDB
- Conectar a um endpoint inválido
- A configuração do driver afeta o número de conexões

Não é possível conectar-se a um endpoint do Amazon DocumentDB

Quando você tentar se conectar ao Amazon DocumentDB, a mensagem de erro a seguir é uma das mais comuns que você pode receber.

```
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
```

O que essa mensagem de erro normalmente significa é que seu cliente (o shell do mongo, nesse exemplo) não pode acessar o endpoint do Amazon DocumentDB. Este pode ser o caso por vários motivos:

Tópicos

- Conectar de endpoints públicos
- Cross-Region conexões
- Conectar a partir de diferentes Amazon VPCs
- O grupo de segurança bloqueia as conexões de entrada.
- Problema de preferência de leitura do driver Java Mongo

Conectar de endpoints públicos

Você está tentando se conectar a um cluster do Amazon DocumentDB diretamente do seu laptop ou máquina de desenvolvimento local.

Tentar se conectar a um cluster do Amazon DocumentDB diretamente de um endpoint público, como seu laptop ou máquina de desenvolvimento local, falhará. O Amazon DocumentDB é apenas uma nuvem privada virtual (VPC) e não oferece suporte a endpoints públicos no momento. Assim, você não poderá se conectar diretamente ao seu cluster do Amazon DocumentDB do seu laptop nem do ambiente de desenvolvimento local fora da sua VPC.

Para se conectar a um cluster do Amazon DocumentDB de fora de uma Amazon VPC, você pode usar o túnel SSH. Para obter mais informações, consulte Conectar-se a um cluster do Amazon DocumentDB de fora de uma Amazon VPC. Além disso, se o seu ambiente de desenvolvimento estiver em uma Amazon VPC diferente, você também poderá usar o VPC Peering e conectar-se ao seu cluster Amazon DocumentDB a partir de outra Amazon VPC na mesma região ou em uma região diferente.

Cross-Region conexões

Você está tentando se conectar a um cluster do Amazon DocumentDB em outra região.

Se você tentar se conectar a um cluster do Amazon DocumentDB a partir de uma instância do Amazon EC2 em uma região diferente daquela do cluster — por exemplo, tentando se conectar a um cluster na região Leste dos EUA (Norte da Virgínia) (us-east-1) a partir da região Oeste dos EUA (Oregon) (us-west-2) - a conexão fracassará.

Para verificar a região do seu cluster do Amazon DocumentDB, execute o comando a seguir. A região está no endpoint.

```
aws docdb describe-db-clusters \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].Endpoint'
```

A saída dessa operação é semelhante à seguinte.

```
[
    "sample-cluster.node.us-east-1.docdb.amazonaws.com"
]
```

Para verificar a região da sua instância do EC2, execute o comando a seguir.

```
 aws ec2 describe-instances \
     --query 'Reservations[*].Instances[*].Placement.AvailabilityZone'
```

A saída dessa operação é semelhante à seguinte.

```
[
    [
        "us-east-1a"
    ]
]
```

Conectar a partir de diferentes Amazon VPCs

Você está tentando se conectar a um cluster do Amazon DocumentDB a partir de uma VPC diferente da Amazon VPC na qual seu cluster está implantado.

Se o cluster do Amazon DocumentDB e a instância do Amazon EC2 estiverem na mesma Amazon VPC Região da AWS, mas não na mesma Amazon VPC, você não poderá se conectar diretamente ao seu cluster Amazon DocumentDB, a menos que o VPC Peering esteja habilitado entre as duas Amazon VPCs.

Para verificar a Amazon VPC de sua instância do Amazon DocumentDB, execute o comando a seguir.

```
aws docdb describe-db-instances \
   --db-instance-identifier sample-instance \
   --query 'DBInstances[*].DBSubnetGroup.VpcId'
```

Para verificar a Amazon VPC da sua instância do Amazon EC2, execute o comando a seguir.

```
aws ec2 describe-instances \
   --query 'Reservations[*].Instances[*].VpcId'
```

O grupo de segurança bloqueia as conexões de entrada.

Você está tentando se conectar a um cluster do Amazon DocumentDB e o grupo de segurança do cluster não permite conexões de entrada na porta do cluster (porta padrão: 27017).

Suponha que o cluster do Amazon DocumentDB e a instância do Amazon EC2 estejam na mesma região e Amazon VPC e usem o mesmo grupo de segurança da Amazon VPC. Se você não conseguir se conectar ao cluster do Amazon DocumentDB, a causa provável é que o grupo de segurança (ou seja, firewall) do cluster não permite conexões de entrada na porta escolhida para o cluster do Amazon DocumentDB (a porta padrão é 27017).

Para verificar a porta do seu cluster do Amazon DocumentDB, execute o comando a seguir.

```
aws docdb describe-db-clusters \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].[DBClusterIdentifier,Port]'
```

Para obter o seu grupo de segurança do Amazon DocumentDB para o seu cluster, execute o seguinte comando.

```
aws docdb describe-db-clusters \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].[VpcSecurityGroups[*],VpcSecurityGroupId]'
```

Para verificar as regras de entrada do seu grupo de segurança, consulte os seguintes tópicos na documentação do Amazon EC2:

- Autorizar tráfego de entrada para as instâncias do Linux

- Autorizar tráfego de entrada para as instâncias do Windows

Problema de preferência de leitura do driver Java Mongo

As preferências de leitura do cliente não são respeitadas e alguns clientes não podem gravar no Amazon DocumentDB após o failover, a menos que sejam reinicializados.

Esse problema, descoberto pela primeira vez no Java Mongo Driver 3.7.x, ocorre quando um cliente estabelece uma conexão com o Amazon DocumentDB usando `MongoClientSettings` e, especificamente, ao encadear o método. `applyToClusterSettings` As configurações do MongoClient cluster podem ser definidas usando alguns métodos diferentes`hosts()`, como`requiredReplicaSetName()`, `mode()` e.

Quando o cliente especifica apenas um host no método `hosts()`, o modo é definido como em `ClusterConnectionMode.SINGLE` vez de `ClusterConnectionMode.MULTIPLE` Isso faz com que o cliente desconsidere a preferência de leitura e se conecte apenas ao servidor configurado em `hosts()`. Portanto, mesmo que as configurações do cliente sejam inicializadas como abaixo, todas as leituras ainda irão para a primária em vez da secundária.

```
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 failover

Usando as configurações de conexão do cliente acima, se houver um failover e uma atualização atrasada do registro DNS para o endpoint do gravador do cluster, o cliente ainda tentará emitir gravações no gravador antigo (agora leitor após o failover). Isso resulta em um erro do lado do

servidor (não principal) que não é tratado adequadamente pelo driver Java (isso ainda está sob investigação). Assim, o cliente pode ficar em mau estado até que o servidor da aplicação seja reinicializado, por exemplo.

Há duas soluções alternativas para isso:

- Clientes que se conectam ao Amazon DocumentDB por meio de uma cadeia de conexão não terão esse problema, pois `ClusterConnectionMode` serão configurados como `MULTIPLE` ao definir a preferência de leitura.

```
MongoClientURI mongoClientURI = new MongoClientURI("mongodb://usr:pass:cluster-
endpoint:27317/test?ssl=false&replicaSet=rs0&readpreference=secondaryPreferred");
MongoClient mongoClient = MongoClients.create(mongoClientURI.getURI());
```

  Ou usando o construtor `MongoClientSettings` com o 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);
```

- Definido explicitamente de `ClusterConnectionMode` para `MULTIPLE`. Isso só é necessário ao usar `applyToClusterSettings` e `hosts().size() == 1`.

```
final ServerAddress serverAddress0 = new ServerAddress("cluster-endpoint", 27317));
final MongoCredential credential = MongoCredential.createCredential("xxx","admin",
 "xxxx".toCharArray());
final MongoClientSettings settings = MongoClientSettings.builder()
    .credential(credential)
    .readPreference(ReadPreference.secondaryPreferred())
    .retryWrites(false)
    .applyToSslSettings(builder # builder
    .enabled(false))
    .applyToClusterSettings(builder # builder
                .hosts(Arrays.asList(serverAddress0))
                .requiredReplicaSetName("rs0"))
                .mode(ClusterConnectionMode.MULTIPLE))
```

```
    .build();
MongoClient mongoClient = MongoClients.create(settings);
```

Testar uma conexão com uma instância do Amazon DocumentDB

É possível testar sua conexão com um cluster usando ferramentas comuns do Linux ou Windows.

Em um terminal Linux ou Unix, teste a conexão inserindo o seguinte (substitua `cluster-endpoint` pelo endpoint e `port` pela porta da sua instância):

```
nc -zv cluster-endpoint port
```

Veja a seguir um exemplo de operação e o valor de retorno:

```
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!
```

Conectar a um endpoint inválido

Ao conectar-se a um cluster do Amazon DocumentDB e usar um endpoint de cluster inválido, um erro semelhante ao seguinte será exibido.

```
mongo --ssl \
   --host sample-cluster.node.us-east-1.docdb.amazonaws.com:27017 \
   --sslCAFile global-bundle.pem \
   --username <user-name> \
   --password <password>
```

O resultado se parece com:

```
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 obter o endpoint válido para um cluster, execute o seguinte comando:

```
aws docdb describe-db-clusters \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].[Endpoint,Port]'
```

Para obter o endpoint válido para uma instância, execute o seguinte comando:

```
aws docdb describe-db-instances \
   --db-instance-identifier sample-instance \
   --query 'DBInstances[*].[Endpoint.Address,Endpoint.Port]'
```

Para obter mais informações, consulte Noções básicas sobre endpoints do Amazon DocumentDB.

A configuração do driver afeta o número de conexões

Ao usar o driver do cliente para se conectar a um cluster do Amazon DocumentDB, é importante considerar o parâmetro de configuração `maxPoolSize`. A configuração `maxPoolSize` determina o número máximo de conexões que o driver do cliente manterá em seu pool de conexões.

Solução de problemas de índices

Os tópicos a seguir abordam o que fazer se a criação do índice ou índice em segundo plano falhar.

Tópicos

- Criação de índice fracassa
- Problemas e falhas de latência de criação de índice em segundo plano
- Sobrecarga de índices de banco de dados

Criação de índice fracassa

O Amazon DocumentDB utiliza armazenamento local em uma instância como parte do processo de criação do índice. Você pode monitorar o uso desse disco usando a FreeLocalStorage CloudWatch métrica (`CloudWatch -> Metrics -> DocDB -> Instance Metrics`). Quando a criação do índice consumia todo o disco local e falhar, você receberá um erro. Ao migrar dados para o

Amazon DocumentDB, recomendamos que você crie índices primeiro e, depois, insira os dados. Para obter mais informações sobre estratégias de migração e criação de índices, consulte Migração e atualização do Amazon DocumentDB na documentação do Amazon DocumentDB e no blog: Migrar do MongoDB para o Amazon DocumentDB usando o método offline.

Ao criar índices em um cluster existente, se a criação do índice estiver demorando mais do que o esperado ou estiver falhando, amplie a instância para criar o índice e, em seguida, reduza a escala após a criação do índice. O Amazon DocumentDB permite que você escale rapidamente os tamanhos das instâncias em minutos usando o Console de gerenciamento da AWS ou o. AWS CLI Para obter mais informações, consulte Gerenciar classes de instância. Com a definição de preço de instâncias por segundo, você paga apenas pelo recurso usado por segundo.

Problemas e falhas de latência de criação de índice em segundo plano

As compilações de índice em segundo plano no Amazon DocumentDB não são iniciadas até que todas as consultas na instância primária iniciadas antes do início da criação do índice sejam concluídas. Se houver uma consulta de longa duração, as compilações do índice em segundo plano serão bloqueadas até que a consulta seja concluída e, portanto, podem levar mais tempo do que o esperado para serem concluídas. Isso será verdadeiro mesmo que as coleções estejam vazias.

As compilações de índice em primeiro plano não exibem o mesmo comportamento de bloqueio. Em vez disso, as compilações de índice em primeiro plano têm um bloqueio exclusivo na coleção até que a construção do índice seja concluída. Para criar índices em uma coleção vazia e evitar o bloqueio de consultas de longa duração, use compilações de índice em primeiro plano.

> **Note**
>
> O Amazon DocumentDB permite que apenas uma compilação de índice de segundo plano ocorra em uma coleção em um determinado momento. Se as operações de DDL (Linguagem de definição de dados) como `createIndex()` ou `dropIndex()` ocorrerem na mesma coleção durante uma compilação de índice de segundo plano, essa compilação falhará.

Sobrecarga de índices de banco de dados

O Amazon DocumentDB usa o Controle de Multi-Version Concorrência (MVCC) para gerenciar transações simultâneas. Quando os documentos são excluídos ou atualizados, suas versões anteriores permanecem em coleções e índices como versões “inativas”. O processo de coleta de resíduos recupera automaticamente o espaço dessas versões inativas para futuras operações.

A sobrecarga de índices ocorre quando os índices de uma coleção se tornam maiores devido ao acúmulo de entradas de índice inativas ou obsoletas ou à fragmentação nas páginas. O percentual relatado representa a quantidade de espaço de índices que pode ser usada por futuras entradas de índice. Essa sobrecarga consome espaço no cache e no armazenamento do buffer. Se você quiser remover a sobrecarga, precisará reconstruir os índices.

Example Exemplo

Execute o comando apresentado a seguir para determinar o armazenamento não utilizado do seu índice:

```
db.coll.aggregate({$indexStats:{}});
```

O resultado deve ser semelhante a este:

```
{
    "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
    }
}
```

É possível recriar índices sem tempo de inatividade usando o comando `reIndex`, que exige uma verificação de toda a coleção. Consulte Manutenção do índice usando `reIndex`.

Solução de problemas de performance e utilização de recursos

Esta seção fornece perguntas e soluções para problemas comuns de diagnóstico em implantações do Amazon DocumentDB. Os exemplos fornecidos usam o shell do mongo e têm escopo definido como uma instância individual. Para encontrar um endpoint de instância, consulte Noções básicas sobre endpoints do Amazon DocumentDB.

Tópicos

- Como determino o número de operações de inserção, atualização e exclusão realizadas na minha coleção por meio da API do Mongo?
- Como faço para analisar o desempenho do cache?
- Como faço para localizar e encerrar consultas bloqueadas ou de longa execução?
- Como posso ver um plano de consulta e otimizar uma consulta?
- Como posso ver um plano de consulta em clusters elásticos?
- Como faço para listar todas as operações em execução em uma instância?
- Como posso saber quando uma consulta está progredindo?
- Como determino por que um sistema fica lento repentinamente?
- Como determino a causa da alta utilização de CPU em uma ou mais instâncias de cluster?
- Como posso determinar os cursores abertos em uma instância?
- Como determinar a versão atual do mecanismo do Amazon DocumentDB?
- Como analiso o uso do índice e identifico índices não utilizados?
- Como identifico índices ausentes?
- Como faço para determinar a sobrecarga da coleção de banco de dados?
- Resumo de consultas úteis

Como determino o número de operações de inserção, atualização e exclusão realizadas na minha coleção por meio da API do Mongo?

Para ver o número de operações de inserção, atualização e exclusão realizadas em uma determinada coleção, execute o seguinte comando nessa coleção:

```
db.collection.stats()
```

A saída deste comando descreve o seguinte no seu campo `opCounters`:

- num DocsIns - O número de documentos inseridos nessa coleção. Isso inclui documentos inseridos usando os comandos `insert` e `insertMany`, bem como documentos inseridos por um upsert.
- num DocsUpd - O número de atualizações de documentos nesta coleção. Isso inclui documentos atualizados usando os comandos `update` e `findAndModify`.
- num DocsDel - O número de documentos excluídos dessa coleção. Isso inclui documentos excluídos usando os comandos `deleteOne`, `deleteMany`, `remove` e `findAndModify`.
- lastReset: o horário em que esses contadores foram redefinidos pela última vez. As estatísticas fornecidas por esse comando são redefinidas quando starting/stopping o cluster ou a instância é up/down escalada.

Um exemplo de resultado da execução `db.collection.stats()` é mostrado abaixo.

```
{
    "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)
}
```

Esse comando “stats” deve ser usado ao visualizar contadores específicos da coleção para operações de inserção, atualização e exclusão por meio da API do Mongo. Outra forma de visualizar os contadores de operações específicos da coleção é habilitar a auditoria de DML. O número de operações de inserção, atualização e exclusão em todas as coleções durante intervalos de um minuto pode ser visualizado em Monitorando o Amazon DocumentDB com CloudWatch.

Como faço para analisar o desempenho do cache?

A análise do desempenho do cache pode fornecer informações sobre a eficiência da recuperação de dados e o desempenho do sistema e é baseada na quantidade de dados lidos do disco em comparação aos do do cache. Fornecemos estatísticas de cache sobre o número de acessos (dados lidos do cache) e falhas de cache (dados que não são encontrados no cache e lidos do disco) para fornecer informações sobre o desempenho do cache. As estatísticas de cache de uma coleção específica podem ser encontradas executando o seguinte comando nessa coleção:

```
db.collection.stats()
```

Os valores no campo `cacheStats` de resultados desse comando fornecem estatísticas de cache para a coleção, bem como as estatísticas totais de cache para os índices criados na coleção. Essas estatísticas estão listadas abaixo:

- **`collBlksHit`**: número de blocos lidos do cache durante as operações nessa coleção.
- **`collBlksRead`**: número de blocos lidos do disco (falha de cache) durante as operações nessa coleção.
- **`collHitRatio`**: taxa de acertos do cache para essa coleção (`100 * [collBlksHit / (collBlksHit + collBlksRead)]`).
- **`idxBlksHit`**: número de blocos lidos do cache para qualquer índice criado nessa coleção.
- **`idxBlksRead`**: número de blocos lidos do disco (falha de cache) para qualquer índice criado nessa coleção.

- **idxHitRatio**: taxa de acerto do cache para os índices criados nessa coleção (`100 * [idxBlksHit / (idxBlksHit + idxBlksRead)]`).
- **lastReset**: horário em que essas estatísticas foram redefinidas pela última vez. As estatísticas fornecidas pelo `db.collection.stats()` são redefinidas quando starting/stopping o cluster ou a instância é up/down escalada.

Um detalhamento dos campos `idxBlksHit` e `idxBlksRead` de cada índice também pode ser encontrado usando-se o comando `indexStats`. As estatísticas de cache específicas do índice podem ser encontradas executando-se o seguinte comando:

```
db.collection.aggregate([{$indexStats:{}}]).pretty()
```

Para cada índice, as seguintes estatísticas de cache podem ser encontradas no campo `cacheStats`:

- **blksHit**: número de blocos lidos do cache para esse índice.
- **blksRead**: número de blocos lidos do disco para esse índice.
- **blksHitRatio**: taxa de acertos do cache arredondada para quatro casas decimais, calculada por `100 * [blksHit / (blksHit + blksRead)]`.

Como faço para localizar e encerrar consultas bloqueadas ou de longa execução?

As consultas de usuário podem ser executadas lentamente devido a um plano de consulta pouco ideal ou podem ser bloqueadas devido à contenção de recursos.

Para encontrar consultas de longa execução que ficam lentas devido a um plano de consulta pouco ideal ou consultas bloqueadas devido à contenção de recursos, use o comando `currentOp`.
É possível filtrar o comando para ajudar a refinar a lista de consultas relevantes para serem encerradas. É necessário ter `opid` associado à consulta de longa duração para poder encerrar a consulta.

A consulta a seguir usa o comando `currentOp` para listar todas as consultas que estão bloqueadas ou em execução por mais 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: {}
});
```

Depois, você pode restringir a consulta para encontrar o `opid` de uma consulta em execução por mais de 10 segundos e encerrá-lo.

Para localizar e encerrar uma consulta em execução por mais de 10 segundos

1. Encontre o `opid` da consulta.

   ```
   db.adminCommand({
       aggregate: 1,
       pipeline: [
           {$currentOp: {}},
           {$match:
               {$or:
                   [{secs_running: {$gt: 10}},
                    {WaitState: {$exists: true}}]}}],
       cursor: {}
   });
   ```

   A saída dessa operação é semelhante ao seguinte (formato JSON).

   ```
   {
       "waitedMS" : NumberLong(0),
       "cursor" : {
           "firstBatch" : [
               {
                   "opid" : 24646,
                   "secs_running" : 12
               }
           ],
           "id" : NumberLong(0),
           "ns" : "admin.$cmd"
   ```

```
    },
    "ok" : 1
}
```

2. Encerre a consulta usando a operação `killOp`.

```
db.adminCommand({killOp: 1, op: 24646});
```

Como posso ver um plano de consulta e otimizar uma consulta?

Se uma consulta for executada lentamente, pode ser porque a execução da consulta requer uma verificação completa da coleção para escolher os documentos relevantes. Às vezes, criar índices apropriados permitirá que a consulta seja executada mais rapidamente. Para detectar esse cenário e decidir os campos nos quais criar os índices, use o comando `explain`.

> **Note**
>
> O Amazon DocumentDB emula a API do MongoDB 3.6 em um mecanismo de banco de dados com propósito específico que utiliza um sistema de armazenamento distribuído, tolerante a falhas e de autorrecuperação. Como resultado, os planos de consulta e a saída de `explain()` podem diferir entre o Amazon DocumentDB e o MongoDB. Os clientes que desejam ter controle sobre seu plano de consulta podem usar o operador `$hint` para impor a seleção de um índice preferencial.

Execute a consulta que você deseja melhorar com o comando `explain` da seguinte forma.

```
db.runCommand({explain: {<query document>}})
```

Veja a seguir um exemplo de operação.

```
db.runCommand({explain:{
    aggregate: "sample-document",
    pipeline: [{$match: {x: {$eq: 1}}}],
    cursor: {batchSize: 1}}
});
```

A saída dessa operação é semelhante ao seguinte (formato JSON).

```
{
    "queryPlanner" : {
        "plannerVersion" : 1,
        "namespace" : "db.test",
        "winningPlan" : {
            "stage" : "COLLSCAN"
        }
    },
    "serverInfo" : {
        "host" : "...",
        "port" : ...,
        "version" : "..."
    },
    "ok" : 1
}
```

A saída acima indica que o estágio `$match` exige a verificação de toda a coleção e a verificação se o campo `"x"` em cada documento é igual a 1. Se houver muitos documentos na coleção, a verificação da coleção (e, portanto, o desempenho geral da consulta) serão muito lentos. Assim, a presença de `"COLLSCAN"` na saída do comando `explain` indica que o desempenho da consulta pode ser aprimorado com a criação de índices apropriados.

Neste exemplo, a consulta verifica se o campo `"x"` é igual a 1 em todos os documentos. Portanto, criar um campo de índice no `"x"` permite que a consulta evite a verificação completa da coleção e use o índice para retornar os documentos relevantes mais cedo.

Depois de criar um índice no campo `"x"`, a saída `explain` será a seguinte.

```
{
    "queryPlanner" : {
         "plannerVersion" : 1,
         "namespace" : "db.test",
         "winningPlan" : {
             "stage" : "IXSCAN",
             "indexName" : "x_1",
             "direction" : "forward"
         }
    },
    "serverInfo" : {
        "host" : "...",
        "port" : ...,
        "version" : "..."
```

```
    },
    "ok" : 1
}
```

A criação de um índice no campo `"x"` permite que o estágio `$match` use uma varredura de índice para reduzir o número de documentos nos quais o predicado `"x = 1"` precisa ser avaliado.

Para pequenas coleções, o processador de consultas do Amazon DocumentDB pode optar por não usar um índice se os ganhos de desempenho forem negligenciáveis.

Como posso ver um plano de consulta em clusters elásticos?

Para examinar um plano de consulta em clusters elásticos, use o comando `explain`. Veja a seguir um exemplo de operação `explain` em uma consulta de busca direcionada a uma coleção fragmentada:

```
db.runCommand(
   {
     explain: { find: "cities", filter: {"name": "Seoul"}}
   }
)
```

> **Note**
>
> O Amazon DocumentDB emula o MongoDB em um mecanismo de banco de dados com propósito específico. Como resultado, os planos de consulta e a saída de `explain()` podem diferir entre o Amazon DocumentDB e o MongoDB. É possível controlar seu plano de consulta com o uso do operador `$hint` para impor a seleção de um índice preferencial.

A saída dessa operação pode ser semelhante ao seguinte (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)
}
```

A saída anterior mostra o plano de consulta para a consulta `find` em um cluster de três fragmentos. Cada fragmento tem várias partições de dados que podem ter diferentes estágios de entrada. Neste exemplo, um “COLLSCAN “(uma varredura de coleções) é executado em todas as partições antes que os resultados sejam mesclados no estágio “PARTITION_MERGE” dentro de cada fragmento. Os resultados dos fragmentos são, então, mesclados no estágio “SHARD_MERGE” antes de serem enviados de volta ao cliente.

Como faço para listar todas as operações em execução em uma instância?

Como usuário ou usuário principal, muitas vezes você deseja listar todas as operações atuais em execução em uma instância para fins de diagnóstico e solução de problemas. (Para obter informações sobre o gerenciamento de usuários, consulte Gerenciar usuários do Amazon DocumentDB.)

Com o shell do `mongo`, você pode usar a seguinte consulta para listar todas as operações em execução em uma instância do Amazon DocumentDB.

```
db.adminCommand({currentOp: 1, $all: 1});
```

A consulta retorna a lista completa de todas as consultas do usuário e tarefas internas do sistema atualmente em operação na instância.

A saída dessa operação é semelhante ao seguinte (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
}
```

Estes são valores válidos para o campo `"desc"`:

- **`INTERNAL`** — tarefas internas do sistema, como a limpeza do cursor ou tarefas de limpeza obsoletas do usuário.
- **`TTLMonitor`**: thread de monitoramento Time to Live (TTL - vida útil). O status de execução é refletido no campo `"active"`.
- **`GARBAGE_COLLECTION`**: thread do coletor de resíduos interno.
- **`CONN`** — consulta do usuário.
- **`CURSOR`** — a operação é um cursor inativo esperando que o usuário chame o comando "getMore" para obter o próximo lote de resultados. Nesse estado, o cursor está consumindo memória, mas não está consumindo nenhuma computação.

A saída anterior também lista todas as consultas do usuário em execução no sistema. Cada consulta de usuário é executada no contexto de um banco de dados e uma coleção, e a união desses dois é chamada de namespace. O namespace de cada consulta do usuário está disponível no campo `"ns"`.

Às vezes, você precisa listar todas as consultas de usuário em execução em um determinado namespace. Portanto, a saída anterior deve ser filtrada no campo `"ns"`. Um exemplo de consulta para alcançar a saída para filtrar é o seguinte. A consulta lista todas as consultas de usuário em execução no momento no banco de dados `"db"` e na coleção `"test"` (ou seja, o namespace `"db.test"`).

```
db.adminCommand({aggregate: 1,
    pipeline: [{$currentOp: {allUsers: true, idleConnections: true}},
               {$match: {ns: {$eq: "db.test"}}}],
```

```
    cursor: {}
});
```

Como usuário principal do sistema, você pode ver as consultas de todos os usuários e também todas as tarefas internas do sistema. Todos os outros usuários só podem ver suas respectivas consultas.

Se o número total de consultas e tarefas internas do sistema exceder o tamanho do lote padrão do cursor, o shell do mongo gerará automaticamente um objeto iterador `'it'` para visualizar o resto dos resultados. Continue executando o comando `'it'` até que todos os resultados sejam esgotados.

Como posso saber quando uma consulta está progredindo?

As consultas do usuário podem ficar lentas devido a um plano de execução de consulta pouco satisfatório ou ficar bloqueadas devido à contenção de recursos. Depurar essas consultas é um processo de várias etapas que pode exigir que a mesma etapa seja executada várias vezes.

A primeira etapa da depuração é listar todas as consultas que estão bloqueada ou de longa execução. A consulta a seguir lista todas as consultas de usuário que estão em execução por mais de 10 segundos ou que estão aguardando 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 a consulta anterior periodicamente para determinar se a lista de consultas muda e identificar as consultas bloqueadas ou de longa execução.

Se o documento de saída da consulta em questão tiver um campo `WaitState`, isso indica que a contenção de recursos é o motivo pelo qual a consulta está lenta ou bloqueada. A contenção de recursos pode ser devido a I/O tarefas internas do sistema ou outras consultas do usuário.

A saída dessa operação é semelhante ao seguinte (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 pode ser um gargalo se muitas consultas em diferentes coleções estiverem sendo executadas simultaneamente na mesma instância ou se o tamanho da instância for muito pequeno para o conjunto de dados em que a consulta está sendo executada. Se as consultas forem somente de leitura, será possível atenuar a situação anterior separando as consultas de cada coleção em réplicas separadas. Para atualizações simultâneas em diferentes coleções ou quando o tamanho da instância for muito pequeno para o conjunto de dados, é possível mitigar aumentando a instância.

Se a contenção de recursos ocorrer devido a outra consulta do usuário, o campo `"blockedOn"` no documento de saída terá o `"opid"` da consulta que está afetando essa consulta. Com o `"opid"` siga a cadeia de campos `"WaitState"` e `"blockedOn"` de todas as consultas para encontrar a consulta à frente da cadeia.

Se a tarefa à frente da cadeia for uma tarefa interna, a única mitigação nesse caso seria encerrar a consulta e executá-la novamente depois de algum tempo.

Veja a seguir um exemplo de saída em que a consulta de busca é bloqueada em um bloqueio de coleção que pertence a outra tarefa.

```
{
    "inprog" : [
        {
            "client" : "...",
```

```
            "desc" : "Conn",
            "active" : true,
            "killPending" : false,
            "opid" : 75,
            "ns" : "...",
            "command" : {
                "find" : "...",
                "filter" : {

                }
            },
            "op" : "query",
            "$db" : "test",
            "secs_running" : 9,
            "microsecs_running" : NumberLong(9449440),
            "threadId" : 24773,
            "clientMetaData" : {
                "application" : {
                   "name" : "MongoDB Shell"
                },
                "driver" : {
                    ...
                },
                "os" : {
                    ...
                }
            },
            "WaitState" : "CollectionLock",
            "blockedOn" : "INTERNAL"
        },
        {
            "desc" : "INTERNAL"
        },
        {
            "client" : "...",
            ...
            "command" : {
                "currentOp" : 1
            },
            ...
        }
    ],
    "ok" : 1
```

```
}
```

Se "WaitState" tiver valores "Latch", "SystemLock", "BufferLock", "BackgroundActivity" ou "Other", a origem da contenção de recursos serão as tarefas internas do sistema. Se a situação continuar por um longo período, a única atenuação será encerrar a consulta e executá-la novamente mais tarde.

Como determino por que um sistema fica lento repentinamente?

Veja a seguir alguns motivos comuns para uma desaceleração do sistema:

- Contenção excessiva de recursos entre consultas simultâneas
- O número de consultas simultâneas ativas aumentando ao longo do tempo
- Tarefas internas do sistema, como "GARBAGE_COLLECTION"

Para monitorar o uso do sistema ao longo do tempo, execute a seguinte consulta "currentOp" periodicamente e gere a saída dos resultados em um armazenamento externo. A consulta conta o número de consultas e operações em cada namespace no sistema. É possível analisar os resultados do uso do sistema para compreender a carga no sistema e tomar a medida apropriada.

```
db.adminCommand({aggregate: 1,
                 pipeline: [{$currentOp: {allUsers: true, idleConnections: true}},
                            {$group: {_id: {desc: "$desc", ns: "$ns", WaitState:
 "$WaitState"}, count: {$sum: 1}}}],
                 cursor: {}
                });
```

Essa consulta retorna um agregado de todas as consultas em execução em cada namespace e em todas as tarefas internas do sistema, e o número exclusivo de estados de espera, se houver, por namespace.

A saída dessa operação é semelhante ao seguinte (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
}
```

Na saída anterior, duas consultas de usuário no namespace `"db.test"` estão bloqueadas no bloqueio da coleção: uma consulta no namespace `"admin.$cmd"` e uma tarefa interna `"TTLMonitor"`.

Se a saída indicar muitas consultas com estados de espera de bloqueio, consulte Como faço para localizar e encerrar consultas bloqueadas ou de longa execução?

Como determino a causa da alta utilização de CPU em uma ou mais instâncias de cluster?

As seções a seguir podem ajudar você a identificar a causa da alta utilização da CPU da instância. Os resultados podem variar dependendo da workload.

- Para determinar por que uma instância está sendo executada lentamente de forma repentina, consulte Como determino por que um sistema fica lento repentinamente?

- Para identificar e encerrar consultas de longa execução em uma instância específica, consulte Como faço para localizar e encerrar consultas bloqueadas ou de longa execução?
- Para entender se uma consulta está progredindo, consulte Como posso saber quando uma consulta está progredindo?
- Para determinar por que uma consulta leva muito tempo para ser executada, consulte Como posso ver um plano de consulta e otimizar uma consulta?
- Para rastrear consultas demoradas ao longo do tempo, consulte Definir o perfil das operações do Amazon DocumentDB.

Dependendo do motivo da alta utilização da CPU da instância, executar uma ou mais das ações a seguir pode ajudar.

- Se a instância primária apresentar alta utilização de CPU, mas as instâncias de réplica não, considere distribuir o tráfego de leitura entre réplicas por meio de configurações de preferência de leitura do cliente (por exemplo, `secondaryPreferred`). Para obter mais informações, consulte Conectar-se ao Amazon DocumentDB como um conjunto de réplicas.

  O uso de réplicas para leituras pode fazer melhor uso dos recursos do cluster, permitindo que a instância primária processe mais tráfego de gravação. As leituras de réplicas do são eventualmente consistentes.
- Se a alta utilização da CPU for resultado da workload de gravação, alterar o tamanho das instâncias do cluster para um tipo de instância maior aumentará o número de núcleos de CPU disponíveis para atender à workload. Para obter mais informações, consulte Gerenciar instâncias do Amazon DocumentDB e Especificações de classe de instância.
- Se todas as instâncias de cluster exibirem alta utilização de CPU e a workload estiver usando réplicas para leituras, adicionar mais réplicas ao cluster aumentará os recursos disponíveis para o tráfego de leitura. Para obter mais informações, consulte Adicionar uma instância do Amazon DocumentDB a um cluster.

Como posso determinar os cursores abertos em uma instância?

Quando estiver conectado a uma instância do Amazon DocumentDB, será possível usar o comando `db.runCommand("listCursors")` para listar os cursores abertos nessa instância. Há uma cota de até 4.560 cursores ativos abertos a qualquer momento em uma determinada instância do Amazon DocumentDB, dependendo do tipo de instância. Feche os cursores que não estão mais em uso

porque os cursores utilizam recursos em uma instância. Para obter mais informações, consulte Cotas do Amazon DocumentDB.

```
db.runCommand("listCursors")
```

Como determinar a versão atual do mecanismo do Amazon DocumentDB?

Para determinar a versão atual do mecanismo do Amazon DocumentDB, execute o seguinte comando.

```
db.runCommand({getEngineVersion: 1})
```

A saída dessa operação é semelhante ao seguinte (formato JSON).

```
{ "engineVersion" : "2.x.x", "ok" : 1 }
```

> **Note**
>
> A versão do mecanismo do Amazon DocumentDB 3.6 é 1.x.x, a versão do mecanismo do Amazon DocumentDB 4.0 é 2.x.x, a versão do mecanismo do Amazon DocumentDB 5.0 é 3.x.x e a versão do mecanismo do Amazon DocumentDB 8.0 é 4.x.x.

Como analiso o uso do índice e identifico índices não utilizados?

Para identificar os índices de uma determinada coleção, execute o seguinte comando:

```
db.collection.getIndexes()
```

Para analisar quantos índices estão sendo usados durante as operações realizadas nas coleções, os comandos `collStats` e `indexStats` podem ser usados. Para visualizar o número total de varreduras realizadas usando índices (varreduras de índices) em comparação com o número de varreduras realizadas sem um índice (varreduras de coleções), execute o seguinte comando:

```
db.collection.stats()
```

O resultado desse comando inclui os seguintes valores:

- **`idxScans`**: número de varreduras realizadas nessa coleção usando um índice.

- **`collScans`**: número de varreduras realizadas nessa coleção sem usar um índice. Essas digitalizações envolveriam a análise dos documentos da coleção, um de cada vez.
- **`lastReset`**: horário em que esses contadores foram redefinidos pela última vez. As estatísticas fornecidas por esse comando são redefinidas quando starting/stopping o cluster ou a instância é up/down escalada.

Um detalhamento de quanto cada índice é usado pode ser encontrado no resultado do comando a seguir. É uma prática recomendada identificar e remover regularmente índices não utilizados para melhorar o desempenho e reduzir custos, pois isso elimina a computação e o armazenamento desnecessários e o I/Os uso para manter os índices.

```
db.collection.aggregate([{$indexStats:{}}]).pretty()
```

O resultado desse comando fornece os seguintes valores para cada índice criado na coleção:

- **`ops`** - número de operações que usaram o índice. Se sua workload estiver sendo executada por um tempo suficientemente longo e você estiver confiante de que ela está em um estado estável, um valor zero de `ops` indicaria que o índice não é usado.
- **`numDocsRead`**: número de documentos lidos durante as operações usando esse índice.
- **`since`** - tempo desde que o Amazon DocumentDB começou a coletar estatísticas sobre o uso do índice, que geralmente é o valor desde a última reinicialização ou ação de manutenção do banco de dados.
- **`size`** - tamanho desse índice em bytes.

O exemplo a seguir é um exemplo de saída da execução do comando acima:

```
{
    "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 o tamanho geral do índice para uma coleção, execute o seguinte comando:

```
db.collection.stats()
```

Para descartar um índice não utilizado, execute o seguinte comando:

```
db.collection.dropIndex("indexName")
```

Como identifico índices ausentes?

É possível usar o profiler do Amazon DocumentDB para registrar consultas lentas em log. Uma consulta que aparece repetidamente no log de consulta lenta pode indicar que um índice adicional é necessário para melhorar o desempenho dessa consulta.

Você pode identificar oportunidades de índices úteis procurando consultas de longa duração que tenham um ou mais estágios que executem pelo menos um `COLLSCAN` estágio, o que significa que

o estágio de consulta precisa ler todos os documentos da coleção para fornecer uma resposta à consulta.

O exemplo a seguir mostra uma consulta em uma coleção de corridas de táxi que foi executada em uma coleção grande.

```
db.rides.count({"fare.totalAmount":{$gt:10.0}}))
```

Para executar este exemplo, a consulta tinha que realizar uma varredura na coleção (ou seja, ler todos os documentos da coleção), pois não há índice no campo `fare.totalAmount`. A saída do profiler do Amazon DocumentDB para essa consulta tem uma aparência semelhante à seguinte:

```
{
    ...
    "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 a consulta neste exemplo, convém criar um índice no `fare.totalAmount`, como mostrado abaixo.

```
db.rides.createIndex( {"fare.totalAmount": 1}, {background: true} )
```

> **Note**
>
> Os índices criados em primeiro plano (ou seja, se a opção `{background:true}` não foi fornecida ao criar o índice) usam um bloqueio de gravação exclusivo, o que impede que as aplicações gravem dados na coleção até que a compilação do índice seja concluída. Esteja

ciente desse possível impacto ao criar índices em clusters de produção. Ao criar índices, `{background:true}` defina.

Em geral, convém criar índices em campos com cardinalidade alta (por exemplo, um grande número de valores exclusivos). Criar um índice em um campo com baixa cardinalidade pode resultar em um índice grande que não é usado. O otimizador de consulta do Amazon DocumentDB considera o tamanho geral da coleção e a seletividade dos índices ao criar um plano de consulta. Às vezes, você verá o processador de consultas selecionar um `COLLSCAN` mesmo quando um índice estiver presente. Isso acontece quando o processador de consultas estima que a utilização do índice não produzirá uma vantagem de desempenho sobre a varredura de toda a coleção. Se você quiser forçar o processador de consultas a utilizar um índice específico, use o operador `hint()` como mostrado abaixo.

```
db.collection.find().hint("indexName")
```

Como faço para determinar a sobrecarga da coleção de banco de dados?

A sobrecarga da coleção ocorre quando seu tamanho se torna maior devido ao acúmulo de entradas de índice documentos inativos ou obsoletos ou à fragmentação nas páginas do banco de dados. O percentual relatado representa a quantidade de espaço de documentos que pode ser usada por futuros documentos. Essa sobrecarga consome espaço no cache e no armazenamento do buffer. Para remover o inchaço, as coleções devem ser recarregadas por meio dump/restore de um loop de migração e alternância durante uma janela de manutenção.

Example Exemplo

Execute o comando a seguir para determinar o armazenamento não utilizado da sua coleção:

```
db.runCommand({collStats:'coll'})
```

O resultado deve ser semelhante a este:

```
{
        "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)
}
```

Resumo de consultas úteis

As consultas a seguir podem ser úteis para monitorar o desempenho e a utilização de recursos no Amazon DocumentDB.

- Use o comando a seguir para visualizar estatísticas sobre uma coleção específica, incluindo contadores de operação, estatísticas de cache, estatísticas de acessos e estatísticas de tamanho:

```
db.collection.stats()
```

- Use o comando a seguir para visualizar estatísticas sobre cada índice criado em uma coleção, incluindo o tamanho do índice, estatísticas de cache específicas do índice e estatísticas de uso do índice:

```
db.collection.aggregate([{$indexStats:{}}]).pretty()
```

- Use a consulta a seguir para listar todas as atividades.

```
db.adminCommand({currentOp: 1, $all: 1});
```

- O código a seguir lista todas as consultas bloqueadas ou de longa execução.

```
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: {}
                });
```

- O código a seguir encerra uma consulta.

```
db.adminCommand({killOp: 1, op: <opid of running or blocked query>});
```

- Use o código a seguir para obter uma visualização agregada do estado do sistema.

```
db.adminCommand({aggregate: 1,
                 pipeline: [{$currentOp: {allUsers: true, idleConnections: true}},
                            {$group: {_id: {desc: "$desc", ns: "$ns", WaitState:
 "$WaitState"}, count: {$sum: 1}}}],
                 cursor: {}
                });
```

Coleta de resíduos no Amazon DocumentDB

O Amazon DocumentDB implementa uma arquitetura de banco de dados de controle de simultaneidade de várias versões (MVCC) que cria novas versões de entradas de documentos e índices para cada operação de atualização. Essa arquitetura permite o isolamento de transações, impedindo que as alterações de uma transação apareçam em outra.

Tópicos

- Entendendo a coleta de lixo no Amazon DocumentDB
- Processo de coleta de resíduos
- Arquitetura de armazenamento e armazenamento estendido
- Monitoramento da coleta de resíduos
- Exemplo de saída CollStats
- Perguntas frequentes

Entendendo a coleta de lixo no Amazon DocumentDB

A coleta de resíduos (GC) é um processo automatizado em segundo plano que mantém a performance e a disponibilidade ideais do sistema no Amazon DocumentDB. Como muitos bancos de dados modernos, a arquitetura MVCC do Amazon DocumentDB cria novas versões de documentos e índices a cada atualização. Cada operação de gravação consome um ID MVCC exclusivo de um contador finito. Esses IDs identificam a qual transação uma versão do documento pertence e se ela foi confirmada ou revertida. Com o tempo, essas versões antigas e seus IDs MVCC se acumulam, exigindo limpeza para evitar a degradação do desempenho.

Funções da coleta de resíduos

O coletor de resíduos tem três funções essenciais:

- Recuperar espaço de armazenamento: ele remove versões obsoletas de documentos e índices que não sejam mais necessárias para consultas ativas, liberando espaço para futuras operações de gravação.
- Evitar o estouro de IDs do MVCC: ele evita o estouro de IDs do MVCC gerenciando o contador finito de IDs do MVCC. Sem esse gerenciamento, o contador acabaria atingindo seu limite, forçando o banco de dados a entrar em um modo temporário somente para leitura até que os IDs fossem reciclados.

- Manter a performance da consulta: ele mantém a performance ideal da consulta eliminando versões inativas de documentos que, de outra forma, se acumulariam e retardariam o processamento de consultas.

Processo de coleta de resíduos

O processo de GC opera por coleção e pode ter vários processos sendo executados simultaneamente em coleções diferentes. O processo consiste em quatro fases sequenciais:

1. Identificação: o sistema identifica versões de documentos e índices que não são mais referenciadas por transações ou consultas ativas.
2. Carregamento na memória: os documentos antigos e as entradas de índice são carregados na memória se ainda não estiverem presentes.
3. Exclusão: as versões obsoletas são excluídas permanentemente para recuperar espaço de armazenamento.
4. Reciclagem de IDs do MVCC: o sistema recicla IDs do MVCC de versões excluídas para novas operações.

Quando a coleta de resíduos conclui o processamento das versões antigas do documento, ela remove os IDs do MVCC mais antigos do sistema. Essa limpeza é crucial para evitar o estouro de IDs do MVCC reciclando e disponibilizando-os para novas operações de gravação em todo o cluster. Sem esse processo de reciclagem, o sistema acabaria por esgotar seu contador finito de IDs do MVCC e entrar em um estado somente de leitura.

Programação da coleta de resíduos

A coleta de resíduos é executada automaticamente em segundo plano, em intervalos periódicos. O tempo e a frequência se ajustam dinamicamente com base na carga do sistema, nos recursos disponíveis, no volume de gravação e nos níveis de consumo de IDs do MVCC. Durante a alta atividade de gravação, o processo de GC é executado com mais frequência para gerenciar o aumento do número de versões do documento.

Arquitetura de armazenamento e armazenamento estendido

O Amazon DocumentDB usa uma arquitetura de armazenamento sofisticada que separa o armazenamento de documentos em dois segmentos distintos:

Segmento básico de armazenamento

O segmento de armazenamento básico contém os principais dados e metadados do documento. Esse segmento armazena:

- Conteúdo do documento que se encaixa no tamanho de página padrão (8 KB).
- Metadados do documento e informações da estrutura.
- Índices primários e suas entradas.
- Collection-level estatísticas e configuração.

Segmento de armazenamento estendido

O segmento de armazenamento estendido utiliza um grande armazenamento especializado de objetos de documentos, projetado para lidar com documentos que excedem o tamanho padrão da página de armazenamento. Esse segmento fornece:

- Processamento eficiente de documentos grandes — Documentos maiores que o limite básico de armazenamento são automaticamente movidos para o segmento de armazenamento estendido.
- Layout de armazenamento otimizado — O segmento usa um formato de armazenamento diferente, otimizado para objetos grandes, reduzindo a fragmentação e melhorando os padrões de acesso.
- Coleta de lixo independente — O segmento de armazenamento estendido tem seu próprio processo de coleta de lixo que pode ser executado independentemente da limpeza do armazenamento básico.
- Acesso transparente — Os aplicativos acessam documentos grandes sem precisar saber qual segmento de armazenamento contém os dados.

O segmento de armazenamento estendido é particularmente benéfico para:

- Coleções com documentos contendo grandes matrizes incorporadas.
- Documentos com extensas estruturas aninhadas.
- Coleções que armazenam dados binários ou grandes campos de texto.
- Aplicativos com tamanhos variados de documentos em que alguns documentos excedem significativamente o tamanho médio.

Monitoramento da coleta de resíduos

Métricas em nível de cluster

`AvailableMVCCIds`

- Localização — Amazon CloudWatch
- Descrição — Um contador que mostra o número de operações de gravação restantes disponíveis a partir de um limite máximo de 1,8 bilhão. Quando esse contador chega a zero, seu cluster entra no modo somente de leitura até que os IDs sejam recuperados e reciclados. O contador diminui a cada operação de gravação e aumenta à medida que a coleta de resíduos recicla IDs de MVCC antigos.
- Recomendação — Defina um alarme quando o valor cair abaixo de 900 milhões no Amazon DocumentDB 5.0 e posterior, ou abaixo de 1,3 bilhão no Amazon DocumentDB 3.6 e 4.0. Esse aviso antecipado permite que você execute as etapas recomendadas discutidas posteriormente.

`LongestActiveGCRuntime`

- Localização — Amazon CloudWatch
- Descrição: duração em segundos do processo de coleta de resíduos ativo mais longo. Atualiza a cada minuto e rastreia somente as operações ativas, excluindo os processos concluídos dentro da janela de um minuto.
- Recomendação — Compare com dados `gcRuntimeStats` históricos para identificar o comportamento anormal da coleta de lixo, como tempos de execução estendidos durante exclusões em massa.

Métricas de nível de coleção

`MVCCIDStats: MVCCIdScale`

- Localização: comando collStats do banco de dados
- Descrição: mede a idade do ID do MVCC em uma escala de 0 a 1, em que 1 indica a idade máxima antes de um cluster entrar em um estado somente de leitura. Use essa métrica junto com `AvailableMVCCIds` para identificar coleções que contenham os IDs do MVCC mais antigos que estejam envelhecendo o cluster.
- Recomendação: mantenha valores abaixo de 0,3 para cada coleção.

`gcRuntimeStats`

- Localização: comando collStats do banco de dados
- Descrição: fornece um histórico de dois meses das métricas de coleta de resíduos, incluindo o total de execuções, a duração média e a duração máxima. Inclui apenas operações de coleta de resíduos com duração superior a cinco minutos para garantir estatísticas significativas.

`storageSizeStats`

- Localização: comando collStats do banco de dados
- Descrição — fornece uma análise detalhada da utilização do armazenamento em diferentes segmentos de armazenamento:
  - `storageSegmentBase`— Armazenamento usado pelo segmento de armazenamento básico para documentos padrão
  - `storageSegmentExtended`— Armazenamento usado pelo segmento de armazenamento estendido para documentos grandes
- Uso — ajuda a identificar coleções com armazenamento significativo de documentos grandes e a entender os padrões de distribuição de armazenamento.

`unusedStorageSize` (nível de coleção)

- Localização: comando collStats do banco de dados
- Descrição: estima o espaço de armazenamento não utilizado em uma coleção com base nas estatísticas amostradas. Ela inclui o espaço de documentos excluídos e segmentos vazios. A métrica fornece totais combinados e detalhamentos por segmento:
  - Combinado `unusedBytes` e `unusedPercent` em todos os segmentos de armazenamento
  - `storageSegmentBase`— Espaço não utilizado especificamente no segmento de armazenamento básico
  - `storageSegmentExtended`— Espaço não utilizado especificamente no segmento de armazenamento estendido

`documentFragmentStats`

- Localização: comando collStats do banco de dados

- Descrição — Fornece informações detalhadas sobre fragmentos de documentos e dados inativos nas coleções. Fragmentos de documentos representam as unidades de armazenamento internas usadas pelo mecanismo de banco de dados, e fragmentos mortos indicam dados que não estão mais acessíveis, mas ainda não foram recuperados. Essa métrica inclui:
  - `totalDocFragmentsCount`— Número total de fragmentos de documentos na coleção
  - `deadDocFragmentsCount`— Número de fragmentos contendo dados mortos (inacessíveis)
  - `deadDocFragmentsPercent`— Porcentagem de fragmentos que contêm dados inativos
  - `deadDocFragmentBytes`— Bytes estimados consumidos por fragmentos de documentos mortos
  - Per-segment detalhamento para `storageSegmentBase` e `storageSegmentExtended`
- Uso — Monitore essa métrica para entender a eficácia da coleta de lixo e identificar coletas que podem se beneficiar das operações de manutenção. Altas porcentagens de fragmentos mortos indicam que a coleta de lixo pode estar ficando para trás ou que a coleta se beneficiaria da otimização.

Métricas de nível de índice

`unusedStorageSize` (nível de índice)

- Localização: comando indexStats do banco de dados
- Descrição: estima o espaço de armazenamento não utilizado em um índice com base nas estatísticas amostradas. Ela inclui o espaço de entradas de índice obsoletas e segmentos vazios.
- Recomendação: use o comando `reIndex` para reconstruir índices sem tempo de inatividade e recuperar espaço não utilizado. Para obter mais detalhes, consulte Gerenciamento de índices.

Exemplo de saída CollStats

O exemplo a seguir mostra uma `collStats` saída típica com métricas de coleta e armazenamento de lixo:

```
{
    "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)
}
```

Perguntas frequentes

Como faço para identificar se a coleta de resíduos não está funcionando de forma eficiente?

Monitore estes sinais de alerta que indicam coleta de resíduos ineficiente:

- Inchaço excessivo da coleção — Aumento constante das `unusedStorageSize` métricas durante gravações pesadas ou exclusões em massa, especialmente com índices grandes.
- Alta porcentagem de fragmentos mortos — `documentFragmentStats` mostrando `deadDocFragmentsPercent` valores consistentemente altos (acima de 10-15%).
- Latência de consulta reduzida — Maior latência de consulta devido ao acúmulo de documentos inativos.

- Duração estendida do GC — As operações de coleta de lixo demoram mais do que as médias históricas em. `gcRuntimeStats`
- Processamento elevado de GC — Alto `LongestActiveGCRuntime` indica que o coletor de lixo não consegue atender às demandas do sistema.

A coleta de resíduos afeta a performance do meu banco de dados?

Em condições normais, a coleta de resíduos tem um impacto mínimo na performance. No entanto, quando a coleta de resíduos atrasa, pode ser que você experimente:

- Aumento dos custos de armazenamento de documentos inativos acumulados.
- Desempenho de consulta mais lento devido a entradas de índice obsoletas.
- Modo somente leitura temporário se os IDs do MVCC estiverem esgotados.
- Maior uso de recursos durante execuções intensivas de coleta, especialmente em instâncias menores.
- Eficiência reduzida nas operações do segmento de armazenamento estendido para documentos grandes.

Posso acionar manualmente a coleta de resíduos?

Não, a coleta de resíduos no Amazon DocumentDB não pode ser acionada manualmente. O sistema gerencia a coleta de resíduos automaticamente como parte de suas operações internas de manutenção.

Quais alarmes devo definir como prática recomendada operacional?

Configure o monitoramento nos níveis de cluster e de coleção para garantir o desempenho ideal do seu sistema Amazon DocumentDB.

Para monitoramento em nível de cluster, comece criando um CloudWatch alarme da Amazon para a `AvailableMVCCIds` métrica com um limite de 900 milhões no Amazon DocumentDB 5.0 e versões posteriores, ou 1,3 bilhão no Amazon DocumentDB 3.6 e 4.0. Isso proporciona tempo suficiente para uma ação antes que a métrica chegue a zero, momento em que seu cluster entraria no modo somente de leitura. Lembre-se de que essa métrica pode flutuar com base em seus padrões de uso específicos. Alguns clientes veem que ela cai abaixo do limite e depois se recupera acima de 1,5 bilhão quando a coleta de lixo termina seu trabalho.

Também é importante monitorar a `LongestActiveGCRuntime` métrica por meio da Amazon CloudWatch. Essa métrica, junto com `gcRuntimeStats`, ajuda você a entender a eficiência com que a coleta de resíduos está funcionando em todo o sistema.

Para monitoramento em nível de coleção, concentre-se nessas métricas principais:

- `MVCCIdScale`— Observe os valores crescentes que sugerem que os IDs do MVCC estão envelhecendo e podem precisar de atenção.
- `gcRuntimeStats`— Identifique processos de coleta de lixo que demoram muito ou se estendem por vários dias.
- `documentFragmentStats`— `deadDocFragmentsPercent` Valores do monitor - porcentagens consistentemente altas (acima de 10 a 15%) podem indicar que a coleta de lixo está atrasada.
- `storageSizeStats`e `unusedStorageSize` — acompanhe os padrões de utilização do armazenamento e identifique coleções com espaço significativo não utilizado em qualquer segmento de armazenamento.

Coleções com operações de gravação frequentes precisam de atenção extra, pois geram mais trabalho para o coletor de resíduos. Verifique essas métricas com mais frequência para coleções com intensa atividade de gravação para garantir que a coleta de lixo acompanhe sua carga de trabalho.

Observe que essas recomendações de monitoramento servem como ponto de partida. À medida que você se familiariza com o comportamento do seu sistema, talvez queira ajustar esses limites para melhor corresponder aos seus padrões e requisitos de uso específicos.

O que devo fazer se meus **`MVCCIDs disponíveis`** caírem abaixo do limite de alarme?

Se sua `AvailableMVCCIds` métrica cair abaixo do limite de alarme, tome medidas imediatas para impedir que seu cluster entre no modo somente leitura. Primeiro, aumente o tamanho da sua instância para fornecer ao coletor de lixo mais recursos de computação. Isso permite que sua aplicação continue com as operações normais e, ao mesmo tempo, forneça ao coletor de resíduos a energia adicional necessária para se recuperar.

Se a expansão por si só não melhorar a situação, considere reduzir suas operações de gravação. Use a métrica `MVCCIdScale` para identificar quais coleções específicas contêm IDs do MVCC mais antigos que precisem de atenção. Além disso, monitore `documentFragmentStats` para identificar

coleções com altas porcentagens de fragmentos mortos que possam estar contribuindo para a ineficiência da coleta de lixo.

Depois de identificar essas coleções, talvez seja necessário reduzir temporariamente as operações de gravação nelas para permitir que a coleta de lixo se atualize. Durante o período de recuperação, monitore de perto a `AvailableMVCCIds` métrica para garantir que suas ações tenham o efeito desejado. Seu cluster será considerado íntegro quando o valor `AvailableMVCCIds` retornar para 1,5 bilhão ou mais.

Lembre-se de que essas etapas são medidas preventivas para ajudar seu sistema a se recuperar antes de atingir um estado crítico. Quanto mais cedo você agir depois de ver a métrica cair abaixo do limite, maior a probabilidade de evitar qualquer impacto em suas operações de gravação.

As operações em lote afetam a taxa de consumo do ID MVCC?

Sim. Como um lote executado em uma única transação atômica consome somente um ID MVCC, o agrupamento de gravações em lotes reduz a taxa na qual os IDs MVCC são consumidos. Como melhor prática, use `insertMany` e `updateMany` sempre que possível para minimizar o consumo de ID MVCC e reduzir a carga na coleta de lixo.

Por que os **`MVCCIDs disponíveis não`** se recuperam até atingir um determinado limite?

Espera-se temporariamente um declínio constante `AvailableMVCCIds` sem recuperação para uma carga de trabalho dominante na gravação que ainda não consumiu IDs MVCC suficientes para acionar o ciclo de recuperação da coleta de lixo. A coleta de lixo é acionada quando qualquer uma das seguintes condições é atendida: uma determinada parte dos documentos de uma coleção se torna obsoleta, uma certa parte dos novos documentos é inserida ou fica abaixo do limite `AvailableMVCCIds` de recuperação.

Até que `AvailableMVCCIds` fique abaixo do limite de recuperação, a recuperação forçada não ocorre. Quando isso acontecer, a recuperação será acionada e `AvailableMVCCIds` se recuperará. No entanto, transações de longa duração e cursores abertos podem definir o horizonte de recuperação e impedir a recuperação mesmo após o limite ser atingido.

Códigos de erro

Quando um comando falha, você recebe um erro do Amazon DocumentDB (com compatibilidade com o MongoDB) que contém um erro numérico `code` e uma mensagem descritiva (). `errmsg` Use

a tabela a seguir para identificar os códigos de erro que o Amazon DocumentDB 8.0 pode retornar ao seu aplicativo cliente. Quando um código de erro corresponde a uma única condição, a descrição corresponde ao texto da mensagem retornado pelo servidor. Os códigos que abrangem várias condições têm uma descrição geral.

> **Important**
>
> Não crie uma lógica de aplicativo que dependa de códigos de erro numéricos específicos, pois o Amazon DocumentDB pode alterar os códigos de erro entre as versões principais.

Código de erro	Description
1	Item não encontrado
2	Um argumento de comando, opção ou valor de campo é inválido ou não é suportado.
4	Falta um campo obrigatório ou um valor de configuração não está formatado corretamente.
5	Uma definição de visualização faz referência a si mesma, criando um ciclo, ou ocorreu um erro no sistema de arquivos.
9	Um comando, campo, dica ou token de currículo não pôde ser analisado ou tem um formato inválido.
10	A coleção alvo não foi encontrada ou nenhuma coleção foi especificada.
11	Não foi possível encontrar um usuário referenciado ou a coleção de destino já existe.
13	Você não está autorizado a executar esse comando ou sua autenticação é inválida.
14	Um campo ou valor tem o tipo de dados errado.
15	Um valor ultrapassou sua faixa permitida durante a avaliação.

Código de erro	Description
16	O lote de gravação está vazio, malformado ou excede a contagem de operações permitida.
17	Nem todos os objetos têm o mesmo tipo
18	A autenticação falhou porque as credenciais são inválidas ou o usuário não existe.
19	Operação primitiva de processamento de dados não encontrada ou tipo primitivo não encontrado
20	A operação solicitada não é permitida nesse contexto.
26	O namespace ou coleção de destino não existe.
27	O índice solicitado não existe ou nenhum índice adequado está disponível.
37	Um argumento do pipeline ou do estágio de agregação está ausente, vazio ou inválido.
38	Nome do banco de dados não encontrado
39	A operação foi cancelada porque excedeu o MaxTimeMS, foi executada por muito tempo ou excedeu a memória disponível.
40	Uma atualização entra em conflito com outro campo ou as gravações são limitadas ou desativadas pela coleta de lixo.
41	JavaScript erro
43	O cursor especificado não existe mais ou expirou.
44	Índice de matriz fora dos limites
48	Já existe uma coleção, visualização ou namespace com esse nome.
51	Operador de atualização ilegal

Código de erro	Description
52	Um nome de campo usa um prefixo '$' reservado que não é válido para armazenamento.
53	O campo _id tem um tipo inválido ou não suportado, como uma matriz ou regex.
54	não é possível inferir campos de consulta a serem definidos
55	Transação já confirmada
56	O nome do campo na solicitação está vazio, o que não é permitido.
57	Transação invalidada por outra transação
58	O valor do campo é muito longo
59	O comando solicitado é desconhecido ou não é suportado.
61	Coleção inválida
62	Não é possível fazer alterações na coleção
65	erros de operação em lote ocorreram
66	Uma atualização tentou modificar o campo _id imutável.
67	O índice não pôde ser criado devido a uma especificação inválida ou sem suporte.
68	Já existe um índice com propriedades conflitantes.
70	o campo usado com $snippet não está configurado como coluna de trecho
71	O índice já existe com opções diferentes
72	Uma ou mais opções especificadas para o comando ou índice são inválidas.
73	O namespace, banco de dados ou nome da coleção é inválido ou não é permitido para essa operação.

Código de erro	Description
74	A coleção não tem um índice de texto definido.
75	Um índice de texto é definido para essa coleção, mas não para esse campo.
77	Não é possível usar instantâneo com classificação
78	Documento não encontrado
80	A consulta excede o tamanho máximo permitido.
82	O período de tempo tem formatação inválida. <number>Formato válido: '1 m' ou '3 y' '//' h/d w/s m/y
85	Já existe um índice com esse nome, mas com opções diferentes.
86	O índice solicitado está em conflito com um índice existente que tem o mesmo nome.
87	A ordem da chave $sort deve ser 1 (para ascendente) ou -1 (para decrescente)
92	Não há espaço suficiente para operação
93	O usuário já existe
94	O privilégio de coleta, cluster ou recurso de destino é inválido ou não está disponível.
95	O banco de dados não existe
96	O parâmetro 'WriteConcern' é inválido
97	A senha não pode estar vazia
99	Função ou privilégio indefinido
102	Não é possível adicionar uma função que já esteja incorporada
103	Não foi possível extrair uma função válida da coleção.

Código de erro	Description
105	O usuário é inválido ou a biblioteca de autorização não pôde ser inicializada.
107	O banco de dados não pode receber funções de outro banco de dados
111	_id deve ser especificado exatamente ao fornecer o documento de atualização
112	Conflito de escrita
113	A função especificada não existe.
114	A função já existe
115	AnyResource deve ser verdadeiro quando especificado
116	os recursos devem ter banco de dados e conjunto de coleção, conjunto de clusters ou conjunto AnyResource
117	A operação entra em conflito com outra operação em andamento, como uma transação ativa.
120	'ShowPrivileges' não pode ser especificado, pois estamos mostrando todos os usuários
121	Falha na validação do documento
122	Erro de atualização
124	Erro ao processar o ID do esquema
125	O comando falhou, geralmente devido a uma referência ambígua ou à falta de uma sessão.
126	O banco de dados já existe
127	O nome do banco de dados já existe em um nome de caso diferente
128	Muitos campos

Código de erro	Description
129	Aguardando a entrega do banco de dados
130	Valor incorreto do parâmetro
136	A posição do cursor foi perdida ou um recurso de fluxo de alteração necessário não está ativado.
139	Falha no intérprete JS
145	Alias não definido na consulta
147	não domina
148	Recursos insuficientes para executar a consulta
149	Alias já definido na consulta
154	Um argumento tem um valor ilegal ou um tipo incorreto (não numérico).
155	Valor NULL encontrado
156	Valor INDEFINIDO encontrado
157	Valor NAN encontrado
158	Coleção já em uso para outra finalidade
159	Um valor de duração estourou ou uma operação de rede falhou.
161	A versão de agrupamento solicitada é inválida ou não está disponível.
163	Um campo ou nome não é válido para armazenamento.
164	Um operador de consulta foi aninhado de uma forma que não é permitida.
165	Uma visualização excede a profundidade máxima de aninhamento de 20 ou faz referência a si mesma em um ciclo.

Código de erro	Description
166	A operação não é suportada em uma visualização ou muitos elementos posicionais foram usados.
167	A operação ou estágio solicitado não é suportado em uma visualização.
168	Uma expressão ou operador de pipeline não é reconhecido ou tem um argumento inválido.
169	BadValue $maxTimeMS tem valor não integral
170	BadValue $maxTimeMS está fora do alcance
171	Um índice não pode ser criado em vários campos de matriz.
173	O valor do salto deve ser positivo
174	Espera-se que a verbosidade do log seja um número
181	O padrão de chave corresponde a vários índices; em vez disso, especifique o índice pelo nome.
197	A especificação do índice contém um campo inválido ou não suportado.
207	uuid deve ser um campo binário de 16 bytes com o subtipo UUID (4)
224	Esse recurso de consulta não é permitido no contexto atual.
225	Uma transação mais recente foi iniciada na sessão, portanto, essa mais antiga não pode continuar.
241	Um valor não pôde ser convertido para o tipo solicitado em $convert.
250	Exibição inválida
251	O número da transação especificado não corresponde a nenhuma transação ativa.
252	Limite de recursão atingido

Código de erro	Description
253	Não foi possível inserir a entrada na exibição
254	Não foi possível excluir a entrada da exibição
256	Documento filtrado pela cláusula $having
257	O tamanho total de todas as operações na transação excede o limite permitido.
263	Essa operação não pode ser executada dentro de uma transação com vários documentos.
291	não foi possível encontrar o índice para a consulta $geoNear
300	A autenticação falhou devido a um aperto de mão do SASL ou a um erro de sessão.
301	Não há suporte para um mecanismo de autenticação ou operação solicitada.
302	A autenticação precisa de dados adicionais para concluir a troca de SASL.
303	A configuração de estágio, opção ou função solicitada não é suportada.
304	O estágio de agregação solicitado não é suportado.
305	O operador de projeção de agregação não é suportado.
306	O operador de acumulador de grupo de agregação especificado não é suportado.
502	A coluna já existe
505	Inserção não permitida aqui
601	As coordenadas estão fora dos limites
602	Erro de índice
700	Alterações nesse parâmetro não são permitidas no modo ao vivo
701	Valor inválido para esse parâmetro

Código de erro	Description
703	Transição de estado ilegal para este parâmetro
999	ID de transação incorreta
4570	Parâmetro $lookup esperando o valor da string
4572	$lookup não tem campos obrigatórios
10065	Um parâmetro que deve ser um objeto (como vars ou $rand) recebeu um valor que não é objeto.
10107	não domina
11000	Uma chave duplicada viola uma restrição de índice exclusiva.
11601	A operação foi interrompida.
11603	Solicitação do cliente: sintaxe corrompida ou ilegal
13103	muitas chaves compostas
13297	Já existe um banco de dados com o mesmo nome, mas difere apenas em letras maiúsculas e minúsculas.
13388	Nome artístico do pipeline não reconhecido: $collStats
13435	não master e slaveOK=false
14031	Espaço em disco insuficiente
15888	deve passar o nome da coleção para criar
15947	os campos de um grupo devem ser especificados em um objeto
15950	O nome do campo agregado do grupo não pode ser um operador
15952	operador $group desconhecido

Código de erro	Description
15955	Erro de sintaxe no operador de agregação do grupo: o campo _id deve ser especificado
15956	O argumento $skip é negativo; deve ser um número não negativo.
15957	o limite deve ser especificado como um número
15958	O valor $limit deve ser um número positivo.
15959	o filtro de correspondência deve ser uma expressão em um objeto
15969	Erro de sintaxe: o valor do operador $project deve ser um documento.
15972	O argumento para $skip deve ser um número
15973	a especificação da chave $sort deve ser um objeto
15974	A ordem da chave $sort deve ser especificada usando um número
15975	A ordem da chave $sort deve ser 1 (para ascendente) ou -1 (para decrescente)
15976	O estágio $sort deve ter pelo menos uma chave de classificação
15981	campo de consulta não é uma string
15983	Um objeto representando uma expressão deve ter exatamente um campo.
15992	campo não permitido tipo NULL na expressão do objeto
15998	Os nomes dos caminhos dos campos não podem ser cadeias de caracteres vazias.
16006	Um valor não pôde ser convertido em Data porque não é um tipo de data válido.
16007	Um valor não pode ser convertido em uma string porque uma string era esperada.
16020	Um operador de expressão foi chamado com o número errado de argumentos.

Código de erro	Description
16034	O índice inicial $substrBytes deve ser do tipo numérico.
16035	O argumento de comprimento $substrBytes deve ser do tipo numérico.
16395	$where não é permitido dentro de uma expressão de agregação $match
16400	não é possível adicionar uma expressão para um campo porque já existe uma expressão para esse campo ou um de seus subcampos.
16401	não é possível adicionar uma expressão para um subcampo porque já existe uma expressão que se aplica a todo o campo
16404	As expressões $ não são permitidas neste contexto
16406	Não é possível excluir campos diferentes de _id
16410	O nome ou caminho de um campo $project começa ilegalmente com ou contém '$'.
16411	Os nomes dos campos não podem conter caracteres nulos
16419	o caminho do campo não deve conter caracteres nulos incorporados
16424	$near não é permitido dentro de uma expressão de agregação $match
16426	$nearsphere/$near não é permitido dentro de uma expressão de agregação $match
16435	O componente do pipeline deve conter um elemento
16493	Tentei criar uma string com mais de 16 MB
16554	$add só oferece suporte a tipos numéricos ou de data.
16605	$geoNear requer um campo 'near' válido e opções geoNear digitadas corretame nte.
16606	$geoNear requer uma opção 'distanceField' como uma string

Código de erro	Description
16607	$geoNear exige que a opção fornecida seja especificada como uma string.
16608	Divida por zero
16610	$mod recebeu um divisor de zero.
16611	$mod só suporta tipos numéricos para ambos os operandos.
16612	somente uma data permitida em uma expressão $add
16613	A operação tentou subtrair um valor não numérico e sem data de uma data.
16702	$concat só suporta argumentos de string.
16730	o texto IndexVersion está fora do alcance
16755	não é possível indexar geometria com ordem de enrolamento estrita
16800	não pode ter 2 campos geográficos
16801	2d deve ser o primeiro no índice
16804	objeto de localização esperado, matriz de localização não está no formato correto
16810	$mod requer uma matriz de exatamente dois valores como parâmetro.
16837	Um caminho de atualização não pode modificar o mesmo campo mais de uma vez.
16868	Nome de variável inválido
16869	Nome de variável inválido
16870	Nome de variável inválido
16872	Um caminho de campo está vazio ou somente '$' é usado como um caminho de campo em $project.

Código de erro	Description
16873	Fieldpath não começa com $
16874	$let suporta apenas um objeto como argumento
16875	$let recebeu um parâmetro não reconhecido.
16876	Falta o parâmetro 'vars' para $let
16877	Falta o parâmetro 'in' para $let
16878	$map suporta apenas um objeto como argumento
16879	$map recebe um parâmetro não reconhecido.
16880	$map não tem seu parâmetro de 'entrada' obrigatório.
16881	Campo obrigatório ausente
16882	Falta o parâmetro 'in' para $map
16883	A entrada para $map deve ser uma matriz.
16898	$minDistance não deve ser negativo
16899	$maxDistance deve ser um número
16900	$maxDistance não deve ser negativo
16945	Limite de memória excedido para $group ou $lookup. O passe DiskUse:true permite a inscrição.
16949	Opção de agregação inválida; allow DiskUse deve ser um valor booleano.
16990	$out requer uma string ou argumento de objeto nomeando o destino de saída.
16996	falha na inserção de $out
16997	$out falhou porque os índices da coleção de destino foram alterados durante o processamento.

Código de erro	Description
17040	O argumento ElementsTrue de $all deve ser uma matriz.
17041	O argumento ElementTrue de $any deve ser uma matriz.
17042	Ambos os operandos de $set IsSubset devem ser matrizes.
17043	Todos os operandos de $setUnion devem ser matrizes.
17044	Um operando de $setEquals não é uma matriz.
17045	$setEquals requer pelo menos dois argumentos.
17046	Ambos os operandos de $set IsSubset devem ser matrizes.
17047	Todos os operandos de $setIntersection devem ser matrizes.
17048	Ambos os operandos de $setDifference devem ser matrizes.
17049	Ambos os operandos de $setDifference devem ser matrizes.
17053	A expressão de $redact não deve retornar nada além das variáveis $$KEEP, $ $DESCEND e $$PRUNE
17080	Falta o parâmetro 'if' para cond
17081	Falta o parâmetro 'then' para cond
17082	Falta o parâmetro 'else' para cond
17083	Parâmetro não reconhecido para $cond
17124	O argumento para $size deve ser uma matriz.
17275	Não é possível redefinir ROOT
17276	Variável não encontrada
17280	O valor da chave do índice é muito grande para ser indexado.

Código de erro	Description
17287	$merge não pode modificar a mesma linha de destino mais de uma vez.
17293	opção de índice de texto 'textoIndexVersion' deve ser um número
17302	Esse valor do argumento deve ser maior ou igual a zero.
17304	$geoNear requer um ponto 'próximo' válido e argumentos de consulta geográfica.
17308	$meta recebe um argumento não suportado.
17310	Ambos os operandos de $set IsSubset devem ser matrizes.
17313	$match com $text só é permitido como o primeiro estágio do pipeline.
17320	CreateCollection não pode ser executado em um namespace contendo um nome reservado.
17385	Não é possível sair para a coleção especial
17390	$group não suporta expressões de estilo de inclusão
17419	O documento resultante após a atualização excede o tamanho máximo permitido .
17420	O documento a ser inserido é maior que 16777216
18533	O campo de formato deve ser uma string
18534	Campo não reconhecido até $date ToString
18535	A string de formato de data é inválida devido a um sinal de porcentagem à direita.
18536	Formato $date ToString e hora inválido
18537	$date ToString não pode formatar um componente de data fora do intervalo 0-9999.

Código de erro	Description
18627	Falta o parâmetro 'format' para $date ToString
18628	Falta o parâmetro 'date' para $date ToString
18629	tipo de argumento incorreto
25278	$geoNear requer um argumento 'DistanceField'
26823	Um ID de operação não pode ser representado nos bits disponíveis.
28539	O primeiro argumento para o comando distinto deve ser uma string
28646	$Filter espera um objeto como parâmetro
28647	Campo inesperado para $filter
28648	$Filter requer um campo de 'entrada'
28649	$Filter requer um campo 'as'
28650	$Filter requer um campo 'cond'
28651	Entrada de matriz necessária
28656	$substrBytes: intervalo inválido, o índice inicial é um UTF-8 byte de continuação.
28657	$substrBytes: intervalo inválido, o índice final está no meio de um caractere. UTF-8
28664	$concatArrays suporta apenas entradas de matriz.
28667	Uma expressão como $slice, $range ou $index OfArray recebeu o número errado de argumentos.
28680	$abs é aplicado a um valor, como o mínimo longo, que não pode ser negado.
28689	Um argumento de operador deve ser uma matriz, mas um tipo diferente foi fornecido.

Código de erro	Description
28690	O segundo argumento para a matriz ElemAt deve ser numérico
28691	O segundo argumento para a matriz ElemAt deve ser representado por um inteiro de 32 bits
28714	O argumento de $sqrt deve ser maior ou igual a 0
28724	O primeiro argumento para $slice deve ser uma matriz.
28725	O segundo argumento para $slice deve ser um valor numérico.
28726	O segundo argumento para $slice deve caber em um inteiro de 32 bits.
28727	O terceiro argumento para $slice deve ser um número.
28728	O terceiro argumento para $slice deve ser um inteiro de 32 bits.
28729	O terceiro argumento de $slice deve ser um inteiro positivo de 32 bits.
28745	a especificação do estágio $sample deve ser um objeto
28746	o argumento de tamanho para $sample deve ser um número
28747	o argumento de tamanho para $sample não deve ser negativo
28748	$sample recebeu uma opção não reconhecida.
28749	O estágio $sample deve especificar um tamanho
28756	O argumento de $log deve ser numérico.
28757	$log exige que seu argumento base seja um valor numérico.
28758	O argumento para $log deve ser um número positivo.
28759	A base de $log deve ser um número positivo diferente de 1.
28761	O argumento de $log10 deve ser um número positivo.

Código de erro	Description
28762	A base de $pow deve ser um valor numérico.
28763	O expoente de $pow deve ser numérico.
28764	$pow não pode ter uma base de 0 e um expoente negativo
28765	$ceil, $exp, $floor, $ln ou $log10 recebem um tipo não numérico.
28766	$ln requer um argumento numérico positivo.
28812	O caminho do campo está vazio
28818	Fieldpath não começa com um '$'
31022	Esse operador exige que um parâmetro de 'entrada' seja especificado.
31023	Esse operador requer um parâmetro 'regex'.
31024	O operador recebe um argumento desconhecido.
31032	Um nome de campo não pode conter um byte nulo incorporado.
31034	Um valor está fora do intervalo permitido de [-32768, 32767].
31095	'iso WeekYear 'deve ser avaliado como um inteiro entre 1 e 9999.
31138	O valor $meta usado para classificação não é uma palavra-chave de metadados válida.
31170	Um objeto vazio era esperado, mas um valor não vazio foi fornecido.
31249	Um caminho de projeção colide com outro campo na projeção.
31254	Uma inclusão não pode ser especificada em uma projeção de exclusão.
31393	$bsonSize requer um documento como entrada.
31394	Um operador de projeção posicional não pode aparecer no meio de um caminho de campo.

Código de erro	Description
34413	Um operador inválido foi usado em uma consulta de geolocalização.
34435	O argumento para $reverseArray deve ser uma matriz.
34443	$range requer um valor inicial numérico.
34444	$range requer um valor inicial representável como um inteiro de 32 bits.
34445	$range exige que seu valor final seja numérico.
34446	O valor final de $range deve caber em um inteiro de 32 bits.
34447	$range requer um valor numérico de etapa.
34448	$range requer um valor de etapa representável como um inteiro de 32 bits.
34449	$range requer um valor de etapa diferente de zero
34450	O índice inicial de $SubstrCp deve ser do tipo numérico.
34451	O índice inicial $SubstrCp não pode ser representado como um inteiro de 32 bits.
34452	O argumento de comprimento para $substrCp não é um tipo numérico.
34453	O comprimento de $substrCp não pode ser representado como um inteiro de 32 bits.
34454	$substrCp: o comprimento deve ser um número inteiro não negativo.
34455	$substrCp: o índice inicial deve ser um inteiro não negativo.
34460	$zip só suporta um objeto como argumento.
34461	As entradas $zip devem ser uma matriz de expressões.
34462	A opção 'defaults' deve ser uma matriz de expressões.
34463	A LongestLength opção de uso de $zip deve ser booleana.

Código de erro	Description
34464	$zip recebeu um argumento desconhecido.
34465	$zip requer pelo menos uma matriz de entrada
34466	não pode especificar padrões, a menos que o uso seja verdadeiro LongestLength
34467	padrões e entradas devem ter o mesmo comprimento
34468	$zip recebe uma expressão que não seja de matriz em sua entrada.
34471	$strLenCP requer um argumento de string.
34473	$str LenBytes requer um argumento de string.
40060	$switch requer um objeto como argumento.
40061	$switch exige que 'branches' sejam uma matriz.
40062	$switch exige que cada ramificação seja um objeto.
40063	$switch recebe um argumento desconhecido dentro de uma ramificação.
40064	$switch exige que cada ramificação tenha uma expressão de 'caso'.
40065	$switch exige que cada ramificação tenha uma expressão 'então'.
40066	$switch não conseguiu encontrar uma ramificação correspondente para uma entrada e nenhum padrão foi especificado.
40067	A expressão $switch contém um argumento desconhecido.
40068	$switch requer pelo menos uma ramificação
40075	$reduce requer objeto
40076	Argumento desconhecido para $reduce
40077	$reduce requer 'entrada'

Código de erro	Description
40078	$reduce requer 'InitialValue'
40079	$reduce requer 'in'
40080	A entrada para $reduce deve ser uma matriz.
40081	O operador exige uma matriz como segundo argumento.
40085	$split requer seu primeiro argumento para ser avaliado como uma string.
40086	O segundo argumento de $split deve ser avaliado como uma string.
40087	$split requer um separador não vazio
40090	$index OfArray requer uma matriz como primeiro argumento
40091	O primeiro argumento para $index não OfBytes é uma string.
40092	$index OfBytes requer uma string como segundo argumento.
40093	$indexOfCp requer uma string como primeiro argumento.
40094	$indexOfCp requer uma string como segundo argumento.
40096	$index OfArray requer valores de índice iniciais e finais integrais.
40097	$index OfArray exige índices numéricos iniciais e finais não negativos.
40148	o ByCount campo de classificação deve ser definido como um caminho prefixado $ ou uma expressão dentro de um objeto
40149	o ByCount campo de classificação deve ser especificado como uma string ou como um objeto
40156	o campo de contagem deve ser uma string não vazia
40157	o campo de contagem deve ser uma string não vazia
40158	o campo de contagem não pode ser um caminho prefixado $

Código de erro	Description
40159	o campo de contagem não pode conter um byte nulo
40160	o campo de contagem não pode conter '.'
40169	a especificação $facet deve ser um objeto não vazio
40170	argumentos para $facet devem ser matrizes
40171	elementos de matrizes na especificação $facet devem ser objetos não vazios
40172	o subpipeline no estágio $facet não pode estar vazio
40176	Nomes de campo conflitantes fornecidos
40177	$project e $addFields exigem pelo menos um campo de saída.
40178	Não é possível excluir campos, exceto '_id' em uma projeção de inclusão
40179	Não é possível incluir campos ou redefinir campos nem adicionar campos computados em uma projeção de exclusão
40180	$project não permite um objeto vazio como valor de campo.
40181	$project:: inválido causado por:: uma especificação de expressão deve conter exatamente um campo.
40183	Um nome de campo pontilhado não pode ser usado dentro de um subobjeto.
40192	O campo $bucket 'limits' deve ter pelo menos 2 valores
40193	Todos os valores na opção 'limits' para $bucket devem ter o mesmo tipo
40194	A opção 'limits' para $bucket deve ser classificada, mas os elementos não estão em ordem ascendente
40195	O campo $bucket 'default' deve ser uma expressão constante
40196	O campo $bucket 'output' deve ser um objeto

Código de erro	Description
40197	Opção não reconhecida para $bucket
40198	$bucket requer que 'groupBy' e 'limits' sejam especificados
40199	O campo $bucket 'default' deve ser menor que o limite mais baixo ou maior ou igual ao limite mais alto.
40200	O campo $bucket 'limits' deve ser uma matriz
40201	O argumento para o estágio $bucket deve ser um objeto
40202	O campo $bucket 'groupBy' deve ser definido como um caminho prefixado em $ ou uma expressão
40203	O campo deve ser um objeto acumulador
40204	O nome do campo não pode conter '.'
40205	O nome do campo não pode ser um nome de operador
40206	O acumulador é um operador unário
40218	a consulta requer metadados de pontuação de texto, mas não está disponível
40228	A expressão $replaceRoot 'newRoot' deve ser avaliada como um objeto.
40229	$replaceRoot requer um objeto como especificação.
40230	$replaceRoot aceita apenas a opção 'newRoot'.
40234	O campo $group deve ser um objeto acumulador.
40235	Um nome de campo não pode conter um caractere '.'.
40236	Os nomes dos campos não podem começar com '$'
40237	Esse acumulador é um operador unário e recebe um único argumento.

Código de erro	Description
40239	O campo $bucketAuto 'groupBy' deve ser definido como um caminho prefixado $ ou um objeto de expressão
40240	O argumento para $bucketAuto deve ser um objeto
40241	O campo $bucketAuto 'buckets' deve ser um valor numérico
40242	O campo $bucketAuto 'buckets' deve ser representável como um inteiro de 32 bits
40243	O campo $bucketAuto 'buckets' deve ser maior que 0
40244	O campo 'output' $bucketAuto deve ser um objeto
40245	Opção não reconhecida para $bucketAuto
40246	$bucketAuto requer que 'groupBy' e 'buckets' sejam especificados
40257	Granularidade de arredondamento desconhecida
40258	$bucketAuto pode especificar uma 'granularidade' apenas com limites numéricos
40260	$bucketAuto pode especificar uma 'granularidade' apenas com números não negativos, mas encontrou um número negativo
40261	O campo de 'granularidade' $bucketAuto deve ser uma string
40272	A especificação do estágio $addFields deve ser um objeto.
40319	a especificação do estágio $lookup deve ser um objeto
40320	falta da opção 'from' para a especificação do estágio $lookup
40321	A opção 'from' para $lookup deve ser uma string
40322	namespace $lookup inválido

Código de erro	Description
40323	Um objeto de especificação do estágio do pipeline deve conter exatamente um campo.
40324	O pipeline contém um nome artístico não reconhecido.
40331	Esse estágio de agregação não é permitido dentro de um pipeline $facet.
40352	Os caminhos do campo $project não podem ser cadeias de caracteres vazias.
40353	Um caminho de campo não deve terminar com um '.'.
40386	$array ToObject requer uma entrada de matriz.
40390	$object ToArray requer um documento como entrada.
40391	$array ToObject requer um formato de entrada consistente. Todos os elementos devem ser matrizes ou objetos. Objeto detectado, agora encontrado: matriz
40392	$array ToObject requer objetos com exatamente as chaves 'k' e 'v'.
40393	$array ToObject requer um objeto com as chaves 'k' e 'v'. Falta uma ou ambas as chaves
40394	$array ToObject requer objetos com chaves de string 'k' e valores 'v'.
40395	$array ToObject requer pares de valores-chave cujas chaves são cadeias de caracteres.
40396	$array ToObject requer um formato de entrada consistente. Todos os elementos devem ser matrizes ou objetos. A matriz foi detectada, agora encontrada: objeto
40397	$array ToObject requer uma matriz de matrizes de dois elementos.
40398	$array ToObject recebe uma entrada de um tipo ou formato não reconhecido.
40400	$mergeObjects exige que todas as entradas sejam objetos.
40414	Um campo de comando obrigatório está ausente ou falhou na validação.

Código de erro	Description
40415	O comando contém um campo não reconhecido.
40422	Um nome de campo de matriz BSON tem um valor inválido.
40485	Um identificador de fuso horário não reconhecido foi fornecido.
40489	$date FromParts não permite misturar campos de calendário regulares com campos ISO de data da semana
40515	Um FromParts argumento $date não é avaliado como um número inteiro.
40516	$date FromParts requer que 'ano' ou 'isoWeekYear' estejam presentes
40517	A opção 'timezone' deve ser avaliada como uma string para operadores de data.
40518	$date FromParts ou $date ToParts obtiveram um parâmetro não reconhecido ou digitado incorretamente.
40519	$date FromParts só suporta um objeto como argumento.
40520	$date ToParts recebeu um parâmetro não reconhecido.
40522	$date ToParts requer um argumento de objeto com um parâmetro 'data'.
40523	$date FromParts 'year' deve ser um número inteiro no intervalo de 1 a 9999.
40525	week/day Os valores FromParts ISO de $date estão fora do intervalo ou faltam campos obrigatórios.
40533	O argumento do fuso horário deve ser uma string.
40535	Uma opção não reconhecida foi fornecida ao palco ou à expressão.
40536	A expressão aceita exatamente um argumento quando recebe uma matriz.
40539	O argumento 'data' necessário para um operador de data está ausente.
40540	$date FromString só suporta um objeto como argumento

Código de erro	Description
40541	$date FromString recebeu um argumento não reconhecido.
40542	Falta o parâmetro 'dateString' para $date FromString
40543	$date FromString exige que 'dateString' seja uma string.
40545	A date/time string está incompleta, com elementos obrigatórios ausentes.
40551	você não pode passar uma date/time string com informações de fuso horário junto com um argumento de fuso horário
40553	$date FromString não pode analisar a string de data fornecida.
40554	você não pode passar uma date/time string com deslocamento GMT junto com um argumento de fuso horário
40575	$changeStream 'fullDocument' deve ser “padrão” ou “updateLookup”.
40600	$out não pode ser usado em um estágio $facet.
40601	$out e $merge só podem ser o estágio final do pipeline.
40602	Um estágio como $changeStream ou $geoNear não é o primeiro no pipeline.
40603	$geoNear não foi o primeiro estágio no pipeline após a otimização. A otimização está desativada ou inibida?
40647	O token de resumo do fluxo de alterações tem um campo _data ausente ou digitado incorretamente.
40674	Somente um tipo de opção de currículo é permitido, mas vários foram encontrados.
50001	auto não IndexId:false é permitido para a coleção especificada.
50694	Esse operador recebeu um argumento desconhecido.
50695	Esse operador requer um campo de 'entrada'.

Código de erro	Description
50696	O operador só oferece suporte a um objeto como argumento.
50699	Esse operador de expressão exige que sua entrada seja uma string.
50700	O argumento 'chars' deve ser uma string.
50736	GetMore não pode ser executado em uma sessão em um cursor não criado em uma sessão.
50737	Executar GetMore em um cursor criado por sessão exige que um lsid seja fornecido.
50752	O índice inicial $substrBytes é negativo.
50808	O estágio $changeStream espera um documento como argumento.
51024	O valor da 'força' do agrupamento deve ser 5 ou menos.
51047	Esse operador ou estágio não é permitido em um subpipeline $lookup.
51075	Não é possível definir opções em $regex e $options
51076	Memória insuficiente para criar o índice vetorial para o número especificado de listas.
51078	Não foi possível encontrar o nome do índice.
51079	As estatísticas de latência não são suportadas
51091	Uma expressão regular é inválida ou está mal formada.
51103	O operador espera um objeto com argumentos nomeados, mas recebeu outro tipo.
51104	O operador exige que seu argumento de 'entrada' seja uma string.
51105	O argumento 'regex' de um operador não é um tipo de string ou regex.

Código de erro	Description
51106	Um operador exige que seu parâmetro 'options' seja uma string.
51107	As opções $regex não podem ser especificadas nos campos 'regex' e 'opções'.
51108	As opções de expressão regular contêm um sinalizador inválido ou um byte nulo incorporado.
51109	Uma expressão regular não pode conter um byte nulo incorporado.
51110	As opções de expressão regular não podem conter um byte nulo incorporado.
51111	A expressão regular fornecida a esse operador é inválida.
51151	O tamanho do buffer para armazenar a saída excedeu o limite do tipo de dados
51156	Falha na execução de uma expressão regular em $regex durante a avaliação.
51270	Um objeto de subprojeção vazio não é um valor de projeção válido.
51272	Erro de sintaxe: $projection requer pelo menos um campo de saída.
51744	O operador exige que 'substituição' seja uma string.
51745	O operador exige que seu argumento 'find' seja uma string.
51746	O argumento de 'entrada' de um operador não é uma string.
51747	Um operador exige um parâmetro de 'substituição'.
51748	Esse operador requer um parâmetro 'localizar'.
51749	Esse operador requer um parâmetro de 'entrada'.
51750	Um operador de agregação recebeu um argumento desconhecido.
100003	Erro interno do sistema ao processar o caminho de consulta otimizado
100005	Um limite configurado, como coleções, funções personalizadas ou privilégios, foi excedido.

Código de erro	Description
100006	O número máximo de usuários permitidos foi excedido.
3040501	$rand atualmente não aceita argumentos
3041701	$getField recebeu um argumento desconhecido em $project.
3041702	$project:: inválido causado por:: $getField exige que o 'campo' seja especificado
3041703	$project:: inválido causado por:: $getField requer que a 'entrada' seja especific ada
3041704	$getField requer que o argumento 'field' seja avaliado como uma string.
4161100	$setField e $unsetField suportam apenas um objeto como argumento.
4161101	$setField ou $unsetField recebeu um argumento desconhecido.
4161102	$setField e $unsetField exigem que um 'campo' seja especificado.
4161103	$project:: inválido causado por:: $setField exige que o 'valor' seja especificado
4161105	$setField ou $unsetField requerem 'entrada' para serem avaliados como um objeto.
4161106	$setField e $unsetField exigem que o 'campo' seja avaliado como uma constante .
4161107	$setField ou $unsetField requerem que 'campo' seja avaliado como uma string.
4161108	$project não permite uma referência de caminho de campo aqui; use {$literal:...} em vez disso.
4161109	$setField e $unsetField exigem que um campo de 'entrada' seja especificado.
4662500	O comando contém um campo não reconhecido.
4862100	O namespace totalmente qualificado excede o tamanho máximo permitido.

Código de erro	Description
5166301	$dateDiff só suporta um objeto como argumento
5166302	$dateDiff recebeu um argumento não reconhecido.
5166303	Falta o parâmetro 'startDate' para $dateDiff
5166304	Falta o parâmetro 'endDate' para $dateDiff
5166305	Falta o parâmetro 'unit' para $dateDiff
5166307	Um argumento $dateDiff não é uma data.
5166308	$DateDiff transbordou ao calcular a diferença de data.
5166400	$dateAdd e $dateSubtract exigem que seu argumento seja um objeto.
5166401	$dateAdd ou $dateSubtract receberam um argumento não reconhecido.
5166402	$dateAdd ou $dateSubtract exigem que a data inicial, a unidade e o valor estejam presentes.
5166403	$dateAdd exige que startDate seja conversível em uma data
5166404	$dateAdd espera uma string definindo a unidade de tempo
5166405	$dateAdd espera uma quantidade inteira de unidades de tempo
5166406	DateAdd transbordou
5166503	Foi fornecido um valor unitário de tempo desconhecido.
5439002	$dateTrunc transbordou ao calcular a data truncada.
5439006	Valor de binSize não suportado por DateTrunc
5439007	$dateTrunc só suporta um objeto como argumento
5439008	Argumento $dateTrunc não reconhecido; data esperada, unidade, tamanho do compartimento, fuso horário, início. OfWeek

Código de erro	Description
5439009	Falta o parâmetro 'date' para $dateTrunc
5439010	Falta o parâmetro 'unit' para $dateTrunc
5439012	$dateTrunc exige que 'data' seja uma data.
5439013	$dateDiff e $dateTrunc exigem que 'unit' seja uma string.
5439014	O valor de 'unidade' de $DateDiff e $DateTrunc não é uma unidade de tempo reconhecida.
5439015	$dateDiff ou $dateTrunc exigem que 'start' seja uma string. OfWeek
5439016	O valor OfWeek 'start' de $dateDiff ou $dateTrunc não é um dia da semana válido.
5439017	$dateTrunc exige que 'binSize' seja um inteiro de 64 bits.
5439018	$dateTrunc exige que 'binSize' seja maior que 0.
5447000	O estágio $collStats deve ser um objeto aninhado.
5626500	$geoNear, $near e $nearSphere não são permitidos nesse contexto, pois esses operadores exigem a classificação de dados geoespaciais. Se você não precisar classificar, considere usar $geoWithin em vez disso.
5654600	$project não permite uma referência de caminho de campo nesse contexto.
5654601	$project:: inválido causado por:: $getField requer que 'campo' seja avaliado como uma constante, mas recebeu um argumento não constante
5654602	$getField requer 'campo' para ser avaliado como uma string.
5733415	não consigo modificar $ por zero
5860400	$geoNear requer um argumento 'próximo'
5887502	Todos os operandos de $setEquals devem ser matrizes.

Código de erro	Description
5911200	PlanExecutor erro durante a agregação:: causado por:: $mergeObjects suporta apenas objetos
5976500	$dateAdd recebeu um valor de parâmetro de “quantidade” inválido.
6045000	$dateSubtract recebeu um valor de parâmetro de “quantidade” inválido.
7158303	$switch não encontrou nenhuma ramificação correspondente para a entrada e nenhum padrão foi especificado.

Notas da versão

Essas notas de lançamento descrevem os recursos, melhorias e correções de erros do Amazon DocumentDB organizados por versão principal do mecanismo. Em cada seção de versão, as entradas são listadas por data em formato de tabela.

Para obter detalhes sobre como as versões do mecanismo e as versões do patch são numeradas (incluindo o mapeamento entre os prefixos da versão do patch do mecanismo e as versões do mecanismo do Amazon DocumentDB), consulte. Numeração da versão do motor

É possível determinar a versão atual do patch do mecanismo do Amazon DocumentDB executando o seguinte comando:

```
db.runCommand({getEngineVersion: 1})
```

> **Note**
>
> Começando com o Amazon DocumentDB 5.0.2 e 8.0.2, esse comando retorna a versão do mecanismo e a versão do patch do mecanismo como uma única string no formulário. *engine-version*+*engine-patch-version* Por exemplo, um cluster no Amazon DocumentDB 5.0.2 retorna `5.0.2+3.0.12345` e um cluster no 8.0.2 retorna. `8.0.2+4.0.12345` Os clusters em versões anteriores retornam a versão do patch do mecanismo sozinha, como`3.0.12345`. Para obter mais informações, consulte Numeração da versão do motor.

Se o cluster não estiver na versão mais recente do mecanismo, é provável que você tenha uma manutenção pendente disponível para atualizar seu mecanismo. Para obter mais informações, consulte Manutenção do Amazon DocumentDB no Guia do Desenvolvedor.

Tópicos

- Notas de lançamento da versão do mecanismo Amazon DocumentDB
- Notas gerais de lançamento do Amazon DocumentDB
- Notas de lançamento dos clusters elásticos do Amazon DocumentDB

Notas de lançamento da versão do mecanismo Amazon DocumentDB

Notas de lançamento para as versões do mecanismo Amazon DocumentDB organizadas por versão principal.

Amazon DocumentDB 8.0

Notas de lançamento do mecanismo Amazon DocumentDB versão 8.0 (compatibilidade com MongoDB 8.0).

Amazon DocumentDB 8.0.2

Data	Notas da versão
28 de setembro de 2026	Amazon DocumentDB 8.0.2 (Versão do mecanismo: 4.0.12042) Novos recursos e melhorias 1. Foi adicionado suporte para o estágio de `$setWindowFields` agregação e seus operadores de janela na versão 3 do Planner. 2. Foi adicionado suporte para o estágio de `$bucketAuto` agregação na versão 3 do Planner. 3. Foi adicionado suporte para o estágio de `$facet` agregação na versão 3 do Planner. 4. Foi adicionado suporte para o estágio de `$graphLookup` agregação na versão 3 do Planner. 5. Foi adicionado suporte para `$lookup` subconsultas correlaci onadas, incluindo a sintaxe concisa de subconsultas correlaci onadas, com um subconjunto de estágios de pipeline na versão 3 do Planner. 6. Foi adicionado suporte para o estágio de `$changeStreamSplit LargeEvent` agregação. 7. Os fluxos de mudança agora emitem eventos para `createCol lection` operações.

Data	Notas da versão
	8. Os fluxos de mudança agora emitem eventos para `createIndex` operações. 9. Melhor desempenho de gravação do fluxo de alterações. 10. Foi adicionado suporte para gravações que podem ser repetidas. 11. Foi adicionado suporte para o `reIndex` comando em índices parciais. 12. A versão 3 do Planner agora suporta varreduras de índice em `$expr` predicados. 13. A versão 3 do Planner introduz o Index Only Scans de vários campos (IXONLYSCAN) para consultas de busca. Para obter mais informações, consulte Index-only escaneamentos (consultas cobertas). 14. A versão 3 do Planner introduz o Index Only Scans (IXONLYSCAN) para estágios de agregação, como. `$group` Para obter mais informações, consulte Index-only escaneamentos (consultas cobertas). 15. Otimização de desempenho para `$count` e `countDocuments()` na versão 3 do Planner. 16. Foi adicionado suporte para classificação incremental na versão 3 do Planner. Para obter mais informações, consulte Classificação incremental. 17. Seleção de plano aprimorada para consultas de índice de texto. 18. O plano explicativo para `$unwind` estágios coalescidos `$lookup` e estádios agora reflete o parâmetro. `includeArrayIndex` 19. Foi adicionado suporte ao filtro de índice em todos os comandos. 20. O planejador de consultas agora prioriza índices exclusivos para predicados de quality (). `$eq` 21. Aumentou o limite de truncamento para comandos agregados na saída. `current0p` Correções de bugs

Data	Notas da versão
	1. Corrigido um bug no planejador de consultas da versão 1 (padrão). 2. Corrigido o tratamento de ida e volta do texto de expressão de índice parcial durante o upgrade de versão principal (MVU). 3. Corrigido um problema de criação de perfil para consultas de contagem rápida. 4. Correção de segurança para o `collStats` comando.

Amazon DocumentDB 8.0.1

Data	Notas da versão
15 de setembro de 2026	Melhorias na segurança e estabilidade Amazon DocumentDB 8.0.1 (versão do mecanismo: 4.0.11638) [em andamento] O Amazon DocumentDB aplica automaticamente esse patch durante sua próxima janela de manutenção.
31 de agosto de 2026	Novo atributo Agora, o Amazon DocumentDB oferece suporte a atualizações de versões principais (MVU) no local diretamente das versões 3.6 e 4.0 do mecanismo para a versão 8.0. Isso elimina a necessidade de atualizações de versão intermediária e permite que você atualize seus clusters 3.6 ou 4.0, preservando dados, configurações e definições de cluster existentes. As principais atualizações de versões do Amazon DocumentDB 3.6 e 4.0 para 8.0 estão disponíveis em todas as AWS regiões em que essas versões são atualmente suportadas. Para obter mais informações, consulte Atualização da versão principal implementada do Amazon DocumentDB no local.
18 de agosto de 2026	Melhorias na segurança e estabilidade Amazon DocumentDB 8.0.1 (Versão do mecanismo: 4.0.9938)

Data	Notas da versão
	Esse patch de mecanismo fornece as seguintes melhorias: • Processamento aprimorado de solicitações O Amazon DocumentDB agora valida as solicitações de clientes recebidas de forma mais completa ao longo do caminho de conexão e análise de mensagens. Isso melhora a resiliência e a estabilidade em uma ampla variedade de padrões de entrada. • Fortalecendo o handshake de autenticação, o Amazon DocumentDB agora valida a troca de autenticação de forma mais completa. Isso melhora a robustez ao lidar com entradas inesperadas ou malformadas. O Amazon DocumentDB aplica automaticamente esse patch durante sua próxima janela de manutenção.
11 de agosto de 2026	Novos recursos Amazon DocumentDB 8.0.1 (Versão do mecanismo: 4.0.8864) • Suporte ativado para SCRAM-SHA-256 autenticação. • Habilitou o RPO ajustável para clusters globais do Amazon DocumentDB. • Recuperação aprimorada do armazenamento durante a coleta de lixo em segundo plano para reduzir o espaço de armazenamento não utilizado ao longo do tempo.
22 de julho de 2026	Novo atributo Os clusters do Amazon DocumentDB que executam a versão 8.0 e posterior do Engine agora suportam um limite máximo de armazenamento de 256 TiB, dobrando o limite anterior de 128 TiB. Para obter mais informações, consulte Cotas do Amazon DocumentDB.

Data	Notas da versão
13 de julho de 2026	Amazon DocumentDB 8.0.1 (Versão do mecanismo: 4.0.8864) Novos recursos e melhorias 1. Suporte para 13 operadores de acumulador no Planner padrão (v3):`$top`,,,`$bottom`,`$topN`,`$bottomN`,`$count`,`$firstN`,`$lastN`,,`$minN`, `$maxN``$median`,`$percentile` . `$stdDevPop` `$stdDevSamp` 2. Suporte para 36 operadores de expressão no Planner padrão (v3): `$median` `$percentile` `$stdDevPop` `$stdDevSamp` ,`$round`, `$sigmoid``$trunc`,`$firstN`,`$lastN`,`$maxN`,`$minN`,`$bitAnd`,`$bitNot`, `$bitOr``$bitXor`,`$binarySize` ,`$bsonSize` ,`$tsIncrement` ,`$tsSecond` ,`$cos`, `$acos``$cosh`,`$acosh`,`$sin`,`$asin`,`$sinh`,`$asinh`,`$tan`, `$atan``$tanh`,`$atan2`,`$atanh`, `$degreesT` `oRadians` `$radiansToDegrees` ,`$isNumber` . `$toUUID` 3. Suporte para `$sampleRate` in `$match` no Planner padrão (v3). 4. Suporte para o `$sortByCount` Planner padrão (v3). 5. Foi adicionado suporte para o estágio `$listSearchIndexes` de agregação no Planner padrão (v3). Retorna metadados do índice de pesquisa (nome, status, consultável, versão) para índices de pesquisa vetorial e de texto em uma coleção. 6. Suporta compressão rápida usando o tipo de mensagem OP_COMPRESSED. 7. Ofereça suporte aos métodos `min` e `max` cursor no comando find no Planner padrão (v3). 8. A versão 3 do Planner apresenta o Index Only Scans (IXONLYSC AN) para. `countDocuments()` 9. Foi adicionado suporte a índices ocultos para todos os tipos de índices (planejador versão 2 e superior). Correções de bugs

Data	Notas da versão
	1. Foi corrigido um problema em que consultas de `$text` pesquisa contendo caracteres sublinhados podiam retornar resultados incorretos quando avaliadas como um filtro residual.

Amazon DocumentDB 8.0.0

Data	Notas da versão
15 de setembro de 2026	Melhorias na segurança e estabilidade Amazon DocumentDB 8.0.0 (versão do mecanismo: 4.0.11440) [em andamento] Correções de bugs Esta versão inclui a seguinte correção de bug: 1. Foi corrigido um problema em que consultas de `$text` pesquisa contendo caracteres sublinhados podiam retornar resultados incorretos quando avaliadas como um filtro residual. O Amazon DocumentDB aplica automaticamente esse patch durante sua próxima janela de manutenção.
18 de agosto de 2026	Melhorias na segurança e estabilidade Amazon DocumentDB 8.0.0 (Versão do mecanismo: 4.0.9912) Esse patch de mecanismo fornece as seguintes melhorias: • Processamento aprimorado de solicitações O Amazon DocumentDB agora valida as solicitações de clientes recebidas de forma mais completa ao longo do caminho de conexão e análise de mensagens. Isso melhora a resiliência e a estabilidade em uma ampla variedade de padrões de entrada.

Data	Notas da versão
	• Fortalecendo o handshake de autenticação, o Amazon DocumentDB agora valida a troca de autenticação de forma mais completa. Isso melhora a robustez ao lidar com entradas inesperadas ou malformadas. Se você não aplicá-lo antes, o Amazon DocumentDB aplicará automaticamente esse patch durante sua próxima janela de manutenção, 30 dias após o lançamento.
22 de julho de 2026	Novo atributo Os clusters do Amazon DocumentDB que executam a versão 8.0 e posterior do Engine agora suportam um limite máximo de armazenamento de 256 TiB, dobrando o limite anterior de 128 TiB. Para obter mais informações, consulte Cotas do Amazon DocumentDB.
13 de maio de 2026	Novo atributo O Amazon DocumentDB serverless agora está disponível para o Amazon DocumentDB 8.0. Para obter mais informações, consulte Uso do Amazon DocumentDB Sem Servidor.
6 de maio de 2026	Correções de bugs e outras mudanças Amazon DocumentDB 8.0 (versão do mecanismo: 4.0.6940) Essa atualização fornece um patch de mecanismo opcional com as seguintes melhorias: 1. Corrigido um bug na coleta de lixo que poderia, em um caso extremo, fazer com que os leitores travassem.

Data	Notas da versão
20 de abril de 2026	Novo atributo In-place A atualização da versão principal (MVU) do Amazon DocumentDB 5.0 para o Amazon DocumentDB 8.0 já está disponível. Para obter mais informações, consulte Atualização da versão principal implementada do Amazon DocumentDB no local.

Data	Notas da versão
20 de abril de 2026	Novos recursos e melhorias Amazon DocumentDB 8.0 (versões do mecanismo: 4.0.5995, 4.0.6473) Esse é um patch obrigatório do mecanismo que inclui os seguintes novos recursos, melhorias e correções: 1. `$sortArray` operador em`$project`: • Suporte para `$sortArray` operador em `$project` estágio no pipeline de agregação para a versão 3 do Planner. O `$sortArray` operador classifica uma matriz com base em seus elementos (para valores escalares) ou em um campo especific ado (para matrizes de documentos), retornando a matriz classific ada em ordem crescente (1) ou decrescente (). `-1` 2. Suporte para`$out`,, `$collStats` `$indexStats` , `$geoNear` usando o Planner v3. Com o Planner V3, o `$out` palco também suporta a gravação em coleções em diferentes bancos de dados. 3. O pipeline de agregação agora se une `$lookup` e `$unwind` se organiza mesmo quando o `$unwind` estágio inclui o parâmetro . `includeArrayIndex` Anteriormente, essa otimização era ignorada quando `includeArrayIndex` estava presente. 4. Aprimoramentos de desempenho para melhorar a taxa de transferê ncia de db.r8g.8xlarge e instâncias maiores executadas na configuração. I/O-Optimized 5. Agora você pode monitorar a `Connections (active)` métrica para instâncias. Para obter mais informações, consulte Cotas de instância. 6. Suporte para operadores `$dateToParts` e `$dateFromParts` agregação. 7. Command-level As métricas de desempenho agora estão disponíve is CloudWatch para operações CRUD (`findinsertfindAndMo dify` ,`update`,`delete`,`aggregate` ,`count`,`distinct`, `getMoreabortTransaction` ,`commitTransaction`),

Data	Notas da versão
	exposição `AvgDuration` e `MaxConcurrent` dimensões para melhorar o monitoramento operacional. `P100Duration` 8. Corrigido um bug no `$replaceRoot` comportamento em que `newRoot` campos contendo valores booleanos de`true`/`false`ou valores numéricos de`0`/`1`estavam ausentes no documento de saída. 9. Corrigido um bug `$indexOfArray` em que consultas válidas retornavam erros indicando que eram inválidas. 10. O plano de explicação agora inclui `docsExamined` métricas na `executionStats` saída. 11. Desempenho aprimorado da autenticação SASL. 12. Atualizações otimizadas de documentos para reduzir a sobrecarg a de armazenamento quando os valores do índice permanecem inalterados e os documentos cabem nas páginas existentes. 13. Suporte para `bypassDocumentValidation` in`$merge`. Todos os clusters do Amazon DocumentDB 8.0 serão automatic amente atualizados para essa versão de patch durante a próxima janela de manutenção.
26 de março de 2026	Novos recursos O Amazon DocumentDB agora oferece suporte à última geração de instâncias otimizadas para memória, db.r8g, com processadores Arm-based AWS Graviton4, oferecendo desempenho até 30% melhor em relação às instâncias R6G. Para obter mais informações, consulte Gerenciar classes de instância neste guia

Data	Notas da versão
11 de novembro de 2025	Novos recursos Agora, o Amazon DocumentDB oferece total compatibilidade de protocolo de conexão com o MongoDB 8.0. O Amazon DocumentDB 8.0 melhora o desempenho das consultas, permitindo que você crie aplicativos de alto desempenho a um custo menor. Amazon DocumentDB 8.0 (patch de motor versão 4.0.4448) • Adicionado suporte para drivers da API MongoDB 8.0. • Foi adicionado suporte ao Planner Version3, que estende as melhorias de desempenho aos operadores do estágio de agregação , além de oferecer suporte a otimizações do pipeline de agregação e comandos distintos. • Foi adicionado suporte para compressão baseada em dicionário por meio do algoritmo de compressão Zstandard, melhorando a taxa de compressão em 5x. • Novos recursos adicionados: o Amazon DocumentDB 8.0 suporta agrupamento e visualizações. • Foi adicionado suporte para novos estágios e operadores de agregação: $replaceWith, $vectorSearch, $merge, $set, $unset, $bucket e 3 novos operadores de agregação $pow, $rand, $dateTrunc. • Suporte adicional para uma nova versão do índice de texto: O índice de texto v2 no Amazon DocumentDB 8.0 introduz tokens adicionais, aprimorando os recursos de pesquisa de texto. • Inclui melhorias na pesquisa vetorial por meio da criação de índices vetoriais paralelos, reduzindo o tempo de criação do índice em até 30x. • Suporte removido para TLS 1.0 e TLS 1.1. A versão mínima do TLS agora é TLS 1.2.

Amazon DocumentDB 5.0

Notas de lançamento do mecanismo Amazon DocumentDB versão 5.0 (compatibilidade com MongoDB 5.0).

Amazon DocumentDB 5.0.2

Data	Notas da versão
28 de setembro de 2026	Amazon DocumentDB 5.0.2 (Versão do mecanismo: 3.0.23065) Novos recursos e melhorias 1. Foi adicionado suporte a índices ocultos para todos os tipos de índices (planejador versão 2 e superior). 2. Suporte para os métodos `min()` e `max()` cursor no comando find. 3. Foi adicionado suporte ao filtro de índice em todos os comandos. 4. O planejador de consultas agora prioriza índices exclusivos para predicados de quality (). `$eq` 5. Aumentou o limite de truncamento para comandos agregados na saída. `currentOp` Correções de bugs 1. Corrigido um bug no planejador de consultas da versão 1 (padrão). 2. Corrigido o tratamento de ida e volta do texto de expressão de índice parcial durante o upgrade de versão principal (MVU). 3. Corrigido um problema de criação de perfil para consultas de contagem rápida.

Amazon DocumentDB 5.0.1

Data	Notas da versão
18 de agosto de 2026	Melhorias na segurança e estabilidade Amazon DocumentDB 5.0.1 (versão do mecanismo: 3.0.22319) [em andamento] Esse patch de mecanismo fornece as seguintes melhorias: • Processamento aprimorado de solicitações O Amazon DocumentDB agora valida as solicitações de clientes recebidas de forma mais completa ao longo do caminho de conexão e análise de mensagens. Isso melhora a resiliência e a estabilidade em uma ampla variedade de padrões de entrada. • Fortalecendo o handshake de autenticação, o Amazon DocumentDB agora valida a troca de autenticação de forma mais completa. Isso melhora a robustez ao lidar com entradas inesperadas ou malformadas. Se você não aplicá-lo antes, o Amazon DocumentDB aplicará automaticamente esse patch durante sua próxima janela de manutenção, 30 dias após o lançamento.
11 de agosto de 2026	Novo atributo Amazon DocumentDB 5.0.1 (Versão do mecanismo: 3.0.21138) Suporte ativado para SCRAM-SHA-256 autenticação.
8 de junho de 2026	Nova versão secundária Amazon DocumentDB 5.0.1 (versão do mecanismo 3.0.21138) O Amazon DocumentDB 5.0.1 é a primeira versão secundária da versão principal do Amazon DocumentDB 5.0. As versões secundárias incluem melhorias, como novos recursos, correções de segurança e correções de erros. Você pode atualizar para 5.0.1 de 5.0.0 usando

Data	Notas da versão
	uma atualização de versão secundária. Para obter mais informações, consulte Atualização da versão secundária do Amazon DocumentDB. Essa versão secundária inclui os seguintes novos recursos e aprimoramentos: 1. Foram `gcRuntimeStats` `documentFragmentStats` adicionadas e divididas as métricas do nível de coleta `storageSegmentExtended` em `storageSegmentBase` e `collstats` para mostrar mais informações sobre a integridade geral do processo de coleta de lixo. Para obter mais detalhes, consulte Coleta de resíduos no Amazon DocumentDB. 2. Suporte para operadores `$rand` e `$pow` agregação. 3. Agora você pode monitorar a `Connections (active)` métrica para instâncias. Para obter mais informações, consulte Cotas de instância. 4. Melhorou a resiliência entre regiões do cluster global, eliminando a reinicialização do cluster secundário com o Global Survivable Read Only. 5. Suporte para operadores `$dateToParts` e `$dateFromParts` agregação. 6. Command-level As métricas de desempenho agora estão disponíveis CloudWatch para operações CRUD (`findinsertfindAndModify` ,`update`,`delete`,`aggregate` ,`count`,`distinct`, `getMoreabortTransaction` ,`commitTransaction`), exposição `AvgDuration` e `MaxConcurrent` dimensões para melhorar o monitoramento operacional. `P100Duration` 7. `$replaceRoot` produz `newRoot` campos contendo valores booleanos (`true`/`false`) e valores numéricos (`0`/). `1` 8. `$indexOfArray` oferece suporte a uma variedade maior de expressões de consulta válidas. 9. O plano de explicação inclui `docsExamined` métricas na `executionStats` produção.

Data	Notas da versão
	10Desempenho aprimorado da autenticação SASL.

Amazon DocumentDB 5.0.0

Data	Notas da versão
18 de agosto de 2026	Melhorias na segurança e estabilidade Amazon DocumentDB 5.0.0 (versão do mecanismo: 3.0.22303) [em andamento] Esse patch de mecanismo fornece as seguintes melhorias: • Processamento aprimorado de solicitações O Amazon DocumentDB agora valida as solicitações de clientes recebidas de forma mais completa ao longo do caminho de conexão e análise de mensagens. Isso melhora a resiliência e a estabilidade em uma ampla variedade de padrões de entrada. • Fortalecendo o handshake de autenticação, o Amazon DocumentDB agora valida a troca de autenticação de forma mais completa. Isso melhora a robustez ao lidar com entradas inesperadas ou malformadas. • Access-control fiscalização O Amazon DocumentDB agora verifica os privilégios necessários de forma consistente em todos os níveis de aninhamento para operações de consulta aninhadas. Se você não aplicá-lo antes, o Amazon DocumentDB aplicará automaticamente esse patch durante sua próxima janela de manutenção, 30 dias após o lançamento.
6 de maio de 2026	Correções de bugs e outras mudanças Amazon DocumentDB 5.0 (Versão do mecanismo: 3.0.20421) Esta versão é para um patch de segurança necessário.

Data	Notas da versão
20 de abril de 2026	Novo atributo In-place A atualização da versão principal (MVU) do Amazon DocumentDB 5.0 para o Amazon DocumentDB 8.0 já está disponível. Para obter mais informações, consulte Atualização da versão principal implementada do Amazon DocumentDB no local.
13 de abril de 2026	Correções de bugs e outras mudanças Amazon DocumentDB 5.0 (versões do mecanismo: 3.0.19917) Esse é um patch obrigatório do mecanismo que inclui os seguintes novos recursos, melhorias e correções: 1. A `DatabaseConnectionsActiveMax` métrica agora rastreia `Connections (active)` as instâncias. Para obter mais informações, consulte Cotas de instância. 2. Melhor desempenho de autenticação SASL. Todos os clusters do Amazon DocumentDB 5.0 serão automatic amente atualizados para essa versão de patch durante a próxima janela de manutenção.

Data	Notas da versão
22 de outubro de 2025	Novos recursos Amazon DocumentDB 5.0 (patch de motor versão 3.0.17983) O Amazon DocumentDB agora oferece suporte à última geração de instâncias otimizadas para memória, db.r8g, com processadores Arm-based AWS Graviton4, oferecendo desempenho até 30% melhor em relação às instâncias R6G. Para obter mais informações, consulte Gerenciar classes de instância neste guia e esta postagem do blog. Novos recursos Amazon DocumentDB 5.0 (patch de motor versão 3.0.15902) • Foi adicionado suporte para um nome de coleção mais longo (comprimento de namespace de até 255 caracteres). • Foi introduzido um novo planejador de consultas (planejador versão 2.0). Para obter mais informações, consulte Planejador de consultas v2 neste guia. • Foram adicionados novos campos `collstats` para “mvccidStats” e “gcRuntimeStats” que fornecem informações sobre a integridade geral da coleção com relação à coleta de lixo e às coleções antigas.
16 de outubro de 2025	Novo atributo Amazon DocumentDB 5.0 (patch de motor versão 3.0.15902) O Amazon DocumentDB agora oferece suporte ao modo de pilha dupla (IPv4/IPv6) quando seu cluster está em uma nuvem privada virtual (VPC). Para obter mais informações, consulte esta postagem de blog e Amazon VPC e o Amazon DocumentDB no Guia do desenvolvedor do Amazon DocumentDB.

Data	Notas da versão
15 de setembro de 2025	Correções de bugs e outras mudanças Amazon DocumentDB 5.0 (patch de motor versão 3.0.17983) Foi corrigida a falha do mecanismo quando as consultas continham os estágios subsequentes `$replaceRoot` e `$lookup`.

Data	Notas da versão
29 de julho de 2025	Novos recursos Amazon DocumentDB 5.0 (patch de motor versão 3.0.15902) • Foi adicionado suporte ao operador de agregação `$regexFindAll` . • Foram lançadas as métricas de sobrecarga de índice e sobrecarga baseada em coleção • CloudWatch Métricas adicionadas `AvailableMVCCIds` e `LongestActiveGCRuntime` que fornecem informações sobre a integridade geral do cluster com relação à coleta de lixo e às coletas antigas. • O `serverStatus` comando agora inclui um campo “nvme_cache” que mostra a contagem de páginas gravadas e não gravadas no cache NVMe nas instâncias. NVMe-backed • Foi removida uma limitação no número de campos para `$group` e `$project` • Foi adicionado suporte para os operadores de string a seguir: `$trim`, `$rtrim`, `$ltrim`, `$regexFindAll` , `$replaceOne` e `$replaceAll` • Foi adicionado suporte para os operadores de matriz a seguir: `$first` e `$last` • Foi adicionado suporte para o operador de estágio a seguir: `$collStats` Correções de bugs e outras mudanças Amazon DocumentDB 5.0 (patch de motor versão 3.0.15902) Foi corrigido um problema durante a criação do índice que afetava coleções grandes.

Data	Notas da versão
28 de julho de 2025	Novos recursos Amazon DocumentDB 5.0 (patch de motor versão 3.0.14228) Os clusters baseados em instâncias do Amazon DocumentDB agora oferecem suporte ao Amazon DocumentDB Sem Servidor, uma configuração com ajuste de escala automático sob demanda. Para obter mais informações, consulte Uso do Amazon DocumentDB Sem Servidor e esta postagem o que há de novo do blog.
24 de março de 2025	Novo atributo Amazon DocumentDB 5.0 (patch de motor versão 3.0.11800) Foi adicionado suporte para `postBatchResumeToken` em fluxos de alterações. Para obter mais informações, consulte Retomando um fluxo de alterações com a postagem `BatchResumeToken`.

Data	Notas da versão
15 de janeiro de 2025	Novos recursos Amazon DocumentDB 5.0 (patch de motor versão 3.0.11800) O Amazon DocumentDB agora mantém a disponibilidade de leitura por meio da reinicialização da instância do gravador. Agora, as instâncias do leitor continuarão a atender às solicitações de leitura durante as reinicializações da instância do gravador. Correções de bugs e outras mudanças Amazon DocumentDB 5.0 (patch de motor versão 3.0.11800) • Foi corrigido o comando `killOp` para tratar um caso especial de inserções em massa. • Uso aprimorado da I/O-related memória de rede nas instâncias do Amazon DocumentDB. • Foi corrigido o comando `count` para consultas do filtro `$text`.
18 de dezembro de 2024	Novos recursos Amazon DocumentDB 5.0 (versão de patch do mecanismo 3.0.5721) Os clusters baseados em instâncias do Amazon DocumentDB agora oferecem suporte NVMe-backed a instâncias R6GD. Para obter mais informações, consulte NVMe-backed instâncias e esta postagem o que há de novo do blog.

Data	Notas da versão
6 de novembro de 2024	Novos recursos Amazon DocumentDB 5.0 (patch de motor versão 3.0.11051) • Foi adicionado suporte para versões mínimas de TLS `tls1.2+` e `tls1.3+` para parâmetros de cluster de `tls`. • Foi habilitado o suporte para os caracteres dólar ($) e ponto (.) nos nomes dos campos. Para diferenças funcionais, consulte Dólar ($) e ponto (.) nos nomes dos campos.
17 de setembro de 2024	Novos recursos Amazon DocumentDB 5.0 (versão de patch do mecanismo 3.0.10696) O Amazon DocumentDB agora oferece suporte à reconstrução de índices dentro do `reIndex` no `runCommand` . Para obter mais informações, consulte Manutenção do índice usando `reIndex`. Note O `reIndex` é compatível somente no Amazon DocumentDB 5.0 (versão de patch do mecanismo 3.0.10696 e superior). Correções de bugs e outras mudanças Amazon DocumentDB 5.0 (versão de patch do mecanismo 3.0.10696) • O `$setOnInsert` agora suporta o campo `_id` durante as inserções resultantes das operações de atualização. • Foi corrigido o problema que impedia a recuperação de espaço de armazenamento após uma atualização de versão principal para o Amazon DocumentDB 5.0.

Data	Notas da versão
22 de agosto de 2024	Novo atributo Amazon DocumentDB 5.0 (todas as versões de patch do mecanismo) Os clusters globais do Amazon DocumentDB agora oferecem suporte a transições de cluster e failovers de cluster gerenciados. Para obter mais informações, consulte Como realizar uma transição para um cluster global do Amazon DocumentDB e Realizar um failover gerenciado para um cluster global do Amazon DocumentDB. **Note** As transições e failovers de clusters globais têm suporte somente no Amazon DocumentDB 4.0 e 5.0.

<table>
<tr><th>Data</th><th>Notas da versão</th></tr>
<tr><td>23 de julho de 2024</td><td>
Novos recursos

Amazon DocumentDB 5.0 (versão de patch do mecanismo 3.0.8126)

• Adicionado suporte para nomes de índices mais longos (até 255 caracteres). Para obter mais informações, consulte Restrições de nomenclatura.
• Aumentamos o limite máximo de conexão em duas vezes:

<table>
<tr><th>Tipo de instância</th><th>Limite original</th><th>Novo limite</th></tr>
<tr><td>t3.medium</td><td>500</td><td>1000</td></tr>
<tr><td>t4g.medium</td><td>500</td><td>1000</td></tr>
<tr><td>r5.large</td><td>1700</td><td>3400</td></tr>
<tr><td>r5.xlarge</td><td>3500</td><td>7000</td></tr>
<tr><td>r5.2xlarge</td><td>7100</td><td>14200</td></tr>
<tr><td>r5.4xlarge</td><td>14200</td><td>28400</td></tr>
<tr><td>r5.8xlarge</td><td>28400</td><td>60000</td></tr>
<tr><td>r5.12xlarge</td><td>30000</td><td>60000</td></tr>
<tr><td>r5.16xlarge</td><td>30000</td><td>60000</td></tr>
<tr><td>r5.24xlarge</td><td>30000</td><td>60000</td></tr>
<tr><td>r6g.large</td><td>1700</td><td>3400</td></tr>
<tr><td>r6g.xlarge</td><td>3500</td><td>7000</td></tr>
<tr><td>r6g.2xlarge</td><td>7100</td><td>14200</td></tr>
<tr><td>r6g.4xlarge</td><td>14200</td><td>28400</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Data</th><th>Notas da versão</th></tr>
<tr><td></td><td>
<table>
<tr><th>Tipo de instância</th><th>Limite original</th><th>Novo limite</th></tr>
<tr><td>r6g.8xlarge</td><td>28400</td><td>60000</td></tr>
<tr><td>r6g.12xlarge</td><td>30000</td><td>60000</td></tr>
<tr><td>r6g.16xlarge</td><td>30000</td><td>60000</td></tr>
</table>
Para obter mais informações, consulte Cotas de instâncias.
Correções de bugs e outras mudanças
Amazon DocumentDB 5.0 (versão de patch do mecanismo 3.0.8126)
Aprimorou a lógica para sincronizar <code>CurrentTime</code> e <code>ResumeToken</code> para o fluxo de alterações nos leitores.
</td></tr>
</table>

Data	Notas da versão

22 de julho de 2024	Novos recursos
	
	Amazon DocumentDB 5.0 (versão de patch do mecanismo 3.0.6742)
	
	• Foi adicionado suporte à filtragem de auditoria DML. Agora você pode configurar condições de filtro para filtrar os registros de auditoria DML com base em seus requisitos específicos, em vez de registrar todas as consultas DML. Para obter mais informações, consulte Filtrar eventos de auditoria de DML.
	• Foi adicionado suporte na compactação de documentos para:
	  • Definir um limite mínimo de compactação
	  • Habilitar a compactação para coleções existentes (aplicável a novos documentos)
	  • Permitir a configuração de compactação padrão no nível do cluster
	
	  Para obter mais informações, consulte Gerenciamento da compactação de documentos a nível de coleção.
	• Foi adicionado suporte para consumir fluxos de alterações em instâncias de leitura. Para obter mais informações, consulte Usar fluxos de alterações em instâncias secundárias.

9 de julho de 2024	Novo atributo Para o Amazon DocumentDB 5.0 (patch mínimo do mecanismo versão 3.0.4780), as atualizações do certificado TLS CA não exigem mais a reinicialização do sistema. Para obter mais informações, consulte Atualizar seus certificados TLS do Amazon DocumentDB.

Data	Notas da versão
29 de maio de 2024	Novos recursos Amazon DocumentDB 5.0 (versão de patch do mecanismo 3.0.6742) • Foi adicionado suporte para os operadores `$regexMatch` e `$regexFind` . • Foi adicionado suporte para garantir precisão total nos registros de auditoria ao abordar números inteiros grandes. Os registros de auditoria agora mantêm a representação numérica exata de todos os números, evitando qualquer perda de precisão.
3 de abril de 2024	Novos recursos Amazon DocumentDB 5.0 (versão de patch do mecanismo 3.0.5721) • Foi adicionado suporte a `bypassDocumentValidation` e mensagem de erro granular para `$jsonSchema` . Para saber mais sobre o `bypassDocumentValidation` , consulte ignorar DocumentValidation. • Foi adicionado suporte para `$expr`. • Foi adicionado suporte para junções não correlacionadas `$lookup`. • Foi adicionado suporte para manter as regras de validação no estágio de agregação `$out`.

Data	Notas da versão
10 de janeiro de 2024	Novos recursos Amazon DocumentDB 5.0 (versões de patch do mecanismo 3.0.4574, 3.0.4780, 3.0.4960) • Foi adicionado suporte para índices vetoriais HNSW. Para obter mais informações, consulte Pesquisa vetorial para Amazon DocumentDB. • Foi adicionado suporte para índices parciais. Para obter mais informações, consulte Índice parcial. • Foi adicionado suporte para o runtime do GC em uma coleção dentro do comando `currentOp` . • Foi adicionado suporte ao índice de texto para pesquisa de texto nativo no Amazon DocumentDB. Para obter mais informações, consulte Realizar pesquisa de texto com o Amazon DocumentDB. • Foi adicionado suporte para palavras-chave do esquema `$jsonSchema` : `type`, `allOf`, `oneOf`, `anyOf`, `not`, `maxItems`, `minItems`, `maxProperties` , `minProperties` , `pattern`, `patternProperties` , `multipleOf` , `dependencies` e `uniqueItems` . Para obter mais informações, consulte Usando a validação do esquema JSON. • Foi adicionado suporte para operadores aritméticos `$ceil`, `$floor`, `$ln`, `$log`, `$log10`, `$sqrt` e `$exp`. Para obter mais informações, consulte Operadores aritméticos. • Foi adicionado suporte para o operador de expressão condicional `$switch`. • Foi adicionado suporte para compilações paralelas de índices vetoriais `IVFFLAT`. A documentação foi atualizada removendo a limitação de compilações paralelas de índices vetoriais `IVFFLAT` do guia do desenvolvedor.

Data	Notas da versão
	Correções de bugs e outras mudanças • Foi adicionada a funcionalidade de invocação sem distinção entre maiúsculas e minúsculas de `db.runCommand("dbstats")` . Clientes do Amazon DocumentDB 5.0 com versões de patch de mecanismo anteriores à 3.0.4960 ou 2.0.10221 devem aplicar esses patches de mecanismo mais recentes. • Corrigido um erro ao invocar `db.coll.stats()` no shell mongo versão 1.7 e posterior. A documentação foi atualizada removendo a dica de solução de problemas de `db.coll.stats()` do shell mongo do guia do desenvolvedor.
29 de novembro de 2023	Novos recursos Amazon DocumentDB 5.0 (versão de patch do mecanismo 3.0.3727) Foi adicionado suporte para pesquisa vetorial. Para obter mais informações, consulte esta postagem de blog e acesse Pesquisa vetorial para Amazon DocumentDB no Guia do desenvolvedor do Amazon DocumentDB.
21 de novembro de 2023	Novos recursos Amazon DocumentDB 5.0 (versão de patch do mecanismo 3.0.3727) Suporte adicional para I/O-optimized armazenamento. Para obter mais informações, consulte Configurações de armazenamento de cluster do Amazon DocumentDB no Guia do desenvolvedor do Amazon DocumentDB. Integração adicionada para aprendizado de máquina sem código com o SageMaker Canvas. Para obter mais informações, consulte No-code aprendizado de máquina com o Amazon SageMaker AI Canvas no Guia do desenvolvedor do Amazon DocumentDB.

Data	Notas da versão
6 de novembro de 2023	Amazon DocumentDB 5.0 (versões de patch de motor 3.0.3727) Novos recursos • Foi adicionado suporte para palavras-chave do esquema `$jsonSchema` `maxLength` , `minLength` , `maximum`, `minimum`, `exclusiveMaximum` , `exclusiveMinimum` , `items` e `additionalItems` . Observe que a validação do esquema JSON é suportada somente em clusters baseados em instâncias. • Foi adicionado suporte para o operador de pipeline de agregação `$convert` e seus operadores derivados abreviados `$toBool`, `$toInt`, `$toLong`, `$toDouble` , `$toString` , `$toDecimal` , `$toObjectId` e `$toDate`. • Foi adicionado suporte para operadores de expressão de conjunto `$setDifference` , `$anyElementTrue` e `$allElementTrue` . Correções de bugs e outras mudanças Foi corrigido um problema de atualização de fluxo de alterações de `-NaN` para `NaN` que não estava sendo exibida.
15 de setembro de 2023	Novos recursos Amazon DocumentDB 5.0 (patch de motor versão 3.0.3140) Foi adicionado suporte para o validador de esquema $jsonSche ma somente em clusters baseados em instâncias. Para obter mais informações, consulte Usando a validação do esquema JSON.

Data	Notas da versão
13 de julho de 2023	Novos recursos Amazon DocumentDB 5.0 (versão de patch do mecanismo 3.0.1948) • Adicionado suporte para compactação de documentos. • Foi adicionado suporte para compilações paralelas de índices. • Foi adicionado suporte para o status de compilação do índice. Correções de bugs e outras mudanças Amazon DocumentDB 5.0 (versão de patch do mecanismo 3.0.1948) • Corrigido um problema de autenticação com o `createCollection` para clusters elásticos do Amazon DocumentDB quando os usuários não têm acesso às coleções do sistema. • Foi corrigido o problema em que as instâncias secundárias da região não podiam usar os mesmos nomes das instâncias da região primária.
7 de junho de 2023	Correções de bugs e outras mudanças Amazon DocumentDB 5.0 • Agora há suporte para as instâncias r5 e t3.medium no Amazon DocumentDB 5.0. • `engineVersion` a opção padrão está `5.0.0` no AWS SDK AWS CLI, e. CloudFormation

Data	Notas da versão
10 de maio de 2023	Correções de bugs e outras mudanças Amazon DocumentDB 5.0 (versão de patch do mecanismo 3.0.1361) • Suporte adicionado para `ignoreunknownindexoptions` no comando createIndex. • Interrompida a adição de consultas de monitoramento interno aos registros de auditoria. • Os nomes de variáveis locais definidos pelo usuário agora oferecem suporte a “_” (sublinhado) para operadores de projeção, como `$let` e `$filter`.

Data	Notas da versão

1 de março de 2023	Novos recursos
	
	Amazon DocumentDB 5.0 (versão de patch do mecanismo 3.0.775)
	
	• Apresentado o Amazon DocumentDB 5.0
	  • Compatibilidade com o MongoDB 5.0 (suporte para drivers de API MongoDB 5.0)
	  • Suporte para criptografia em nível de Client-side campo (FLE). Agora você pode criptografar campos no lado do cliente antes de gravar os dados no cluster do Amazon DocumentDB. Para obter mais informações, consulte criptografia em nível de Client-side campo
	  • Novos operadores de agregação: `$dateAdd`, `$dateSubtract`
	• Aumento do limite de armazenamento para 128 TiB para todos os clusters do Amazon DocumentDB baseados em instâncias e clusters elásticos baseados em fragmentos.
	• O Amazon DocumentDB 5.0 agora oferece suporte a escaneamentos de índice com o operador `$elemMatch` no primeiro nível de aninhamento. Varreduras de índice são suportadas quando a consulta tem apenas um nível de filtro `$elemMatch` e a consulta `$elemMatch` aninhada não oferece suporte à varredura de índice.
	
	Forma de consulta compatível com varredura de índice:
	
	`db.foo.find( { "a": {$elemMatch: { "b": "xyz", "c": "abc"} } })`
	
	Forma de consulta não suporta varredura de índice:
	
	`db.foo.find( { "a": {$elemMatch: { "b": {$elemMatch: { "d": "xyz", "e": "abc"} }} } })`

Amazon DocumentDB 4.0

Notas de lançamento do mecanismo Amazon DocumentDB versão 4.0 (compatibilidade com MongoDB 4.0).

Data	Notas da versão
31 de agosto de 2026	Novo atributo Agora, o Amazon DocumentDB oferece suporte a atualizações de versões principais (MVU) no local diretamente da versão 4.0 do mecanismo para a versão 8.0. Isso elimina a necessidade de atualizaç ões de versão intermediária e permite que você atualize seus clusters 4.0 enquanto preserva os dados, configurações e definições de cluster existentes. Essa atualização está disponível em todas as AWS regiões em que essas versões são suportadas atualmente. Para obter mais informações, consulte Atualização da versão principal implement ada do Amazon DocumentDB no local.
18 de agosto de 2026	Melhorias na segurança e estabilidade Amazon DocumentDB 4.0 (versão do mecanismo: 2.0.14287) [em andamento] Esse patch de mecanismo fornece as seguintes melhorias: • Processamento aprimorado de solicitações O Amazon DocumentD B agora valida as solicitações de clientes recebidas de forma mais completa ao longo do caminho de conexão e análise de mensagens. Isso melhora a resiliência e a estabilidade em uma ampla variedade de padrões de entrada. • Fortalecendo o handshake de autenticação, o Amazon DocumentD B agora valida a troca de autenticação de forma mais completa. Isso melhora a robustez ao lidar com entradas inesperadas ou malformadas. • Access-control fiscalização O Amazon DocumentDB agora verifica os privilégios necessários de forma consistente em todos os níveis de aninhamento para operações de consulta aninhadas.

Data	Notas da versão
	Se você não aplicá-lo antes, o Amazon DocumentDB aplicará automaticamente esse patch durante sua próxima janela de manutenção, 30 dias após o lançamento.
27 de abril de 2026	Correções de bugs e outras mudanças Amazon DocumentDB 4.0 (versão do mecanismo: 2.0.14278) Esse é um patch de mecanismo necessário que soluciona os seguintes problemas: 1. Foi corrigido um problema que, em casos raros, fazia com que a recuperação pontual (PITR) e as principais atualizações de versão (MVU) parassem de responder. Todos os clusters do Amazon DocumentDB 4.0 serão automatic amente atualizados para essa versão de patch durante a próxima janela de manutenção.
9 de março de 2026	Correções de bugs e outras mudanças Amazon DocumentDB 4.0 (versão do mecanismo: 2.0.14165) Essa atualização fornece um patch de mecanismo opcional com as seguintes melhorias: 1. A `DatabaseConnectionsActiveMax` métrica agora rastreia `Connections (active)` as instâncias. Para obter mais informações, consulte Cotas de instância. 2. Melhor desempenho de autenticação SASL.

Data	Notas da versão
22 de outubro de 2025	Novos recursos Amazon DocumentDB 4.0 (patch de motor versão 2.0.11747) Foram adicionados novos campos `collstats` para “mvccidStats” e “gcRuntimeStats” que fornecem informações sobre a integridade geral da coleção com relação à coleta de lixo e às coleções antigas.
16 de outubro de 2025	Novo atributo Amazon DocumentDB 4.0 (patch de motor versão 2.0.11747) O Amazon DocumentDB agora oferece suporte ao modo de pilha dupla (IPv4/IPv6) quando seu cluster está em uma nuvem privada virtual (VPC). Para obter mais informações, consulte esta postagem de blog e Amazon VPC e o Amazon DocumentDB no Guia do desenvolvedor do Amazon DocumentDB.

Data	Notas da versão
29 de julho de 2025	Novos recursos Amazon DocumentDB 4.0 (patch de motor versão 2.0.11747) • Foram lançadas as métricas de sobrecarga de índice e sobrecarga baseada em coleção • CloudWatch Métricas adicionadas `AvailableMVCCIds` e `LongestActiveGCRuntime` que fornecem informações sobre a integridade geral do cluster com relação à coleta de lixo e às coletas antigas. • Foi adicionado suporte para os operadores de string a seguir :`$trim`, `$rtrim` e `$ltrim` • Foi adicionado suporte para o operador de estágio a seguir: `$collStats` Correções de bugs e outras mudanças Amazon DocumentDB 4.0 (patch de motor versão 2.0.11747) Foi corrigido um problema durante a criação do índice que afetava coleções grandes.
15 de janeiro de 2025	Correções de bugs e outras mudanças Amazon DocumentDB 4.0 (patch de motor versão 2.0.11096) • Foi corrigido o comando `killOp` para tratar um caso especial de inserções em massa. • Uso aprimorado da I/O-related memória de rede nas instâncias do Amazon DocumentDB.

Data	Notas da versão
6 de novembro de 2024	Novos recursos Amazon DocumentDB 4.0 (patch de motor versão 2.0.10980) • Foi adicionado suporte para versões mínimas de TLS `tls1.2+` e `tls1.3+` para parâmetros de cluster de `tls`.
17 de setembro de 2024	Correções de bugs e outras mudanças Amazon DocumentDB 4.0 (patch de motor versão 2.0.10898) • O `$setOnInsert` agora suporta o campo `_id` durante as inserções resultantes das operações de atualização. • Foi corrigido o problema que impedia a recuperação de espaço de armazenamento após uma atualização de versão principal para o Amazon DocumentDB 5.0.
22 de agosto de 2024	Novo atributo Amazon DocumentDB 4.0 (patch de motor versão 2.0.5704) Os clusters globais do Amazon DocumentDB agora oferecem suporte a transições de cluster e failovers de cluster gerenciados. Para obter mais informações, consulte Como realizar uma transição para um cluster global do Amazon DocumentDB e Realizar um failover gerenciado para um cluster global do Amazon DocumentDB. **Note** As transições e failovers de clusters globais têm suporte somente no Amazon DocumentDB 4.0 e 5.0.

<table>
<tr><th>Data</th><th>Notas da versão</th></tr>
<tr><td>23 de julho de 2024</td><td>
Novos recursos

Amazon DocumentDB 4.0 (patch de motor versão 2.0.10709)

• Adicionado suporte para nomes de índices mais longos (até 255 caracteres). Para obter mais informações, consulte Restrições de nomenclatura.
• Aumentamos o limite máximo de conexão em duas vezes:

<table>
<tr><th>Tipo de instância</th><th>Limite original</th><th>Novo limite</th></tr>
<tr><td>t3.medium</td><td>500</td><td>1000</td></tr>
<tr><td>t4g.medium</td><td>500</td><td>1000</td></tr>
<tr><td>r5.large</td><td>1700</td><td>3400</td></tr>
<tr><td>r5.xlarge</td><td>3500</td><td>7000</td></tr>
<tr><td>r5.2xlarge</td><td>7100</td><td>14200</td></tr>
<tr><td>r5.4xlarge</td><td>14200</td><td>28400</td></tr>
<tr><td>r5.8xlarge</td><td>28400</td><td>60000</td></tr>
<tr><td>r5.12xlarge</td><td>30000</td><td>60000</td></tr>
<tr><td>r5.16xlarge</td><td>30000</td><td>60000</td></tr>
<tr><td>r5.24xlarge</td><td>30000</td><td>60000</td></tr>
<tr><td>r6g.large</td><td>1700</td><td>3400</td></tr>
<tr><td>r6g.xlarge</td><td>3500</td><td>7000</td></tr>
<tr><td>r6g.2xlarge</td><td>7100</td><td>14200</td></tr>
<tr><td>r6g.4xlarge</td><td>14200</td><td>28400</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Data</th><th>Notas da versão</th></tr>
<tr><td></td><td>
<table>
<tr><th>Tipo de instância</th><th>Limite original</th><th>Novo limite</th></tr>
<tr><td>r6g.8xlarge</td><td>28400</td><td>60000</td></tr>
<tr><td>r6g.12xlarge</td><td>30000</td><td>60000</td></tr>
<tr><td>r6g.16xlarge</td><td>30000</td><td>60000</td></tr>
</table>
Para obter mais informações, consulte Cotas de instâncias.
</td></tr>
<tr><td>22 de julho de 2024</td><td>Novos recursos

Amazon DocumentDB 4.0 (versão de patch do mecanismo 2.0.10593)

• Foi adicionado suporte à filtragem de auditoria DML. Agora você pode configurar condições de filtro para filtrar os registros de auditoria DML com base em seus requisitos específicos, em vez de registrar todas as consultas DML. Para obter mais informações, consulte Filtrar eventos de auditoria de DML.</td></tr>
<tr><td>9 de julho de 2024</td><td>Novo atributo

Para o Amazon DocumentDB 4.0 (patch mínimo do mecanismo versão 2.0.10179), as atualizações do certificado TLS CA não exigem mais a reinicialização do sistema. Para obter mais informações, consulte Atualizar seus certificados TLS do Amazon DocumentDB.</td></tr>
<tr><td>29 de maio de 2024</td><td>Novos recursos

Amazon DocumentDB 4.0 (versão de patch do mecanismo 2.0.10593)

• Foi adicionado suporte para garantir precisão total nos registros de auditoria ao abordar números inteiros grandes. Os registros de auditoria agora mantêm a representação numérica exata de todos os números, evitando qualquer perda de precisão.</td></tr>
</table>

Data	Notas da versão
3 de abril de 2024	Novos recursos Amazon DocumentDB 4.0 (versão de patch do mecanismo 2.0.10392) • Foi adicionado suporte para `bypassDocumentValidation` no `$jsonSchema`. Para saber mais sobre o `bypassDocumentValidation`, consulte ignorar DocumentValidation. • Foi adicionado suporte para `$expr`. • Foi adicionado suporte para junções não correlacionadas `$lookup`. • Foi adicionado suporte para manter as regras de validação no estágio de agregação `$out`.
10 de janeiro de 2024	Novos recursos Amazon DocumentDB 4.0 (versões de patch do mecanismo 2.0.10124, 2.0.10179, 2.0.10221) • Foi adicionado suporte para o runtime do GC em uma coleção dentro do comando `currentOp`. • Foi adicionado suporte para palavras-chave do esquema `$jsonSchema`: `type`, `allOf`, `oneOf`, `anyOf`, `not`, `maxItems`, `minItems`, `maxProperties`, `minProperties`, `pattern`, `patternProperties`, `multipleOf`, `dependencies` e `uniqueItems`. Para obter mais informações, consulte Usando a validação do esquema JSON. • Foi adicionado suporte para operadores aritméticos `$ceil`, `$floor`, `$ln`, `$log`, `$log10`, `$sqrt` e `$exp`. Para obter mais informações, consulte Operadores aritméticos. • Foi adicionado suporte para o operador `$switch` de expressão condicional.

Data	Notas da versão
20 de dezembro de 2023	Outras alterações Suporte habilitado para atualização local da versão principal no Amazon DocumentDB 4.0. Para obter mais informações, consulte Atualização da versão principal implementada do Amazon DocumentDB no local.
6 de novembro de 2023	Amazon DocumentDB 4.0 (versões de patch de motor 2.0.9876) Novos recursos • Foi adicionado suporte para palavras-chave do esquema `$jsonSchema maxLength` , `minLength` , `maximum`, `minimum`, `exclusiveMaximum` , `exclusiveMinimum` , `items` e `additionalItems` . Observe que a validação do esquema JSON é suportada somente em clusters baseados em instâncias. • Foi adicionado suporte para o operador de pipeline de agregação `$convert` e seus operadores derivados abreviados `$toBool`, `$toInt`, `$toLong`, `$toDouble` , `$toString` , `$toDecimal` , `$toObjectId` e `$toDate`. • Foi adicionado suporte para operadores de expressão de conjunto `$setDifference` , `$anyElementTrue` e `$allElementTrue` . Correções de bugs e outras mudanças Foi corrigido um problema de atualização de fluxo de alterações de `-NaN` para `NaN` que não estava sendo exibida.
20 de setembro de 2023	Novos recursos Foi adicionado suporte para atualizações locais de versões principais no Amazon DocumentDB 4.0. Para obter mais informações, consulte Atualização da versão principal implementada do Amazon DocumentDB no local.

Data	Notas da versão
15 de setembro de 2023	Novos recursos Amazon DocumentDB 4.0 (patch de motor versão 2.0.9686) Foi adicionado suporte para o validador de esquema $jsonSchema somente em clusters baseados em instâncias. Para obter mais informações, consulte Usando a validação do esquema JSON.
13 de julho de 2023	Novos recursos Amazon DocumentDB 4.0 (versão de patch do mecanismo 2.0.9259) • Foi adicionado suporte para compilações paralelas de índices. Correções de bugs e outras mudanças Amazon DocumentDB 4.0 (versão de patch do mecanismo 2.0.9259) • Interrompida a adição de consultas de monitoramento interno aos registros de auditoria.

Data	Notas da versão
4 de abril de 2023	Correções de bugs e outras mudanças Amazon DocumentDB 4.0 (versão de patch do mecanismo 2.0.8934) • Corrigido problema com a auditoria de DML quando habilitada durante uma workload contínua. • Corrigido um problema com a auditoria de DML quando comandos agregados com dica recebem um valor de string. • Corrigido um problema em que o comando `listCollections` não funcionava quando usuários com a função readwriteanydataba se tinham as opções AuthorizedCollections e NameOnly definidas como verdadeiras. • Corrigido problema para analisar corretamente a string numérica em um nome de campo. • Cancele cursores de longa duração quando eles estiverem afetando a coleta de resíduos. • Os nomes de variáveis locais definidos pelo usuário agora oferecem suporte a “_” (sublinhado) para operadores de projeção, como `$let` e `$filter`.
27 de fevereiro de 2023	Correções de bugs e outras mudanças Amazon DocumentDB 4.0 Suporte adicionado para AWS Lambda. Para obter mais informações, consulte Usando AWS Lambda com fluxos de alterações.

Data	Notas da versão
9 de agosto de 2022	Novos recursos 4,0 • Adicionado suporte para tipo de dado Decimal128. O Decimal128 é um tipo de dados BSON suportado em todas as regiões em que o DocumentDB está disponível. Para obter mais informações, consulte Tipos de dados. • Foi adicionado suporte para auditoria de consultas DML com o Amazon CloudWatch Logs. Agora, o Amazon DocumentDB pode gravar eventos de linguagem de manipulação de dados (DML) e eventos de linguagem de definição de dados (DDL) no Amazon Logs. CloudWatch Veja este post de blog para obter mais informações. Correções de bugs e outras mudanças 4,0 • Agora, você pode alterar a própria senha com o privilégio `changeOwnPassword` .
25 de julho de 2022	Novos recursos Amazon DocumentDB 4.0 Agora você pode criar clusters mais rapidamente com a capacidade de criar clones que usam o mesmo volume de cluster do DocumentDB e têm os mesmos dados do cluster original. Para detalhes, consulte Gerenciar clusters do Amazon DocumentDB.

Data	Notas da versão
27 de junho de 2022	Novos recursos Amazon DocumentDB 4.0 (versão de patch do mecanismo 2.0.7509) O Amazon DocumentDB redimensiona dinamicamente seu banco de dados com base nos padrões de uso. Adicionar dados aumenta o espaço em até 64 Tebibytes (TiB) e excluir dados diminui o espaço alocado.
7 de abril de 2022	Novos recursos 4.0 (versões de patch de motor 2.0.6142 e 2.0.6948) O Amazon DocumentDB Performance Insights agora está disponível para visualização. Agora, você pode armazenar sete dias de histórico de desempenho em uma janela contínua sem custo adicional. Para obter mais informações, consulte Monitorando com insights de desempenho.

24 de janeiro de 2022	Novos recursos
	
	4.0 (Patch de motor versão 2.0.5170)
	
	• O Amazon DocumentDB agora oferece um teste gratuito. Para detalhes, consulte a página de teste gratuito do Amazon DocumentDB.
	• Agora, você pode usar atributos aprimorados com a consulta Geoespacial, que inclui as seguintes APIs:
	  • `$geoWithin`
	  • `$geoIntersects`
	• Adicionado o suporte para os seguintes operadores MongoDB:
	  • `$mergeObjects`
	  • `$reduce`
	
	Para mais informações, consulte Consultar dados geoespaciais com o Amazon DocumentDB.

Data	Notas da versão

21 de janeiro de 2022	Novos recursos
	
	Amazon DocumentDB 4.0 (versão de patch do mecanismo 2.0.5706)
	
	• Agora, há suporte para as instâncias do Amazon DocumentDB Graviton2 (r6g.large, r6g.2xlarge, r6g.4xlarge, r6g.8xlarge, r6g.12xlarge, r6g.16xlarge e t4g.medium)
	
	Novos recursos
	
	4.0 (Patch de motor versão 2.0.5706)
	
	• Suporte adicional para as seguintes APIs do MongoDB:
	  • `$reduce`
	  • `$mergeObjects`
	  • `$geoWithin`
	  • `$geoIntersects`

Data	Notas da versão

25 de outubro de 2021	Novos recursos
	
	4.0 (Patch de motor versão 2.0.5704)
	
	• Suporte adicional para as seguintes APIs do MongoDB
	  • `$literal`
	  • `$map`
	  • `$$ROOT`
	• Suporte para recursos de GeoSpatial consulta. Veja este post de blog para obter detalhes
	• Suporte para controle de acesso com perfis definidos pelo usuário. Veja este post de blog para obter detalhes
	• Driver JDBC do Amazon DocumentDB para permitir a conectividade de ferramentas de BI como o Tableau e ferramentas de consulta como o SQL Workbench
	
	Correções de bugs e outras mudanças
	
	4.0 (Patch de motor versão 2.0.5704)
	
	• Correção de bug para `$natural` classificar corretamente quando um `.sort()` explícito estiver presente com `$natural`
	• Correção de bug para o fluxo de alterações trabalhar com `$redact`
	• Correção de bug para `$ifNull` trabalhar com matriz vazia
	• Correção de bug para consumption/server falha excessiva de recursos quando um usuário atualmente conectado é excluído ou o privilégio desse usuário para uma atividade contínua é revogado
	• Correção de bug em `listDatabase` e na verificação de privilégi os `listCollection`
	• Solução de bug de lógica de desduplicação para elementos de múltiplas chaves

Data	Notas da versão
24 de junho de 2021	Novos recursos 4.0 (Patch de motor versão 2.0.3371) • Agora, há suporte para instâncias r5.8xlarge e r5.16xlarge. Saiba mais no post de blog Amazon DocumentDB agora suporta instâncias r5.8xlarge e r5.16xlarge. • Agora, os clusters globais são suportados para fornecer recuperação de desastres em caso de Region-wide interrupções e permitir leituras globais de baixa latência, permitindo leituras do cluster Amazon DocumentDB mais próximo. Observe que, no momento, não há suporte para clusters globais para as regiões América do Sul (São Paulo), Europa (Milão), China (Pequim) e China (Ningxia).

Data	Notas da versão
4 de maio de 2021	Novos recursos 4.0 (Patch de motor versão 2.0.3371) • `renameCollection` • `$zip` • `$indexOfArray` • `$reverseArray` • `$natural` • Suporte do `$hint` para atualização • Varredura de índice para `distinct` Correções de bugs e outras mudanças 4.0 (Patch de motor versão 2.0.3371) • Uso de memória reduzido para consultas `$in` • Corrigido um vazamento de memória em índices de várias chaves • Corrigido o plano de explicação e a saída do profiler para `$out` • Adicionado um tempo limite para as operações do sistema de monitoramento interno para melhoria de confiabilidade • Corrigido um defeito que impactava os predicados de consulta passados a índices de várias chaves
15 de janeiro de 2021	Novos recursos Amazon DocumentDB 4.0 (versão de patch do mecanismo 2.0.722) • Nenhum

Data	Notas da versão
9 de novembro de 2020	Novos recursos Amazon DocumentDB 4.0 (versão de patch do mecanismo 2.0.722) • Compatibilidade do MongoDB 4.0 • Transações ACID • Suporte para `cluster(client.watch()` ou `mongo.watch())` e fluxos de alterações `(db.watch())` do nível do banco de dados • Capacidade de iniciar ou retomar fluxos de alteração usando `startAtOperationTime` • Estenda o período de retenção do fluxo de alterações para 7 dias (antes, 24 horas) • AWS DMS alvo para o Amazon DocumentDB 4.0 • CloudWatch métricas: `TransactionsOpen` `TransactionsOpenMax` ,`TransactionsAborted` ,`TransactionsStarted` , e `TransactionsCommitted` • Novos campos para transações em `currentOp` , `ServerStatus` e `profiler`. • Capacidade de uso de um índice com o estágio de agregação `$lookup` • Consultas `find()` com projeções que podem ser servidas na direção de um índice (consulta coberta) • Capacidade de usar `hint()` com o `findAndModify` • Otimizações de desempenho para o operador `$addToSet` • Melhorias para reduzir os tamanhos gerais dos índices. • Novos operadores de agregação: `$ifNull`, `$replaceRoot` , `$setIsSubset` , `$setInstersection` , `$setUnion` e `$setEquals` • Com os comandos `ListCollection` e `ListDatabase` , agora, você pode opcionalmente usar os parâmetros `authorizedCollections` e `authorizedDatabases` para permitir

Data	Notas da versão
	que os usuários listem as coleções e bancos de dados que tiverem permissão para acessar sem exigirem as funções `listCollections` e `listDatabase`, respectivamente • Os usuários também podem finalizar os próprios cursores sem exigir a função `KillCursor` • A comparação de tipos numéricos de subdocumentos agora é consistente com a comparação de tipos numéricos de documentos de primeiro nível. O comportamento no Amazon DocumentDB 4.0 agora é compatível com o MongoDB. Correções de bugs e outras mudanças Amazon DocumentDB 4.0 (versão de patch do mecanismo 2.0.722) • O `$setOnInsert` não permite mais atualizações ao usar o operador posicional $. O comportamento no Amazon DocumentDB 4.0 agora é compatível com o MongoDB. • Problema corrigido com `$createCollection` e definido `autoIndexId` • Projeção para documentos aninhados • Configuração padrão alterada para a memória de trabalho para escalar com o tamanho da memória da instância • Melhorias da coleta de resíduos • Pesquisa com chave vazia no caminho, diferença de comportamento com mongo • Corrigido um bug `dateToString` no comportamento do fuso horário • Corrigido `$push` (agregação) para respeitar a ordem de classificação • Corrigido o bug `$currentOp` com a agregação • Corrigido problema com `readPreference` no secundário • Problema corrigido com a validação `$createIndex` é o mesmo banco de dados do comando emitido

Data	Notas da versão
	• Corrigido comportamento inconsistente para `minKey`, falhas de pesquisa `maxKey` • Corrigido problema com o operador `$size` que não funcionava com a matriz composta • Corrigido problema com a negação de `$in` com expressão regular • Corrigido problema com o comando `$distinct` executado em uma visualização • Corrigido problema com agregações e comandos de busca que classificavam os campos ausentes de modo diferente • Corrigido `$eq` para expressão regular sem verificar o tipo • Corrigido o comportamento da posição ordinal do carimbo de hora `$currentDate` • Granularidade fixa de milissegundos para `$currentDate`

Amazon DocumentDB 3.6

Notas de lançamento do mecanismo Amazon DocumentDB versão 3.6 (compatibilidade com MongoDB 3.6).

Data	Notas da versão
31 de agosto de 2026	Novo atributo Agora, o Amazon DocumentDB oferece suporte a atualizações de versões principais (MVU) no local diretamente da versão 3.6 do mecanismo para a versão 8.0. Isso elimina a necessidade de atualizaç ões de versão intermediária e permite que você atualize seus clusters 3.6, preservando os dados, as configurações e as definições do cluster existentes. Essa atualização está disponível em todas as AWS regiões em que essas versões são suportadas atualmente. Para obter mais informações, consulte Atualização da versão principal implement ada do Amazon DocumentDB no local.

Data	Notas da versão
18 de agosto de 2026	Melhorias na segurança e estabilidade Amazon DocumentDB 3.6 (versão do mecanismo: 1.0.208702) [em andamento] Esse patch de mecanismo fornece as seguintes melhorias: • Processamento aprimorado de solicitações O Amazon DocumentDB agora valida as solicitações de clientes recebidas de forma mais completa ao longo do caminho de conexão e análise de mensagens. Isso melhora a resiliência e a estabilidade em uma ampla variedade de padrões de entrada. • Fortalecendo o handshake de autenticação, o Amazon DocumentDB agora valida a troca de autenticação de forma mais completa. Isso melhora a robustez ao lidar com entradas inesperadas ou malformadas. Se você não aplicá-lo antes, o Amazon DocumentDB aplicará automaticamente esse patch durante sua próxima janela de manutenção, 30 dias após o lançamento.
27 de abril de 2026	Novos recursos Amazon DocumentDB 3.6 (Versão do mecanismo: 1.0.208692) Essa atualização fornece um patch de mecanismo opcional com as seguintes melhorias: 1. Foi adicionado suporte para `startAtOperationTime` em fluxos de alterações.

Data	Notas da versão
12 de fevereiro de 2026	Correções de bugs e outras mudanças Amazon DocumentDB 3.6 (Versão do mecanismo: 1.0.208675) 1. A `DatabaseConnectionsActiveMax` métrica agora rastreia `Connections (active)` as instâncias. Para obter mais informações, consulte Cotas de instância. 2. Melhor desempenho de autenticação SASL. 3. Correção de bug para o comando $indexStats.
20 de agosto de 2024	Novo atributo Para o Amazon DocumentDB 3.6 (versão mínimo de patch do mecanismo 1.0.208662), as atualizações do certificado CA de TLS não exigem mais a reinicialização do sistema. Para obter mais informações, consulte Atualizar seus certificados TLS do Amazon DocumentDB.
22 de julho de 2024	Novos recursos Amazon DocumentDB 3.6 (versão de patch do mecanismo 1.0.20866 2) Limitações de índice removidas nas instâncias db.r5.* e db.r6.* no Amazon DocumentDB MVU. Para obter mais informações, consulte Pré-requisitos. Correções de bugs e outras mudanças Amazon DocumentDB 3.6 (versão de patch do mecanismo 1.0.20866 2) O Amazon DocumentDB agora reconhece -NaN como um token JSON válido.

Data	Notas da versão
20 de dezembro de 2023	Outras alterações Suporte habilitado para atualização local da versão principal no Amazon DocumentDB 3.6. Para obter mais informações, consulte Atualização da versão principal implementada do Amazon DocumentDB no local.
17 de novembro de 2023	Correções de bugs e outras mudanças Amazon DocumentDB 3.6 (versão de patch do mecanismo 1.0.208570) Os nomes de variáveis locais definidos pelo usuário agora oferecem suporte a “_” (sublinhado) para operadores de projeção, como `$let` e `$filter`.
20 de setembro de 2023	Novos recursos Foi adicionado suporte para atualizações locais de versões principais no Amazon DocumentDB 3.6. Para obter mais informações, consulte Atualização da versão principal implementada do Amazon DocumentDB no local.

Data	Notas da versão
2 de fevereiro de 2023	Correções de bugs e outras mudanças Amazon DocumentDB 3.6 (versão de patch do mecanismo 1.0.20843 2) • Corrigido problema com a auditoria de DML quando habilitada durante uma workload contínua. • Corrigido um problema com a auditoria de DML quando comandos agregados com dica recebem um valor de string. • Corrigido um problema em que o comando `listCollections` não funcionava quando usuários com a função readwriteanydataba se tinham as opções AuthorizedCollections e NameOnly definidas como verdadeiras. • Corrigido problema para analisar corretamente a string numérica em um nome de campo. • Cancele cursores de longa duração quando eles estiverem afetando a coleta de resíduos.

Data	Notas da versão
9 de agosto de 2022	Novos recursos Amazon DocumentDB 3.6 (patch de motor versão 1.0.208152) • Adicionado suporte para tipo de dado Decimal128. O Decimal128 é um tipo de dados BSON suportado em todas as regiões em que o DocumentDB está disponível. Para obter mais informações, consulte Tipos de dados. • Foi adicionado suporte para auditoria de consultas DML com o Amazon CloudWatch Logs. Agora, o Amazon DocumentDB pode gravar eventos de linguagem de manipulação de dados (DML) e eventos de linguagem de definição de dados (DDL) no Amazon Logs. CloudWatch Veja este post de blog para obter mais informações. Correções de bugs e outras mudanças Amazon DocumentDB 3.6 (patch de motor versão 1.0.208152) • Agora, você pode alterar a própria senha com o privilégio `changeOwnPassword` .
7 de abril de 2022	Novos recursos Amazon DocumentDB 3.6 (versões de patch de motor 1.0.207836 e 1.0.208015) O Amazon DocumentDB Performance Insights agora está disponível para visualização. Agora, você pode armazenar sete dias de histórico de desempenho em uma janela contínua sem custo adicional. Para obter mais informações, consulte Monitorando com insights de desempenho.

Data	Notas da versão

24 de janeiro de 2022	Novos recursos
	
	Amazon DocumentDB 3.6 (patch de motor versão 1.0.207684)
	
	• O Amazon DocumentDB agora oferece um teste gratuito. Para detalhes, consulte a página de teste gratuito do Amazon DocumentDB.
	• Agora, você pode usar atributos aprimorados com a consulta Geoespacial, que inclui as seguintes APIs:
	  • `$geoWithin`
	  • `$geoIntersects`
	• Adicionado o suporte para os seguintes operadores MongoDB:
	  • `$mergeObjects`
	  • `$reduce`
	
	Para mais informações, consulte Consultar dados geoespaciais com o Amazon DocumentDB.

21 de janeiro de 2022	Novos recursos
	
	Amazon DocumentDB 3.6 (patch de motor versão 1.0.207781)
	
	• Suporte adicional para as seguintes APIs do MongoDB:
	  • `$reduce`
	  • `$mergeObjects`
	  • `$geoWithin`
	  • `$geoIntersects`

Data	Notas da versão

25 de outubro de 2021	Novos recursos
	
	Amazon DocumentDB 3.6 (patch de motor versão 1.0.207780)
	
	• Suporte adicional para as seguintes APIs do MongoDB
	  • `$literal`
	  • `$map`
	  • `$$ROOT`
	• Suporte para recursos de GeoSpatial consulta. Veja este post de blog para obter detalhes
	• Suporte para controle de acesso com perfis definidos pelo usuário. Veja este post de blog para obter detalhes
	• Driver JDBC do Amazon DocumentDB para permitir a conectividade de ferramentas de BI como o Tableau e ferramentas de consulta como o SQL Workbench
	
	Correções de bugs e outras mudanças
	
	Amazon DocumentDB 3.6 (patch de motor versão 1.0.207780)
	
	• Correção de bug para `$natural` classificar corretamente quando um `.sort()` explícito estiver presente com `$natural`
	• Correção de bug para o fluxo de alterações trabalhar com `$redact`
	• Correção de bug para `$ifNull` trabalhar com matriz vazia
	• Correção de bug para consumption/server falha excessiva de recursos quando um usuário atualmente conectado é excluído ou o privilégio desse usuário para uma atividade contínua é revogado
	• Correção de bug em `listDatabase` e na verificação de privilégi os `listCollection`
	• Solução de bug de lógica de desduplicação para elementos de múltiplas chaves

Data	Notas da versão
24 de junho de 2021	Novos recursos Amazon DocumentDB 3.6 (patch de motor versão 1.0.207117) • Agora, há suporte para instâncias r5.8xlarge e r5.16xlarge. Saiba mais no post de blog Amazon DocumentDB agora suporta instâncias r5.8xlarge e r5.16xlarge. • Agora, os clusters globais são suportados para fornecer recuperação de desastres em caso de Region-wide interrupções e permitir leituras globais de baixa latência, permitindo leituras do cluster Amazon DocumentDB mais próximo. Observe que, no momento, não há suporte para clusters globais para as regiões América do Sul (São Paulo), Europa (Milão), China (Pequim) e China (Ningxia).

Data	Notas da versão
4 de maio de 2021	Novos recursos Amazon DocumentDB 3.6 (patch de motor versão 1.0.207117) • `renameCollection` • `$zip` • `$indexOfArray` • `$reverseArray` • `$natural` • Suporte do `$hint` para atualização • Varredura de índice para `distinct` Correções de bugs e outras mudanças Amazon DocumentDB 3.6 (patch de motor versão 1.0.207117) • Uso de memória reduzido para consultas `$in` • Corrigido um vazamento de memória em índices de várias chaves • Corrigido o plano de explicação e a saída do profiler para `$out` • Adicionado um tempo limite para as operações do sistema de monitoramento interno para melhoria de confiabilidade • Corrigido um defeito que impactava os predicados de consulta passados a índices de várias chaves

Data	Notas da versão
15 de janeiro de 2021	Novos recursos Amazon DocumentDB 3.6 (versão de patch do mecanismo 1.0.206295) • Capacidade de uso de um índice com o estágio de agregação `$lookup` • Consultas `find()` com projeções que podem ser servidas na direção de um índice (consulta coberta) • Capacidade de usar `hint()` com o `findAndModify` • Otimizações de desempenho para o operador `$addToSet` • Melhorias para reduzir os tamanhos gerais dos índices • Novos operadores de agregação: `$ifNull`, `$replaceRoot` , `$setIsSubset` , `$setInstersection` , `$setUnion` e `$setEquals` • Os usuários também podem finalizar os próprios cursores sem exigir a função `KillCursor`
9 de novembro de 2020	Novos recursos Amazon DocumentDB 3.6 (versão de patch do mecanismo 1.0.206295) • Nenhum Correções de bugs e outras mudanças Amazon DocumentDB 3.6 (versão de patch do mecanismo 1.0.206295) • Nenhum

Data	Notas da versão
30 de outubro de 2020	Novos recursos Amazon DocumentDB 3.6 (versão de patch do mecanismo 1.0.206295) • Foi adicionada a capacidade de abrir um cursor de fluxo de alterações no nível do cluster (`client.watch()` ou `mongo.watch()`) e banco de dados (`db.watch()`) • Habilidade de aumentar o período de retenção do fluxo de alterações para 7 dias (antes, 24 horas) Correções de bugs e outras mudanças Amazon DocumentDB 3.6 (versão de patch do mecanismo 1.0.206295) • Várias melhorias gerais no desempenho do caso • Uma melhoria de segurança direcionada • Corrigido um problema com pulo de classificação no segundo campo de um índice composto • Habilitar índice regular para igualdade em um só campo de um índice de várias chaves (não composto) • Condição de corrida de autenticação corrigida • Corrigido o problema que causava falência pouco frequente na coleta de resíduos • Melhoria de segurança RBAC • Métrica `databaseConnectionsMax` adicionada • Melhorias de desempenho para determinadas workloads em instâncias do `r5.24xlarge`

Data	Notas da versão
22 de setembro de 2020	Novos recursos Amazon DocumentDB 3.6 (versão de patch do mecanismo 1.0.206295) • Estágio de agregação `$out` • Aumentado o número máximo de conexões e cursor por instância em até 10 vezes Correções de bugs e outras mudanças Amazon DocumentDB 3.6 (versão de patch do mecanismo 1.0.206295) • Nenhum
10 de julho de 2020	Novos recursos Amazon DocumentDB 3.6 (versão de patch do mecanismo 1.0.206295) • Cópia de capturas de tela entre regiões Correções de bugs e outras mudanças Amazon DocumentDB 3.6 (versão de patch do mecanismo 1.0.206295) • Nenhum

Data	Notas da versão
30 de junho de 2020	Novos recursos Amazon DocumentDB 3.6 (versão de patch do mecanismo 1.0.206295) • Instâncias médias T3 Correções de bugs e outras mudanças Amazon DocumentDB 3.6 (versão de patch do mecanismo 1.0.206295) • Recuperação de memória ociosa para instâncias t3 • Melhorias de autenticação • Melhoria no desempenho de autenticação SASL • Corrigido o problema `currentOp` ao exceder o máximo de operações possíveis • Corrigido o problema `killOps` para atualização e exclusão em massa • Melhorias de desempenho do `$sample` com `$match` • Suporte fixo para `$$` em caso de condição no estágio de redação • Foram corrigidas várias causas raiz recorrentes de falências • Melhorias na varredura de TTL para reduzir o IOs e a latência • Utilização de memória otimizada para `$unwind` • Condição de corrida de estatísticas de coleção corrigida com índice de queda • Condição de corrida corrigida durante a criação de índice simultâneo • Corrigida uma falha pouco frequente em `hash_search` no índice

Notas gerais de lançamento do Amazon DocumentDB

Atualizações gerais de serviços, incluindo disponibilidade regional, integrações de ferramentas e recursos entre versões.

Data	Notas da versão
15 de setembro de 2026	Novo atributo O tipo de instância sem servidor do Amazon DocumentDB agora está disponível nas versões de mecanismo do Amazon DocumentDB 5.0 e 8.0 nas seguintes regiões: • Ásia-Pacífico (Osaka) • Ásia-Pacífico (Jacarta) • Ásia-Pacífico (Malásia) • Ásia-Pacífico (Melbourne) • Ásia-Pacífico (Tailândia) • Oeste do Canadá (Calgary) • Europa (Zurique) • Europa (Estocolmo) • Israel (Tel Aviv) • México (Centro) Para obter mais informações, consulte Classes de instância compatíveis por região e Uso do Amazon DocumentDB Sem Servidor.
29 de abril de 2026	Novo atributo O Amazon DocumentDB agora está disponível na região Oeste do Canadá (Calgary). Veja este post de blog para obter mais informações.
17 de abril de 2026	Novo atributo Agora está disponível uma escalabilidade mais rápida no DocumentDB sem servidor.

Data	Notas da versão
11 de fevereiro de 2026	Novo atributo O Amazon DocumentDB agora está disponível na região Ásia-Pacífico (Melbourne). Veja este post de blog para obter mais informações. O Amazon DocumentDB agora está disponível na região da Europa (Zurique). Veja este post de blog para obter mais informações.
08 de janeiro de 2026	Novo atributo O Amazon DocumentDB agora está disponível na região Ásia-Pacífico (Jacarta). Veja este post de blog para obter mais informações.
13 de outubro de 2025	Novo atributo Agora, o Amazon DocumentDB oferece suporte a instâncias R6G e T4G na região Ásia-Pacífico (Hyderabad). Para obter mais informações, consulte Classes de instância compatíveis por região.
7 de outubro de 2025	Novo atributo Agora, o Amazon DocumentDB está disponível nas seguintes regiões: • Ásia-Pacífico (Malásia) • Ásia-Pacífico (Osaka) • Ásia-Pacífico (Tailândia) • México (Centro) Veja este post de blog para obter mais informações.

Data	Notas da versão
26 de setembro de 2025	Novos recursos Agora, o Amazon DocumentDB oferece suporte à cópia e ao compartil hamento de snapshots entre regiões e entre contas. Para obter mais informações, consulte Copiando instantâneos em Regiões da AWS e Compartilhamento de snapshots de cluster do Amazon DocumentDB.
18 de junho de 2025	Novo atributo O Amazon DocumentDB agora está disponível na região de Israel (Tel Aviv).
8 de maio de 2025	Novo atributo O Amazon DocumentDB agora está disponível na região da Europa (Estocolmo). Veja este post de blog para obter mais informações.
2 de abril de 2025	Correções de bugs e outras mudanças Foi corrigido um erro na criação de índices vetoriais com índices de fundo.
6 de fevereiro de 2025	Novo atributo O Amazon DocumentDB agora está integrado com o. AWS Toolkit for Visual Studio Code Para obter mais informações, consulte esta postagem O que há de novo no blog e consulte Trabalho com o Amazon DocumentDB no Kit de Ferramentas no Guia do usuário do AWS Toolkit for Visual Studio Code .

Data	Notas da versão
22 de outubro de 2024	Novo atributo O Amazon DocumentDB agora oferece suporte a métricas de throughput de rede de armazenamento `StorageNetworkReceiveThroughput`, `StorageNetworkTransmitThroughput` e `StorageNetworkThroughput`. Para obter mais informações, consulte Avaliação do uso de instâncias do Amazon DocumentDB com métricas CloudWatch.
18 de setembro de 2024	Novo atributo O Amazon DocumentDB agora está disponível na região da África (Cidade do Cabo). Veja este post de blog para obter mais informações. O Amazon DocumentDB agora está disponível na região da Europa (Espanha). Veja este post de blog para obter mais informações.
25 de junho de 2024	Novo atributo A autenticação com ARNs AWS do IAM está disponível em clusters 5.0 baseados em instâncias do Amazon DocumentDB em todas as regiões suportadas. Para obter mais informações, consulte Autenticação usando identidade do IAM.
3 de abril de 2024	Novo atributo O Amazon DocumentDB agora está disponível na região do Oriente Médio (EAU). Veja este post de blog para obter mais informações. Correções de bugs e outras mudanças • Corrigido um erro ao invocar `db.coll.stats()` no shell mongo versão 1.7 e posterior. • Corrigido um problema de vazamento de memória para consultas de fluxo de alterações que contêm `$regex` como parte do mesmo pipeline de agregação.

Data	Notas da versão
30 de janeiro de 2024	Novos recursos Cluster globais do Amazon DocumentDB Agora, os clusters globais estão disponíveis nas duas AWS GovCloud (US) regiões: AWS GovCloud (US-East) e AWS GovCloud (US-West).
13 de dezembro de 2023	Novos recursos Foi adicionado suporte para conectividade EC2 com 1 clique. Para obter mais informações, consulte Conecte usando o Amazon EC2.
17 de novembro de 2023	Novos recursos O Amazon DocumentDB agora está disponível na região AWS GovCloud (US-East). Veja este post de blog para obter mais informações.
25 de setembro de 2023	Novos recursos O Amazon DocumentDB agora está disponível na região Ásia-Pacífico (Hong Kong). Veja este post de blog para obter mais informações.
11 de setembro de 2023	Novos recursos O Amazon DocumentDB agora está disponível na região Ásia-Pacífico (Hyderabad). Veja este post de blog para obter mais informações.
29 de abril de 2022	Novos recursos O Amazon DocumentDB agora está disponível na região da China (Pequim). Veja este post de blog para obter mais informações.
16 de março de 2022	Novos recursos O Amazon DocumentDB agora está disponível na região da Europa (Milão). Veja este post de blog para obter mais informações.

Data	Notas da versão
8 de fevereiro de 2022	Novos recursos As instâncias R6G e T4G do Amazon DocumentDB agora estão disponíveis nas regiões Ásia Pacífico, América do Sul e Europa. Veja este post de blog para obter mais informações.
4 de maio de 2021	Novos recursos Veja todos os novos atributos neste post de blog.
9 de novembro de 2020	Novos recursos Veja todos os novos atributos neste post de blog.
30 de outubro de 2020	Novos recursos Veja todos os novos atributos neste post de blog.
22 de setembro de 2020	Novos recursos Veja todos os novos atributos neste post de blog.
10 de julho de 2020	Novos recursos Veja todos os novos atributos neste post de blog.
30 de junho de 2020	Novos recursos Veja todos os novos atributos neste post de blog.

Notas de lançamento dos clusters elásticos do Amazon DocumentDB

Notas de lançamento para clusters elásticos do Amazon DocumentDB.

Data	Notas da versão
28 de janeiro de 2025	Novo atributo

Data	Notas da versão
	Para conectividade com um clique, o Amazon DocumentDB agora está integrado com clusters elásticos AWS CloudShell e baseados em instâncias. Para obter mais informações, consulte um dos itens a seguir: • Conceitos básicos do Amazon DocumentDB • Como iniciar com clusters elásticos do Amazon DocumentDB • Postagem O que há de novo no blog • Postagem de instruções técnicas no blog
12 de novembro de 2024	Novos recursos Os clusters elásticos do Amazon DocumentDB agora oferecem suporte a índices de segundo plano.
1.º de novembro de 2024	Novo atributo O Amazon DocumentDB agora oferece suporte a ações de manutenção de clusters elásticos. Para obter mais informações, consulte Manutenção dos clusters elásticos do Amazon DocumentDB.
8 de agosto de 2024	Novo atributo Os clusters elásticos do Amazon DocumentDB agora estão disponíveis nas regiões Ásia-Pacífico (Hong Kong), Canadá (Central) e Europa (Paris). Para obter mais informações, consulte Disponibilidade regional para clusters elásticos.
8 de julho de 2024	Novo atributo Os clusters elásticos do Amazon DocumentDB agora estão disponíveis na região da Europa (Milão). Para obter mais informações, consulte Disponibilidade regional para clusters elásticos.

Data	Notas da versão
22 de fevereiro de 2024	Novos recursos Clusters elásticos do Amazon DocumentDB Os clusters elásticos do Amazon DocumentDB agora suportam as seguintes operações: • Réplicas de instâncias de fragmento secundárias legíveis: para obter mais informações, consulte a etapa 5b de Etapa 1: Criar um cluster do ElastiCache. • Start/stop cluster - para obter mais informações, consulteInterromper e iniciar um cluster elástico do Amazon DocumentDB. • Instâncias de fragmento configuráveis: para obter mais informações, consulte a etapa 5b de Etapa 1: Criar um cluster do ElastiCache. • Backups automáticos para snapshots - para obter mais informaçõ es, consulte Gerenciar um backup automático de snapshot de cluster elástico. • Copiar snapshot - para obter mais informações, consulte Copiar um snapshot de cluster elástico.
30 de janeiro de 2024	Novos recursos Clusters elásticos do Amazon DocumentDB Os clusters elásticos do Amazon DocumentDB agora estão disponíveis nas seguintes regiões: • Ásia-Pacífico (Mumbai) • Ásia-Pacífico (Seul) • América do Sul (São Paulo) • Europa (Londres) Para obter mais informações, consulte Disponibilidade de regiões e versões do cluster elástico.

Data	Notas da versão

3 de agosto de 2023	Novos recursos
	
	Amazon DocumentDB Elastic Clusters
	
	• O Amazon DocumentDB Elastic Clusters agora suporta as seguintes operações:
	  • `top`
	  • `collStats`
	  • `hint`
	  • `dataSize`
	
	  Consulte APIs, operações e tipos de dados do MongoDB compatíveis no Amazon DocumentDB para ver a lista completa de comandos e operações com suporte.
	• Índices Time-to-Live (TTL) agora têm suporte.
	• O `hints` de índice agora é compatível com expressões de índice.

22 de março de 2023	Novos recursos Os clusters elásticos do Amazon DocumentDB agora estão disponíveis nas regiões Ásia-Pacífico (Cingapura), Ásia-Pacífico (Sydney) e Ásia-Pacífico (Tóquio). Para obter mais informações, consulte Disponibilidade de regiões e versões do cluster elástico.

Data	Notas da versão
30 de novembro de 2022	Novos recursos Amazon DocumentDB Elastic Clusters Os clusters elásticos do Amazon DocumentDB são um novo tipo de cluster do Amazon DocumentDB que permite que os usuários usem as APIs de fragmentação do MongoDB para aumentar a escala horizontalmente do cluster. Os clusters elásticos lidam com praticamente qualquer número de leituras e gravações com petabytes de capacidade de armazenamento, distribuindo os dados e a computação em várias instâncias e volumes de computação subjacentes. Para saber mais, consulte Como usar clusters elásticos do Amazon DocumentDB.

Histórico do documento do Guia do desenvolvedor do Amazon DocumentDB

- Versão da API: 31-10-2014
- Última atualização da documentação: 27 de maio de 2026

A tabela a seguir descreve a documentação desta versão do Guia do desenvolvedor do Amazon DocumentDB.

Alteração	Descrição	Data
Novo guia do desenvolvedor do Amazon DocumentDB	O Guia do desenvolvedor do Amazon DocumentDB agora está publicado em uma nova URL. A referência da API está disponível separadamente no Amazon DocumentDB API Reference.	27 de maio de 2026
AWS atualização de política gerenciada - mudança de política	O Amazon DocumentDB atualiza as políticas de acesso total para clusters elásticos.	11 de fevereiro de 2025
AWS atualização de política gerenciada - mudança de política	O Amazon DocumentDB atualiza as políticas de acesso total para clusters elásticos.	21 de fevereiro de 2024
AWS atualização de política gerenciada - mudança de política	O Amazon DocumentD B atualiza as políticas de somente leitura e de acesso total para clusters elásticos.	21 de junho de 2023
AWS atualização de política gerenciada - nova política	O Amazon DocumentDB apresenta uma nova política de somente leitura para clusters elásticos.	8 de junho de 2023

AWS atualização de política gerenciada - nova política	O Amazon DocumentDB apresenta uma nova política de acesso total para clusters elásticos.	5 de junho de 2023
Compatibilidade do MongoDB 5.0	O Amazon DocumentDB agora é compatível com a versão 5.0 do MongoDB.	1 de março de 2023
Atualização da política	Para oferecer suporte ao recurso de cluster elástico do Amazon DocumentDB, a AmazonDocDBConsole FullAccess política é atualizada e a AmazonDocDB-ElasticServiceRolePolicy é introduzida.	30 de novembro de 2022
Clusters elásticos	Foram adicionados atributos de clusters elásticos do Amazon DocumentDB que suportam fragmentação baseada em hash para particionar dados em um sistema de armazenamento distribuído.	30 de novembro de 2022
Clusters globais	Foi adicionada documentação sobre como usar os clusters globais.	2 de junho de 2021
Assinaturas de eventos	Foi adicionada a documentação de assinatura do evento.	26 de março de 2021

Atualizações da versão 3.6	Melhorias documentadas na versão 3.6 em controles de acesso baseados em funções, operadores de agregação e desempenho.	15 de janeiro de 2021
Compatibilidade do MongoDB 4.0	O Amazon DocumentDB agora é compatível com a versão 4.0 do MongoDB.	9 de novembro de 2020
Guias para começar	Novos guias de introduçã o para começar a usar o Amazon DocumentDB usando Amazon EC2 AWS Cloud9, Robo3T ou Studio3T.	15 de agosto de 2020
Zonas de disponibilidade adicionais compatíveis	O Amazon DocumentDB incluiu suporte para uma zona de disponibilidade adicional na Ásia-Pacífico (Seul) (ap-north east-2).	14 de julho de 2020
Adicionamos suporte à cópia de snapshots entre regiões.	O Amazon DocumentDB adicionou suporte para copiar snapshots de cluster entre Regiões da AWS. Para ter mais informações, consulte o tópico sobre como Copiar snapshots entre regiões.	10 de julho de 2020

Adicionamos suporte para classe de instância T3.	Foi adicionado suporte para tipos de instância T3 em todas as regiões que oferecem suporte ao Amazon DocumentDB. Para obter mais informações, consulte Classes de instância compatíveis por região e Especificações de classe de instância.	30 de junho de 2020
Suporte adicionado para AWS GovCloud (US).	O Amazon DocumentDB agora está disponível na AWS GovCloud (US) região (us-gov-west-1).	29 de junho de 2020
Foram adicionadas 16 novas CloudWatch métricas.	O Amazon DocumentDB adicionou suporte para 16 novas CloudWatch métricas da Amazon. Para obter mais informações, consulte Monitorando o Amazon DocumentDB com CloudWatch.	23 de junho de 2020
Adicionamos suporte para caracteres nulos e operador $regex.	O Amazon DocumentDB adicionou suporte para caracteres nulos em strings e a capacidade de usar um índice para $regex. Para visualizar as APIs do MongoDB e os recursos de pipeline de agregação compatíveis com o Amazon DocumentDB, consulte Diferenças funcionais com o MongoDB.	22 de junho de 2020

Adicionamos suporte para recursos melhorados de indexação de várias chaves.	O Amazon DocumentDB adicionou suporte para recursos melhorados de indexação de várias chaves, que incluem indexação de matrizes maiores que 2.048 bytes e a capacidade de criar um índice composto de várias chaves com várias chaves na mesma matriz. Para obter mais informações, consulte Diferenças funcionais com relação ao MongoDB.	23 de abril de 2020
Foi adicionado suporte para proteção contra exclusão de uma pilha do Amazon DocumentDB CloudFormation .	O Amazon DocumentDB adicionou suporte para habilitar a proteção contra exclusão ao criar uma pilha do Amazon CloudFormation DocumentDB.	20 de abril de 2020
Adicionamos suporte para o controle de acesso baseado em função.	O Amazon DocumentDB adicionou suporte para controle de acesso baseado em função usando funções integradas.	26 de março de 2020
Adicionamos suporte para uma zona de disponibilidade adicional no Canadá (Central) (ca-central-1).	Agora, o Amazon DocumentDB está disponível na região Canadá (Central) (ca-central-1) com instâncias de classe R5 e 3 zonas de disponibilidade.	26 de março de 2020

Adicionamos suporte para três APIs adicionais do MongoDB.	O Amazon DocumentDB adicionou suporte às APIs `$dateFromString` e `executionStats` do MongoDB.	23 de março de 2020
Adição de suporte para cinco APIs do MongoDB adicionais.	O Amazon DocumentDB adicionou suporte às APIs `$objectToArray` , `$arrayToObject` , `$slice`, `$mod` e `$range` e do MongoDB.	6 de fevereiro de 2020
Suporte incluído para o Canadá (Central).	Agora, o Amazon DocumentDB está disponível na região Canadá (Central) (ca-central-1) com instâncias de classe R5.	11 de dezembro de 2019
Suporte adicionado para ChangeStreamLogSize.	O Amazon DocumentDB adicionou suporte para `ChangeStreamLogSize` para métricas do CloudWatch.	22 de novembro de 2019
Suporte adicionado para a região Europa (Paris)	O Amazon DocumentDB agora está disponível na região da Europa (Paris) (eu-west-3) com instâncias da classe R5.	30 de outubro de 2019
Adicionado suporte para a região Ásia-Pacífico (Mumbai)	O Amazon DocumentDB agora está disponível na região Ásia-Pacífico (Mumbai) (ap-south-1) com instâncias da classe R5.	17 de outubro de 2019

Adicionamos o suporte para três APIs do MongoDB adicionais	O Amazon DocumentDB adicionou suporte às APIs `$addFields` , `$concatArrays` e `$lookup` do MongoDB.	16 de outubro de 2019
Adicionado suporte para a região Ásia-Pacífico (Cingapura)	O Amazon DocumentDB agora está disponível na região Ásia-Pacífico (Cingapura) (ap-southeast-1) com instâncias da classe R5.	14 de outubro de 2019
Adicionamos novo documento para atualizar os certificados TLS	Adicionadas instruções para atualizar certificados de autoridade de certificação (CA) para usar o novo certificado da autoridade de certificação na criação de conexões TLS.	2 de outubro de 2019
Adicionamos suporte à API para certificados	Amazon DocumentDB um novo tipo de dados de certificado para instâncias. Para obter mais informações, consulte DBInstance.	1 de outubro de 2019
Suporte à criação de perfis de consulta	O Amazon DocumentDB adicionou a capacidade de criar o perfil de operações compatíveis nas instâncias e bancos de dados do cluster.	19 de agosto de 2019

Adição do terceiro AZ na Ásia-Pacífico (Tóquio)	O Amazon DocumentDB adicionou uma terceira zona de disponibilidade (AZ) para suas instâncias computacionais na Ásia-Pacífico (Tóquio).	9 de agosto de 2019
Suporte a APIs do Mongo adicionais	Foi adicionado suporte para recursos adicionais do pipeline de agregação que incluem os operadores de agregação `$in`, `$isoWeek`, `$isoWeekYear` , `$isoDayOfWeek` e `$dateToString` e o estágio de agregação `$addToSet` . O Amazon DocumentDB também adicionou suporte ao comando `top()` para diagnóstico em nível de coleção e a capacidade de modificar o parâmetro `expireAfterSeconds` para índices TTL usando o comando `collMod()` .	31 de julho de 2019
Foi adicionado suporte para a região da Europa (Londres)	O Amazon DocumentDB já está disponível na região da Europa (Londres) (eu-west-2) com instâncias de classe R5.	18 de julho de 2019
Adicionamoss exemplos de código	Adicionamoss exemplos de código em R e Ruby para a conexão de forma programática ao Amazon DocumentDB.	17 de julho de 2019

Adicionadas práticas recomendadas	Adicionada uma prática recomendada para ajudar você a gerenciar seus custos do Amazon DocumentDB.	17 de julho de 2019
Suporte à interrupção e ao início de um cluster	O Amazon DocumentDB adicionou suporte à interrupção e ao início de clusters para gerenciar os custos dos ambientes de teste e desenvolvimento.	1 de julho de 2019
Suporte à proteção contra exclusão do cluster	Para proteger seus clusters contra a exclusão acidental, o Amazon DocumentDB adicionou proteção contra exclusão. Para obter mais informações, consulte os seguintes tópicos: Criar um cluster do Amazon DocumentDB, Modificar um cluster do Amazon DocumentDB, Excluir um cluster do Amazon DocumentDB e `DeletionProtection` no tópico da API DBCluster.	1 de julho de 2019
Atualização das diferenças funcionais	Adição de transações implícitas a diferenças funcionais.	26 de junho de 2019
Adição de diferenças funcionais	Adição de uma observação sobre armazenamento e compressão de índice no Amazon DocumentDB.	13 de junho de 2019

Região adicional suportada	O Amazon DocumentDB já está disponível na região da Ásia-Pacífico (Sydney) (ap-southeast-2) com instâncias de classe R5.	5 de junho de 2019
Classe de instância R5 suportada em outras regiões	Foi adicionado suporte à classe de instância R5 para 4 regiões adicionais: Leste dos EUA (Ohio), Leste dos EUA (Norte da Virgínia), Oeste dos EUA (Oregon) e UE (Irlanda) . Com essa mudança, as instâncias R5 são suportadas em todas as regiões que oferecem suporte ao Amazon DocumentDB.	17 de maio de 2019
Regiões adicionais suportadas	Foi adicionado suporte para duas regiões adicionais, Ásia-Pacífico (Tóquio) (ap-northeast-1) e Ásia-Pacífico (Seul) (ap-northeast-2) com classes de instância R5. Para obter mais informações, consulte Classes de instância compatíveis por região e Especificações de classe de instância.	8 de maio de 2019
Adição de mais exemplos de código de conexão	Adição de exemplos de código em Java e C# para conexão com o Amazon DocumentDB.	24 de abril de 2019

Suporte adicional à API do Mongo	Adicionamos suporte para sete operadores de string de agregação (`$indexOfBytes` , `$indexOfCP` , `$strLenBytes` , `$strLenCP` , `$toLower`, `$toUpper` e `$split`), nove operadores de data/hora (`$dayOfYear` , `$dayOfMonth` , `$dayOfWeek` , `$year`, `$month`, `$hour`, `$minute`, `$second` e `$millisecond`) e o estágio de pipeline de agregação `$sample`.	4 de abril de 2019
Adicionamoss exemplos de código de conexão	Foram adicionados exemplos de código em Python Node.js, PHP e Go para conexão com o Amazon DocumentDB.	21 de março de 2019
Suporte para a região de Frankfurt e instâncias R5	Adicionamos suporte para a Europa (Frankfurt) (eu-central-1) com classes de instância R5. Para obter mais informações, consulte Classes de instância compatíveis por região e Especificações de classe de instância.	13 de março de 2019

Suporte a operadores de pipeline de agregação	Adicionamos suporte para novos operadores de string de agregação (`$concat`, `$substr`, `$substrBytes`, `$substrCP`, `$strcasecmp`), um operador de agregação de matriz (`$size`), um operador de acumulador do grupo de agregação (`$push`) e estágios de agregação (`$redact` e `$indexStats`). Também adicionamos suporte para operadores de matriz posicional (`$[]` e `$[<identifier>]`) e `hint()`.	28 de fevereiro de 2019
Atualizações do mecanismo	Adicionamos documentação para determinar as modificações de cluster pendentes e atualizar a versão do mecanismo do cluster.	15 de fevereiro de 2019
Auditoria de eventos	Foi adicionado suporte para auditar eventos de banco de dados com o Amazon CloudWatch Logs.	12 de fevereiro de 2019
Início rápido	Foi adicionado um tópico de início rápido para ajudar você a começar a usar CloudFormation o Amazon DocumentDB com facilidade.	11 de janeiro de 2019

Versão pública	Essa é a versão pública inicial do Amazon DocumentDB (compatível com MongoDB). Esse lançamento inclui o Guia do desenvolvedor e a Referência de API de gerenciamento de recursos integrada.	9 de janeiro de 2019

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.