aws

Panduan Developer

Amazon DocumentDB



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

Amazon DocumentDB: Panduan Developer

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

Merek dagang dan tampilan dagang Amazon tidak boleh digunakan sehubungan dengan produk atau layanan apa pun yang bukan milik Amazon, dengan cara apa pun yang dapat menyebabkan kebingungan di antara pelanggan, atau dengan cara apa pun yang merendahkan atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan hak milik masing-masing pemiliknya, yang mungkin atau mungkin tidak terafiliasi, terkait dengan, atau disponsori oleh Amazon.

Table of Contents

Apa itu Amazon DocumentDB 1
Ikhtisar 1
klaster 3
Contoh 4
Wilayah dan AZ 7
Wilayah 7
Zona ketersediaan 8
Harga 10
Uji coba gratis 11
Memantau 11
Antarmuka 12
Konsol Manajemen AWS 12
AWS CLI 12
Driver MongoDB 12
Apa selanjutnya? 12
Cara kerjanya 13
Versi Amazon DocumentDB 13
Titik akhir Amazon DocumentDB 16
Dukungan TLS 19
Penyimpanan Amazon DocumentDB 19
Replikasi Amazon DocumentDB 20
Keandalan Amazon DocumentDB 21
Baca opsi preferensi 22
TTL menghapus 26
Sumber daya yang dapat ditagih 27
Apa itu database dokumen? 30
Kasus penggunaan 31
Memahami dokumen 32
Bekerja dengan dokumen 38
Panduan Memulai 50
Prasyarat 50
Langkah 1: Buat cluster 52
Langkah 2: Connect ke cluster Anda 55
Langkah 3: Masukkan dan kueri data 56

Langkah 4: Jelajahi ... 58
Fitur dan Konfigurasi ... 59
Status mesin ... 59
Jenis cluster ... 59
Fitur ... 60
Tipe instans ... 61
AWS integrasi ... 62
Mulai cepat menggunakan CloudFormation ... 64
Prasyarat ... 65
Izin IAM yang Diperlukan ... 65
Pasangan Kunci Amazon EC2 ... 65
Meluncurkan Amazon DocumentDB CloudFormation tumpukan ... 65
Mengakses cluster Amazon DocumentDB ... 70
Perlindungan penghentian dan perlindungan penghapusan ... 71
Kompatibilitas MongoDB ... 72
MongoDB 8.0 kompatibilitas ... 72
Apa yang baru di Amazon DocumentDB 8.0 ... 72
Memulai Amazon DocumentDB ... 73
Tingkatkan atau migrasi ke Amazon DocumentDB 5.0 atau 8.0 ... 74
Perbedaan fungsional ... 74
Kompatibilitas MongoDB 5.0 ... 74
Apa yang baru di Amazon DocumentDB 5.0 ... 75
Apa yang baru di Amazon DocumentDB 5.0.1 ... 75
Memulai Amazon DocumentDB 5.0 ... 73
Tingkatkan atau migrasi ke Amazon DocumentDB 5.0 ... 74
Perbedaan fungsional ... 74
Kompatibilitas MongoDB 4.0 ... 78
Fitur Amazon DocumentDB 4.0 ... 79
Memulai Amazon DocumentDB 4.0 ... 80
Tingkatkan atau migrasi ke Amazon DocumentDB 4.0 ... 81
Perbedaan fungsional ... 81
Transaksi ... 83
Persyaratan ... 83
Praktik terbaik ... 84
Batasan ... 84
Pemantauan dan diagnostik ... 85

Tingkat isolasi transaksi ... 86
Kasus penggunaan ... 86
Multi-Statement Transaksi ... 86
Multi-collection transaksi ... 88
Contoh API transaksi untuk API callback ... 90
Contoh API transaksi untuk API inti ... 90
Perintah yang Didukung ... 124
Kemampuan yang tidak didukung ... 125
Sesi ... 125
Konsistensi kausal ... 126
Penulisan yang dapat dicoba lagi ... 127
Kesalahan transaksi ... 127
Praktik terbaik ... 128
Pedoman operasional dasar ... 128
Ukuran instans ... 129
Bekerja dengan indeks ... 131
Membangun Indeks ... 131
Selektivitas indeks ... 131
Dampak indeks pada penulisan data ... 132
Mengidentifikasi indeks yang hilang ... 132
Mengidentifikasi indeks yang tidak digunakan ... 133
Praktik terbaik keamanan ... 133
Optimalisasi biaya ... 134
Menggunakan metrik untuk mengidentifikasi masalah performa ... 134
Melihat metrik performa ... 135
Mengatur CloudWatch alarm ... 135
Mengevaluasi metrik performa ... 135
Mengevaluasi penggunaan instans Amazon DocumentDB dengan metrik CloudWatch ... 137
Menyetel kueri ... 138
TTL dan beban kerja time series ... 139
Migrasi ... 139
Bekerja dengan kelompok parameter cluster ... 140
Kueri pipa agregasi ... 140
`BatchInsert` dan `BatchUpdate` ... 140
Perbedaan fungsional dengan MongoDB ... 141
Manfaat fungsional Amazon DocumentDB ... 141

Transaksi implisit 141
Perbedaan fungsional yang diperbarui 142
Pengindeksan array 143
Multi-key indeks 144
Karakter nol dalam string 145
Role-based kontrol akses 145
`$regex pengindeksan` 145
Proyeksi untuk dokumen bersarang 146
Perbedaan fungsional dengan MongoDB 146
`$ VectorSearch` operator 147
`OpCountersCommand` 147
Database dan koleksi admin 147
`CursorMaxTimems` 147
jelaskan() 147
Indeks membangun 148
Pencarian dengan kunci kosong di jalur 148
MongoDB API, operasi, dan tipe data 148
`utilitas mongodump dan mongorestore` 148
Pemesanan hasil 149
Penulisan yang dapat dicoba lagi 149
Indeks jarang 150
`Menggunakan $elemMatch dalam ekspresi $all` 150
Dolar ($) dan titik (.) dalam nama bidang 151
`$lookup` 151
`$ penyortiran alami` dan terbalik 155
API MongoDB yang didukung, operasi, dan tipe data 156
Perintah basis data 157
Perintah administratif 157
Agregasi 159
Autentikasi 159
Perintah diagnostik 159
Operasi kueri dan tulis 160
Perintah manajemen peran 161
Perintah sesi 162
Manajemen pengguna 163
Perintah sharding 164

Operator kueri dan proyeksi 166
Operator Array 166
Operator Bitwise 166
Operator komentar 167
Operator perbandingan 167
Operator elemen 168
Operator kueri evaluasi 168
Operator logis 168
Operator proyeksi 169
Perbarui operator 169
Operator array 169
Operator Bitwise 170
Operator lapangan 170
Perbarui pengubah 171
Geospasial 171
Penentu geometri 171
Penyeleksi kueri 172
Metode kursor 173
Operator pipa agregasi 175
Ekspresi akumulator 176
Operator aritmatika 177
Operator array 178
Operator Boolean 179
Operator perbandingan 180
Operator ekspresi bersyarat 180
Operator tipe data 181
Operator ukuran data 181
Operator tanggal 181
Operator literal 183
Gabungkan operator 183
Operator alami 183
Tetapkan operator 183
Operator panggung 184
Operator String 186
Variabel sistem 188
Operator pencarian teks 188

Jenis operator konversi ... 188
Operator variabel ... 189
Operator lain-lain ... 189
Jenis Data ... 190
Indeks dan properti indeks ... 191
Indeks ... 191
Properti indeks ... 192
Operator pipa agregasi ... 193
$$DESCEND ... 198
$$KEEP ... 201
$$PRUNE ... 203
$ AKAR ... 206
$abs ... 210
$add ... 212
$ tambahkan ToSet ... 215
$addFields ... 217
$ semua ElementsTrue ... 220
$and ... 222
$ apa saja ElementTrue ... 225
$ array ElemAt ... 228
$ array ToObject ... 230
$avg ... 233
$bucket ... 235
$ceil ... 238
$ ChangeStream ... 240
$cmp ... 243
$collStats ... 246
$concat ... 249
$concatArrays ... 251
$cond ... 254
$convert ... 258
$count ... 261
$currentOp ... 263
$ DateAdd ... 265
$ DateDiff ... 269
$ tanggal FromString ... 272

$ DateSubtract 276
$ tanggal ToString 279
$ DateTrunc 282
$ hari OfMonth 285
$ hari OfWeek 288
$ hari OfYear 290
$divide 293
$eq 295
$exp 297
$filter 300
$first 303
$floor 306
$geoNear 308
$group 311
$gt 314
$gte 317
$hour 319
$ifNull 323
$in 325
$ indeks OfArray 328
$ indeks OfBytes 330
$indexOfCP 332
$indexStats 334
$isArray 337
$ iso DayOfWeek 340
$isoWeek 343
$ iso WeekYear 345
$last 347
$let 351
$limit 354
$literal 357
$ln 360
$log 363
$log10 365
$lt 368
$lte 370

$lookup 373
$ltrim 376
$map 379
$match 382
$max 384
$meta 386
$ bergabung 389
$mergeObjects 392
$millisecond 395
$min 397
$minute 400
$mod 403
$month 405
$multiply 408
$natural 411
$ne 413
$not 416
$ objek ToArray 418
$or 423
$out 425
$pow 428
$push 430
$project 433
$ rand 435
$range 437
$redact 441
$reduce 444
$RegexFind 447
$ regex FindAll 449
$RegexMatch 453
$ GantiSemua 456
$ GantiSatu 460
$replaceRoot 464
$ GantiDengan 466
$reverseArray 469
$rtrim 472

$sample 475
$ pencarian 477
$second 480
$set 481
$setDifference 484
$setEquals 487
$setIntersection 490
$ set IsSubset 492
$setUnion 494
$skip 497
$slice 500
$size 502
$sort 504
$split 506
$sqrt 508
$ str LenBytes 511
$strLenCP 514
$strcasecmp 516
$substr 519
$substrBytes 521
$substrCP 524
$subtract 527
$sum 530
$switch 533
$toBool 536
$toDate 539
$toDecimal 541
$toDouble 543
$toInt 546
$toLong 548
$toLower 551
$ untuk ObjectId 553
$toString 555
$toUpper 557
$trim 559
$type 561

$unset ... 563
$unwind ... 566
$ VectorSearch ... 568
$week ... 572
$year ... 574
$zip ... 576
Geospasial ... 580
$geometry ... 580
$geointersects ... 584
$geowithin ... 588
$maxDistance ... 592
$minDistance ... 594
$near ... 598
$nearSphere ... 602
Operator kueri dan proyeksi ... 605
$ ... 606
$ semua ... 608
$and ... 610
$ bit AllClear ... 613
$ bit AllSet ... 615
$ bit AnyClear ... 617
$ bit AnySet ... 619
$comment ... 621
$elemMatch ... 623
$eq ... 626
$exists ... 628
$expr ... 630
$gt ... 632
$gte ... 634
$in ... 636
$jsonSchema ... 638
$lt ... 641
$mod ... 643
$lte ... 645
$meta ... 647
$ne ... 649

$nin ... 651
$nor ... 654
$not ... 656
$or ... 658
$regex ... 661
$slice ... 663
$size ... 665
$text ... 667
$type ... 669
Perbarui operator ... 671
$ ... 672
$[] ... 674
$ [<pengidentifikasi>] ... 677
$ tambahkan ToSet ... 679
$bit ... 681
$currentDate ... 684
$each ... 686
$inc ... 688
$max ... 690
$min ... 692
$mul ... 694
$pop ... 697
$position ... 699
$pull ... 702
$pullAll ... 705
$push ... 707
$rename ... 709
$set ... 712
$ set OnInsert ... 714
$slice ... 718
$sort ... 720
$unset ... 723
AI Generatif ... 726
SageMaker Kanvas ... 726
Cara membuat model MLtanpa kode dengan SageMaker AI Canvas ... 726
Mengkonfigurasi domain SageMaker AI dan profil pengguna ... 727

Mengkonfigurasi izin akses IAM untuk Amazon DocumentDB dan AI Canvas SageMaker ... 727
Membuat pengguna database dan peran untuk SageMaker AI Canvas ... 728
Wilayah yang tersedia ... 728
Pencarian Vektor ... 729
Memasukkan vektor ... 729
Membuat indeks vektor ... 730
Mendapatkan definisi indeks ... 734
Menanyakan vektor ... 735
Fitur dan keterbatasan ... 740
Praktik terbaik ... 741
Migrasi ke DocumentDB ... 742
Panduan memulai cepat ... 742
Siapkan sumber DMS ... 742
Pengaturan DMS ... 743
Aktifkan kompresi DocumentDB ... 743
Buat tugas replikasi ... 743
Memantau kemajuan ... 744
Informasi tambahan ... 744
Runbook migrasi ... 744
Kompatibilitas ... 745
Penemuan beban kerja ... 748
Migrasi indeks ... 752
Migrasi pengguna ... 753
Migrasi data ... 755
Memantau ... 766
Validasi ... 769
Migrasi dari Couchbase Server ... 774
Pengantar ... 774
Perbandingan dengan Amazon DocumentDB ... 774
Penemuan ... 777
Perencanaan ... 782
Migrasi ... 785
Validasi ... 793
Upgrade DocumentDB ... 797
Tanggal dukungan versi mesin ... 797
Kalender rilis untuk Amazon DocumentDB versi utama ... 797

Kalender rilis untuk Amazon DocumentDB versi minor ... 798
Memutakhirkan versi mesin utama ... 798
Jalur pemutakhiran yang didukung ... 799
Prasyarat ... 800
Praktik terbaik ... 801
Apa yang berubah setelah memutakhirkan dari Amazon DocumentDB 5.0 ke 8.0 ... 804
Post-upgrade pertimbangan untuk cluster ditingkatkan dari 3.6 atau 4.0 ... 805
Melakukan upgrade ... 805
Pemecahan masalah ... 808
Memutakhirkan versi mesin minor ... 808
Sebelum melakukan upgrade versi minor ... 808
Melakukan upgrade versi minor ... 809
Memverifikasi peningkatan versi minor ... 810
Upgrade menggunakan AWS DMS ... 810
Langkah 1: Aktifkan aliran perubahan ... 811
Langkah 2: Ubah durasi retensi aliran perubahan ... 811
Langkah 3: Migrasikan indeks Anda ... 812
Langkah 4: Buat AWS DMS Tipe instans replikasi ... 813
Langkah 5: Buat AWS DMS titik akhir sumber ... 814
Langkah 6: Buat AWS DMS titik akhir target ... 817
Langkah 7: Buat dan jalankan tugas migrasi ... 819
Langkah 8: Mengubah endpoint aplikasi ke cluster Amazon DocumentDB target ... 821
Menggunakan rilis dukungan jangka panjang (LTS) ... 821
Rilis LTS Saat Ini ... 821
Siapa yang Harus Menggunakan LTS ... 822
Cara membuat cluster LTS baru ... 822
Cara memutakhirkan dari versi mesin 3.6.0 atau 4.0.0 ke cluster LTS 5.0.0 ... 823
Cara memeriksa apakah cluster mesin versi 5.0.0 saya yang ada ada di LTS ... 823
Extended Support ... 824
Ikhtisar Extended Support ... 825
Biaya Extended Support ... 825
Tanggung Jawab ... 826
Keamanan ... 827
Manajemen kata sandi dengan AWS Secrets Manager ... 828
Batasan untuk integrasi Secrets Manager dengan Amazon DocumentDB ... 829
Ikhtisar mengelola kata sandi pengguna utama dengan AWS Secrets Manager ... 829

Menegakkan pengelolaan Amazon DocumentDB dari kata sandi pengguna utama di AWS Secrets Manager ... 830
Mengelola kata sandi pengguna utama untuk klaster dengan Secrets Manager ... 830
Perlindungan data ... 831
Client-side enkripsi tingkat lapangan ... 832
Mengenkripsi data diam ... 839
Mengenkripsi data dalam perjalanan ... 847
Manajemen kunci ... 858
Identity and Access Management ... 858
Audiens ... 859
Mengautentikasi dengan identitas ... 859
Mengelola akses menggunakan kebijakan ... 861
Bagaimana Amazon DocumentDB bekerja dengan IAM ... 862
Identity-based contoh kebijakan ... 869
Pemecahan masalah ... 872
Mengelola izin akses ke sumber daya Amazon DocumentDB Anda ... 874
Menggunakan kebijakan berbasis identitas (kebijakan IAM) ... 880
AWS kebijakan terkelola untuk Amazon DocumentDB ... 884
Referensi izin API Amazon DocumentDB ... 903
Cross-service pencegahan wakil bingung ... 912
Otentikasi menggunakan identitas IAM ... 913
Memulai dengan pengguna dan peran IAM ... 914
Mengkonfigurasi jenis komputasi ... 917
Memantau permintaan otentikasi ... 918
Menggunakan otentikasi IAM ... 918
Driver yang mendukung IAM ... 919
FAQ otentikasi identitas IAM ... 919
Mengelola pengguna Amazon DocumentDB ... 920
Pengguna utama dan `serviceadmin` ... 920
Membuat pengguna tambahan ... 921
Putar kata sandi secara otomatis ... 923
Role-Based Kontrol Akses ... 924
Konsep RBAC ... 925
Memulai dengan peran bawaan RBAC ... 927
Memulai dengan peran yang ditentukan pengguna RBAC ... 930
Menghubungkan ke Amazon DocumentDB sebagai Pengguna ... 935

Perintah umum ... 937
Perbedaan fungsional ... 942
Batas ... 942
Akses database menggunakan Role-Based Access Control ... 942
Pencatatan log dan pemantauan ... 951
Memperbarui sertifikat ... 951
Memperbarui aplikasi Anda dan klaster Amazon DocumentDB ... 952
Rotasi sertifikat server otomatis ... 956
Pemecahan masalah ... 957
Pertanyaan yang Sering Diajukan ... 958
Validasi kepatuhan ... 964
Ketahanan ... 965
Keamanan infrastruktur ... 966
Titik akhir VPC (AWS PrivateLink) ... 967
Pertimbangan untuk titik akhir VPC ... 968
Ketersediaan wilayah ... 968
Membuat titik akhir VPC antarmuka untuk Amazon DocumentDB API ... 969
Membuat kebijakan titik akhir VPC untuk Amazon DocumentDB API ... 970
Praktik terbaik keamanan ... 971
Mengaudit peristiwa ... 972
Acara yang didukung ... 973
Aktifkan audit ... 978
Nonaktifkan audit ... 986
Akses acara Anda ... 988
Filter acara DML ... 989
Amazon VPC dan Amazon DocumentDB ... 994
Cluster DocumentDB dalam VPC ... 995
Mengakses cluster di VPC ... 1007
Buat IPv4-only VPC untuk sebuah cluster ... 1010
Buat VPC dual-stack untuk sebuah cluster ... 1016
Mencadangkan dan memulihkan ... 1027
Cadangkan dan pulihkan: konsep ... 1028
Memahami penggunaan penyimpanan cadangan ... 1030
Membuang, memulihkan, mengimpor, dan mengekspor data ... 1031
`mongodump` ... 1032
`mongorestore` ... 1033

mongoekspor ... 1033
mongoimpor ... 1034
Tutorial ... 1034
Pertimbangan snapshot cluster ... 1037
Penyimpanan cadangan ... 1038
Periode pencadangan ... 1038
Periode retensi cadangan ... 1040
Salin enkripsi snapshot cluster ... 1041
Membandingkan snapshot otomatis dan manual ... 1041
Membuat snapshot cluster manual ... 1043
Menyalin snapshot cluster ... 1046
Menyalin snapshot bersama ... 1047
Menyalin snapshot di seluruh Wilayah AWS ... 1047
Batasan ... 1047
Menangani enkripsi ... 1048
Pertimbangan grup parameter ... 1048
Menyalin snapshot cluster ... 1049
Berbagi snapshot klaster ... 1058
Berbagi snapshot terenkripsi ... 1059
Berbagi snapshot ... 1062
Memulihkan dari snapshot cluster ... 1066
Memulihkan ke titik waktu ... 1073
Menghapus snapshot cluster ... 1079
Mengelola Amazon DocumentDB ... 1082
Ikhtisar tugas operasional ... 1082
Menambahkan replika ke cluster Amazon DocumentDB ... 1083
Menjelaskan cluster dan instance ... 1084
Membuat snapshot cluster ... 1086
Memulihkan dari snapshot ... 1087
Menghapus instance dari cluster ... 1088
Menghapus klaster ... 1089
Klaster Global ... 1089
Apa itu klaster global? ... 1089
Bagaimana klaster global berguna? ... 1090
Apa keterbatasan terkini klaster global? ... 1090
Panduan memulai cepat ... 1091

Mengelola klaster global 1103
Menghubungkan kluster global 1111
Memantau klaster global 1111
Pemulihan bencana 1112
Mengelola klaster 1128
Memahami cluster 1129
Pengaturan cluster 1131
Konfigurasi penyimpanan cluster 1134
Menentukan status klaster 1137
Siklus hidup cluster 1139
Menskalakan klaster 1181
Mengkloning volume untuk cluster 1185
Memahami toleransi kesalahan cluster 1198
Mengelola contoh 1200
Menentukan status instans 1200
Siklus hidup instans 1200
Mengelola kelas instance 1224
NVMe-backed contoh 1237
Mengelola grup subnet 1240
Membuat grup subnet 1241
Menjelaskan grup subnet 1246
Mengubah grup subnet 1249
Menghapus grup subnet 1252
Ketersediaan dan replikasi tinggi 1253
Penskalaan baca 1254
Ketersediaan tinggi 1254
Menambahkan replika 1255
Failover 1256
Replikasi lag 1261
Mengelola indeks 1262
Pembuatan indeks Amazon DocumentDB 1262
Mempertahankan indeks Amazon DocumentDB 1268
Mengelola kompresi dokumen 1270
Mengelola kompresi dokumen 1271
Memantau kompresi dokumen 1273
Mengelola kompresi berbasis kamus 1274

Pertimbangan Kinerja 1275
Mengaktifkan kompresi berbasis kamus 1275
Memulai 1276
Memantau 1276
Mengelola acara 1277
Melihat kategori acara 1278
Melihat acara Amazon DocumentDB 1280
Memilih wilayah dan zona ketersediaan 1283
Ketersediaan wilayah 1284
Mengelola kelompok parameter cluster 1287
Menjelaskan kelompok parameter cluster 1287
Membuat grup parameter cluster 1295
Memodifikasi kelompok parameter cluster 1298
Memodifikasi cluster untuk menggunakan kelompok parameter cluster yang disesuaikan .. 1303
Menyalin kelompok parameter cluster 1304
Menyetel ulang kelompok parameter cluster 1306
Menghapus grup parameter cluster 1309
Referensi parameter cluster 1312
Memahami titik akhir 1335
Menemukan titik akhir klaster 1336
Menemukan titik akhir instance 1338
Menghubungkan ke titik akhir 1342
Memahami Amazon DocumentDB ARN 1343
Membuat konsep ARN 1343
Menemukan ARN 1347
Memberikan tag ke sumber daya 1349
Ikhtisar tag sumber daya 1350
Kendala tag 1351
Menambahkan atau memperbarui tag 1351
Mencantumkan tanda 1353
Menghapus tanda 1355
Memelihara Amazon DocumentDB 1356
Penomoran versi 1357
Kelola jendela pemeliharaan 1358
Pemberitahuan patch mesin 1361
Lihat pemeliharaan yang tertunda 1362

Pembaruan mesin ... 1365
Versi minor ... 1368
Pembaruan sistem operasi ... 1369
User-initiated pembaruan ... 1372
Klaster Global ... 1373
Memahami peran terkait layanan ... 1374
Service-linked izin peran ... 1374
Membuat peran tertaut layanan ... 1376
Memodifikasi peran terkait layanan ... 1376
Menghapus peran tertaut layanan ... 1377
Wilayah yang didukung untuk peran terkait layanan Amazon DocumentDB ... 1378
Menggunakan aliran perubahan ... 1378
Operasi yang didukung ... 1379
Penagihan ... 1379
Batasan ... 1379
Mengaktifkan aliran perubahan ... 1380
Menggunakan aliran perubahan dengan Python ... 1382
Pencarian dokumen lengkap ... 1385
Melanjutkan aliran perubahan ... 1386
Melanjutkan dengan `startAtOperationTime` ... 1387
Melanjutkan dengan `postBatchResumeToken` ... 1389
Transaksi ... 1391
Memodifikasi durasi retensi log ... 1391
Ubah aliran pada instance sekunder ... 1394
Menggunakan Collation ... 1395
Batasan ... 1396
Menggunakan Tampilan ... 1396
Praktik Terbaik ... 1397
Kompatibilitas Operator Agregator ... 1398
Memanfaatkan Indeks dan Tampilan ... 1398
Penggunaan AWS Lambda dengan aliran perubahan ... 1398
Batasan ... 1399
Menggunakan cluster elastis ... 1401
Kasus penggunaan cluster elastis ... 1402
Profil pengguna ... 1402
Manajemen konten dan catatan sejarah ... 1402

Keuntungan dari cluster elastis 1402
AWS Integrasi layanan 1402
Wilayah dan ketersediaan versi 1403
Ketersediaan wilayah untuk cluster elastis 1403
Ketersediaan versi 1404
Batasan 1404
Manajemen cluster elastis 1404
Operasi kueri dan tulis 1405
Pengumpulan dan manajemen indeks 1405
Administrasi dan diagnostik 1405
Opt-in fitur 1406
Cara kerjanya 1406
Sharding cluster elastis Amazon DocumentDB 1406
Migrasi cluster elastis 1410
Penskalaan cluster elastis 1410
Keandalan cluster elastis 1410
Penyimpanan dan ketersediaan cluster elastis 1410
Perbedaan fungsional antara Amazon DocumentDB 4.0 dan cluster elastis 1411
Memulai 1412
Prasyarat 1413
Langkah 1: Buat cluster elastis 1414
Langkah 2: Connect ke cluster elastis Anda 1420
Langkah 3: Pecahkan koleksi, sisipkan, dan kueri data Anda 1421
Langkah 4: Jelajahi 1425
Praktik terbaik 1425
Memilih kunci pecahan 1425
Manajemen koneksi 1426
Koleksi unsharded 1426
Penskalaan cluster elastis 1426
Memantau cluster elastis 1427
Mengelola cluster elastis 1427
Memodifikasi konfigurasi cluster elastis 1428
Memantau cluster elastis 1431
Menghapus cluster elastis 1435
Mengelola snapshot cluster elastis 1437
Menghentikan dan memulai cluster elastis 1451

Mempertahankan cluster elastis ... 1456
Enkripsi data saat istirahat ... 1464
Bagaimana cluster elastis Amazon DocumentDB menggunakan hibah di AWS KMS ... 1466
Buat kunci yang dikelola pelanggan ... 1467
Memantau kunci enkripsi Anda untuk cluster elastis Amazon DocumentDB ... 1468
Pelajari selengkapnya ... 1473
Service-linked peran ... 1474
Service-linked izin peran untuk cluster elastis ... 1474
Memantau Amazon DocumentDB ... 1478
Memantau status klaster ... 1479
Nilai status cluster ... 1480
Memantau status klaster ... 1481
Memantau status instans ... 1483
Nilai status instans ... 1484
Memantau status instans menggunakan Konsol Manajemen AWS atau AWS CLI ... 1486
Nilai status kesehatan contoh ... 1488
Memantau status kesehatan contoh menggunakan Konsol Manajemen AWS ... 1488
Melihat rekomendasi Amazon DocumentDB ... 1490
Langganan peristiwa ... 1493
Berlangganan peristiwa ... 1494
Kelola langganan ... 1496
Kategori dan pesan ... 1499
Memantau Amazon DocumentDB dengan CloudWatch ... 1502
Metrik Amazon DocumentDB ... 1503
Melihat CloudWatch data ... 1520
Dimensi Amazon DocumentDB ... 1526
Memantau metrik Opcounter ... 1526
Memantau koneksi database ... 1526
Mencatat panggilan API Amazon DocumentDB dengan CloudTrail ... 1527
Informasi Amazon DocumentDB di CloudTrail ... 1527
Operasi pembuatan profil ... 1528
Operasi yang didukung ... 1529
Batasan ... 1530
Mengaktifkan profiler ... 1530
Menonaktifkan profiler ... 1535
Menonaktifkan ekspor log profiler ... 1535

Mengakses log profiler Anda 1538
Pertanyaan umum 1538
Memantau dengan Wawasan Kinerja 1539
Konsep Performance Insights 1540
Mengaktifkan dan menonaktifkan Wawasan Kinerja 1544
Mengonfigurasi kebijakan akses untuk Wawasan Performa 1547
Menganalisis metrik dengan dasbor Wawasan Performa 1551
Mengambil metrik dengan API Wawasan Performa 1570
CloudWatch Metrik Amazon untuk Performance Insights 1585
Performance Insights untuk metrik penghitung 1587
OpenSearch integrasi 1589
OpenSearch Layanan Amazon sebagai tujuan 1589
Langkah 1: Buat domain OpenSearch Layanan Amazon atau koleksi OpenSearch tanpa server 1590
Langkah 2: Aktifkan aliran perubahan di cluster Amazon DocumentDB 1590
Langkah 3: Siapkan peran pipeline dengan izin untuk menulis ke bucket Amazon S3 dan domain atau koleksi tujuan 1590
Langkah 4: Tambahkan izin yang diperlukan pada peran pipeline untuk membuat X-ENI .. 1591
Langkah 5: Buat pipa 1592
Batasan 1592
Menggunakan DocumentDB tanpa server 1594
Kasus penggunaan tanpa server 1594
Menggunakan Amazon DocumentDB tanpa server untuk beban kerja yang sudah tersedia 1596
Keuntungan dari Amazon DocumentDB tanpa server 1597
Cara kerja tanpa server 1598
Ikhtisar 1598
Konfigurasi untuk cluster Amazon DocumentDB 1600
Kapasitas penskalaan tanpa server Amazon DocumentDB 1600
Penskalaan tanpa server Amazon DocumentDB 1602
Keadaan menganggur (0,5 DCU) 1604
Persyaratan dan batasan tanpa server 1605
Persyaratan fitur 1605
Membuat cluster yang menggunakan serverless 1606
Membuat klaster tanpa server Amazon DocumentDB 1606
Menambahkan instance tanpa server Amazon DocumentDB 1609

Migrasi ke tanpa server 1610
Migrasi cluster DocumentDB yang ada ke tanpa server 1610
Migrasi dari MongoDB ke DocumentDB tanpa server 1616
Mengelola tanpa server 1616
Melihat dan memodifikasi konfigurasi rentang kapasitas penskalaan klaster 1616
Melihat dan memodifikasi tingkat promosi pembaca tanpa server 1619
Batas instans tanpa server 1621
Konfigurasi penskalaan tanpa server 1624
Memilih rentang kapasitas penskalaan untuk cluster tanpa server DocumentDB 1624
Memilih `MinCapacity`pengaturan untuk cluster tanpa server DocumentDB 1624
Memilih `MaxCapacity`pengaturan untuk cluster tanpa server DocumentDB 1627
Menghindari kesalahan kehabisan memori 1629
Mengapa instance tanpa server saya tidak diperkecil? 1630
Pemantauan tanpa server 1631
Kehabisan Memori: status parameter yang tidak kompatibel 1631
CloudWatch Metrik Amazon untuk DocumentDB tanpa server 1631
Memantau kinerja tanpa server DocumentDB dengan Performance Insights 1636
Melaksanakan Pengembangan dengan Amazon DocumentDB 1637
Connect ke DocumentDB 1637
Menghubungkan secara terprogram 1637
Menghubungkan dari luar VPC Amazon 1666
Menghubungkan sebagai set replika 1667
Connect menggunakan Studio 3T 1673
Connect menggunakan DataGrip 1682
Connect menggunakan Amazon EC2 1687
Connect menggunakan driver JDBC 1725
Connect menggunakan driver ODBC 1741
Program dengan DocumentDB 1755
Panduan pemrograman Java DocumentDB 1755
Menggunakan validasi skema JSON 1788
Kuota dan batas 1799
Tipe instans yang didukung 1799
Wilayah yang didukung 1801
Kuota regional 1803
Batas agregasi 1806
Batas cluster 1806

Batas instans 1808
Kendala penamaan 1814
Kendala TTL 1816
Batas cluster elastis 1816
Batas pecahan cluster elastis 1817
CPU cluster elastis, memori, koneksi, dan batas kursor per pecahan 1818
Melakukan kueri 1819
Menanyakan dokumen 1819
Mengambil semua dokumen 1820
Mencocokkan nilai bidang 1820
Dokumen tertanam 1820
Nilai bidang dalam dokumen yang disematkan 1821
Mencocokkan array 1821
Mencocokkan nilai dalam array 1821
Menggunakan operator 1822
Rencana kueri 1822
Rencana Kueri 1822
Cache rencana kueri 1824
Jelaskan hasil 1824
Tahap pemindaian dan filter 1825
Persimpangan indeks 1826
Serikat indeks 1827
Beberapa indeks intersection/union 1828
Indeks majemuk 1828
Sortir tahap 1829
Panggung grup 1829
Perencana kueri v2 1829
Prasyarat 1830
Memilih perencana versi 2.0 sebagai perencana kueri default 1830
Praktik terbaik 1831
Batasan 1831
Perbaikan untuk `Menemukan` dan `Memperbarui` Operator 1831
Rencanakan API filter cache 1834
Potensi perbedaan perilaku antara perencana versi 1.0, 2.0, dan MongoDB 1836
Planner versi 2.0 menjembatani kesenjangan perilaku dengan MongoDB 1843
Perencana kueri v3 1846

Prasyarat ... 1846
Memilih perencana versi 3.0 sebagai perencana kueri default ... 1847
Praktik terbaik ... 1847
Batasan ... 1847
Perbaikan pada `$operator agregat` dan `berbeda` ... 1848
Potensi perbedaan perilaku antara perencana versi 1.0, 3.0, dan MongoDB ... 1849
Data geospasial ... 1850
Ikhtisar ... 1
Mengindeks dan menyimpan data geospasial ... 1851
Meminta data geospasial ... 1853
Batasan ... 1857
Indeks sebagian ... 1857
Buat indeks sebagian ... 1857
Operator yang didukung ... 1857
Kueri menggunakan indeks paral ... 1858
Fungsionalitas indeks sebagian ... 1859
Keterbatasan indeks sebagian ... 1863
Pencarian teks ... 1864
Fungsionalitas yang didukung ... 1864
Menggunakan indeks teks Amazon DocumentDB ... 1865
Perbedaan dengan MongoDB ... 1870
Praktik dan pedoman terbaik ... 1871
Indeks Teks V2 ... 1871
Batasan ... 1872
Indeks ... 1873
Manajemen Indeks ... 1873
Pembuatan indeks ... 1873
Mempertahankan indeks ... 1880
Praktik terbaik untuk bekerja dengan indeks ... 1882
Jenis Indeks ... 1884
Indeks Bidang Tunggal ... 1884
Indeks Senyawa ... 1886
Multi-key Indeks ... 1887
Indeks Teks ... 1889
Indeks Geospasial ... 1890
Indeks vektor ... 1891

Properti Indeks ... 1893
Topik ... 1893
name ... 1893
khas ... 1896
jarang ... 1899
sebagian FilterExpression ... 1903
kedaluwarsa AfterSeconds ... 1907
Optimalisasi Kinerja ... 1910
Masalah Kinerja Umum ... 1910
CPU tinggi ... 1910
Masalah Koneksi ... 1912
Kueri Berjalan Lambat ... 1919
Anti-patterns ... 1924
Indeks senyawa anti-pola lebih dari 3 atribut ... 1924
Kueri yang berjalan lama ... 1926
Indeks yang digunakan rendah dan redundan ... 1929
Multi-key indeks dengan array besar ... 1932
Tips Peningkatan Kinerja ... 1933
1. Gunakan $match sebagai Tahap Pertama di Agregation Pipelines ... 1933
2. Gunakan $project di Agregation Pipeline untuk Meminimalkan Ukuran Data Pipeline ... 1933
3. Aktifkan Kompresi Dokumen untuk Biaya Penyimpanan yang Lebih Rendah, I/O Biaya, dan Peningkatan Kinerja Kueri ... 1934
4. Indeks Leverage untuk Kinerja Kueri Optimal ... 1934
5. Optimalkan Model Data Berdasarkan Pola Kueri ... 1935
Analisis Rencana Kueri ... 1937
Waktu Keseluruhan ... 1940
Tahapan Eksekusi ... 1940
Memahami DocsExamined dalam Menjelaskan Statistik Eksekusi Rencana ... 1942
Pemecahan Masalah ... 1950
Masalah konektivitas ... 1950
Tidak dapat terhubung ke titik akhir Amazon DocumentDB ... 1950
Menguji koneksi ke instans Amazon DocumentDB ... 1956
Menghubungkan ke titik akhir yang tidak valid ... 1956
Konfigurasi driver memengaruhi jumlah koneksi ... 1957
Indeks ... 1957
Pembuatan indeks gagal ... 1957

Indeks latar belakang membangun masalah latensi dan gagal 1958
Bloat indeks 1959
Kinerja dan pemanfaatan sumber daya 1960
Lihat statistik sisipan, perbarui, dan hapus 1961
Menganalisis kinerja cache 1962
Temukan dan Akhiri Kueri yang Berjalan Lama atau Diblokir 1963
Lihat Rencana Kueri dan Mengoptimalkan Kueri 1965
Bagaimana saya bisa melihat rencana kueri dalam cluster elastis? 1967
Daftar Semua Operasi yang Berjalan pada Instans 1969
Ketahui Kapan Kueri Sedang Berjalan 1972
Tentukan Mengapa Sistem Tiba-tiba Berjalan Lambat 1975
Tentukan Penyebab Pemanfaatan CPU yang Tinggi 1976
Temukan kursor terbuka pada sebuah instance 1977
Lihat versi mesin Amazon DocumentDB saat ini 1978
Menganalisis penggunaan indeks dan mengidentifikasi indeks yang tidak digunakan 1978
Identifikasi indeks yang terlewat 1980
Bagaimana cara menentukan pengumpulan database bloat? 1982
Ringkasan pertanyaan yang berguna 1983
Pengumpulan sampah 1985
Memahami pengumpulan Sampah di Amazon DocumentDB 1985
Proses pengumpulan sampah 1986
Arsitektur penyimpanan dan penyimpanan yang diperluas 1986
Memantau pengumpulan sampah 1987
Contoh keluaran CollStats 1990
Pertanyaan umum 1992
Catatan rilis 1995
Catatan rilis versi mesin Amazon DocumentDB 1995
Catatan rilis umum Amazon DocumentDB 2050
Catatan rilis cluster elastis Amazon DocumentDB 2055
Riwayat dokumen 2059
..... mmlxxi

Apa itu Amazon DocumentDB (dengan kompatibilitas MongoDB)

Amazon DocumentDB (dengan kompatibilitas MongoDB) adalah layanan basis data terkelola penuh yang cepat dan andal. Amazon DocumentDB memudahkan pengaturan, pengoperasian, dan MongoDB-compatible skala database di cloud. Dengan Amazon DocumentDB, Anda dapat menjalankan kode aplikasi yang sama dan menggunakan driver dan alat-alat yang sama yang Anda gunakan dengan MongoDB.

Sebelum menggunakan Amazon DocumentDB, Anda harus meninjau konsep dan fitur yang dijelaskan dalam Cara kerjanya. Setelah itu, selesaikan langkah-langkah di Panduan Memulai.

Topik

- Gambaran umum Amazon DocumentDB
- klaster
- Contoh
- Wilayah dan zona ketersediaan
- Harga Amazon DocumentDB
- Memantau
- Antarmuka
- Apa selanjutnya?
- Amazon DocumentDB: cara kerjanya
- Apa itu database dokumen?

Gambaran umum Amazon DocumentDB

Berikut ini adalah beberapa fitur tingkat tinggi dari Amazon DocumentDB:

- Amazon DocumentDB mendukung dua jenis cluster: cluster berbasis instance dan cluster elastis. Cluster elastis mendukung beban kerja dengan jutaan reads/writes per detik dan kapasitas penyimpanan petabyte. Untuk informasi lebih lanjut tentang gugus elastis, lihatMenggunakan cluster elastis Amazon DocumentDB. Konten di bawah ini mengacu pada cluster berbasis instans Amazon DocumentDB.

- Amazon DocumentDB secara otomatis meningkatkan ukuran volume penyimpanan Anda seiring kebutuhan penyimpanan basis data Anda semakin besar. Volume penyimpanan Anda bertambah 10 GB, hingga maksimum 128 TiB. Anda tidak perlu menyediakan kelebihan penyimpanan apa pun untuk klaster Anda untuk menangani pertumbuhan masa depan.
- Dengan Amazon DocumentDB, Anda dapat meningkatkan throughput baca untuk mendukung permintaan aplikasi volume tinggi dengan membuat hingga 15 instans replika. Replika Amazon DocumentDB berbagi penyimpanan yang mendasari sama, menurunkan biaya dan menghindari kebutuhan untuk melakukan penulisan pada simpul replika. Kemampuan ini membebaskan lebih banyak daya pemrosesan untuk melayani permintaan baca dan mengurangi waktu jeda replika—sering kali hingga satu digit milidetik. Anda dapat menambahkan replika dalam hitungan menit terlepas dari ukuran volume penyimpanan. Amazon DocumentDB juga menyediakan reader endpoint, sehingga aplikasi dapat terhubung tanpa harus melacak replika karena mereka ditambahkan dan dihapus.
- Amazon DocumentDB memungkinkan Anda menskalakan ke atas atau ke bawah sumber daya komputasi dan memori untuk setiap instans Anda. Operasi penskalaan komputasi biasanya selesai dalam beberapa menit.
- Amazon DocumentDB berjalan di Amazon Virtual Private Cloud (Amazon VPC), sehingga Anda dapat mengisolasi basis data Anda di jaringan virtual Anda sendiri. Anda juga dapat mengonfigurasi pengaturan firewall untuk mengontrol akses jaringan ke klaster Anda.
- Amazon DocumentDB terus memantau kesehatan klaster Anda. Pada kegagalan instans, Amazon DocumentDB secara otomatis memulai ulang instans dan proses yang berkaitan. Amazon DocumentDB tidak memerlukan pemutaran ulang pemulihan kecelakaan pada log redo basis data, yang sangat mengurangi waktu pemulaian ulang. Amazon DocumentDB juga mengisolasi cache basis data dari proses basis data, mengaktifkan cache untuk mempertahankan pemulaian ulang instans.
- Pada kegagalan instans, Amazon DocumentDB mengotomatisasi failover ke salah satu hingga 15 replika Amazon DocumentDB yang Anda buat di Availability Zone lainnya. Jika tidak ada replika telah disediakan dan terjadi kegagalan, Amazon DocumentDB mencoba untuk membuat instans Amazon DocumentDB baru secara otomatis.
- Kemampuan backup di Amazon DocumentDB memungkinkan pemulihan titik waktu (PITR) untuk klaster Anda. Fitur ini memungkinkan Anda untuk memulihkan klaster ke detik berapa pun selama periode retensi, hingga 5 menit terakhir. Anda dapat mengonfigurasi periode retensi cadangan otomatis hingga 35 hari. Pencadangan otomatis disimpan di Amazon Simple Storage Service (Amazon S3), yang dirancang untuk daya tahan 99,999999999%. Backup Amazon DocumentDB

bersifat otomatis, inkremental, dan terus menerus, dan mereka tidak berdampak pada kinerja klaster Anda.

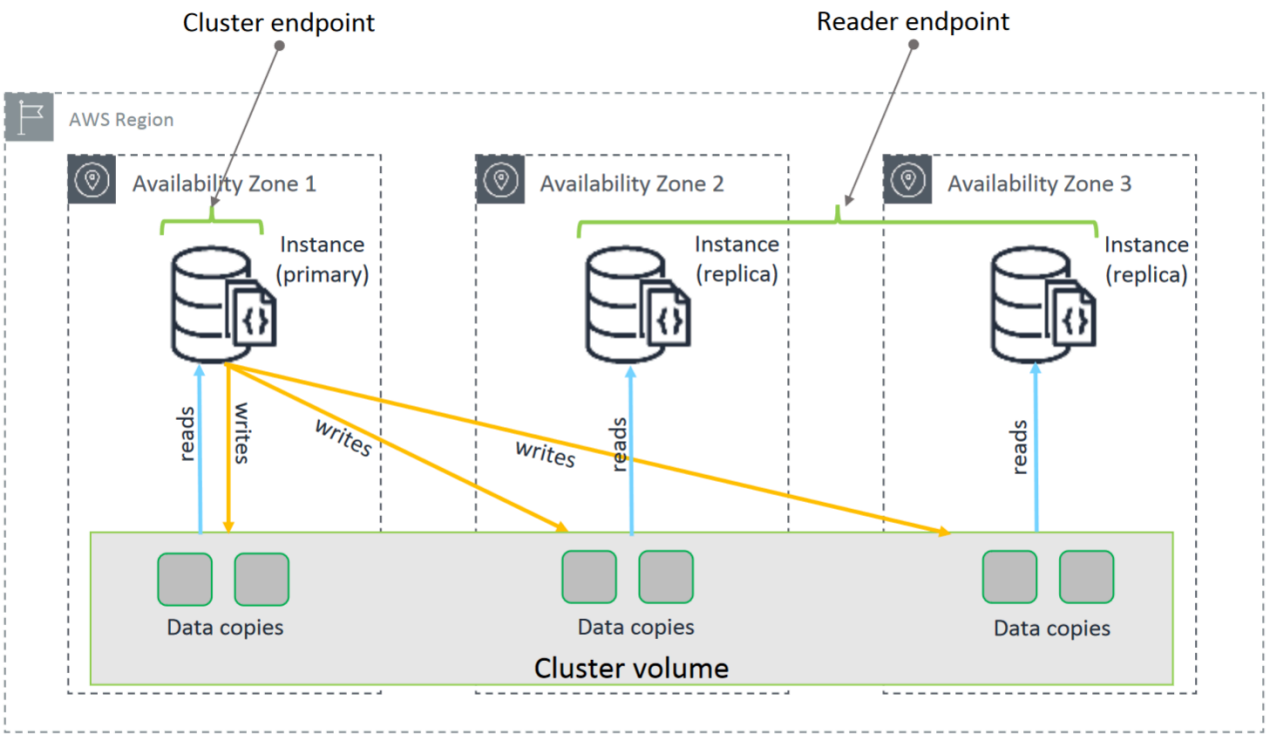
- Dengan Amazon DocumentDB, Anda dapat mengenkripsi database menggunakan kunci yang Anda buat dan kontrol melalui (). AWS Key Management Service AWS KMS Pada klaster basis data yang berjalan dengan enkripsi Amazon DocumentDB, data yang disimpan di rest dalam penyimpanan yang mendasari dienkripsi. Backup otomatis, snapshot, dan replika di klaster yang sama juga dienkripsi.
- Amazon DocumentDB disahkan berdasarkan Federal Risk and Authorization Management Program (FedRAMP). Ini memiliki otorisasi FedRAMP High AWS GovCloud untuk wilayah (AS) dan otorisasi FedRAMP Moderate untuk Wilayah AS. AWS East/West Untuk detail tentang AWS dan upaya kepatuhan, lihat AWS Layanan dalam Lingkup oleh Program Kepatuhan.

Jika Anda baru mengenal AWS layanan, gunakan sumber daya berikut untuk mempelajari lebih lanjut:

- AWS menawarkan layanan untuk komputasi, database, penyimpanan, analitik, dan fungsionalitas lainnya. Untuk ikhtisar semua AWS layanan, lihat Komputasi Awan dengan Amazon Web Services.
- AWS menyediakan sejumlah layanan database. Untuk panduan tentang layanan terbaik untuk lingkungan Anda, lihat Basis data pada AWS.

klaster

Klaster terdiri dari 0 sampai 16 instans dan volume penyimpanan klaster yang mengelola data untuk instans tersebut. Semua penulisan dilakukan melalui instans primer. Semua instans (primer dan replika) mendukung pembacaan. Data klaster disimpan dalam volume klaster dengan salinan dalam tiga Availability Zone yang berbeda.



Cluster berbasis instans Amazon DocumentDB 5.0 mendukung dua konfigurasi penyimpanan untuk kluster database: standar Amazon DocumentDB dan Amazon DocumentDB. I/O-optimized Untuk mengetahui informasi selengkapnya, lihat Konfigurasi penyimpanan cluster Amazon DocumentDB.

Contoh

Instans Amazon DocumentDB adalah lingkungan basis data terisolasi di cloud. Instans Anda dapat berisi beberapa basis data yang dibuat pengguna. Anda dapat membuat dan memodifikasi instance menggunakan Konsol Manajemen AWS atau AWS CLI.

Komputasi dan kapasitas memori dari instans ditentukan oleh kelas instans-nya. Anda dapat memilih instans yang paling sesuai dengan kebutuhan Anda. Jika kebutuhan Anda berubah seiring waktu, Anda dapat memilih kelas instans yang berbeda. Untuk spesifikasi kelas instans, lihat Spesifikasi kelas instans.

Instans Amazon DocumentDB berjalan hanya di lingkungan Amazon VPC. Amazon VPC memberikan Anda kontrol atas lingkungan jaringan virtual Anda: Anda dapat memilih rentang alamat IP Anda sendiri, membuat subnet, dan mengonfigurasi perutean dan daftar kontrol akses (ACL).

Sebelum Anda dapat membuat instans Amazon DocumentDB, Anda harus membuat klaster yang akan berisi instans.

Tidak semua kelas instans didukung di setiap wilayah. Tabel berikut ini menunjukkan kelas instans mana yang didukung di setiap wilayah.

> **Note**
>
> Untuk daftar lengkap jenis instans yang didukung oleh Amazon DocumentDB di setiap kelas instance, lihat. Spesifikasi kelas instans

Kelas instans yang didukung berdasarkan Wilayah

Region	R8G	R6GD	R6G	R5	R4	T4G	T3	Nirserver
AS Timur (Ohio)	Didukung	Didukung	Didukung	Didukung	Didukung	Didukung	Didukung	Didukung
US East (N. Virginia)	Didukung	Didukung	Didukung	Didukung	Didukung	Didukung	Didukung	Didukung
US West (Oregon)	Didukung	Didukung	Didukung	Didukung	Didukung	Didukung	Didukung	Didukung
Africa (Cape Town)			Didukung	Didukung		Didukung	Didukung	Didukung
Amerika Selatan (Sao Paulo)		Didukung	Didukung	Didukung		Didukung	Didukung	Didukung
Asia Pasifik (Hong Kong)			Didukung	Didukung		Didukung	Didukung	Didukung
Asia Pasifik (Hyderabad)			Didukung	Didukung		Didukung	Didukung	Didukung
Asia Pasifik (Malaysia)			Didukung			Didukung	Didukung	
Asia Pasifik (Mumbai)	Didukung	Didukung	Didukung	Didukung		Didukung	Didukung	Didukung

Region	R8G	R6GD	R6G	R5	R4	T4G	T3	Nirserver
Asia Pasifik (Osaka)		Didukung	Didukung	Didukung		Didukung	Didukung	
Asia Pasifik (Seoul)	Didukung	Didukung	Didukung	Didukung		Didukung	Didukung	Didukung
Asia Pacific (Sydney)	Didukung	Didukung	Didukung	Didukung		Didukung	Didukung	Didukung
Asia Pasifik (Jakarta)	Didukung	Didukung	Didukung	Didukung		Didukung	Didukung	
Asia Pacific (Melbourne)			Didukung	Didukung		Didukung	Didukung	
Asia Pacific (Singapore)	Didukung	Didukung	Didukung	Didukung		Didukung	Didukung	Didukung
Asia Pasifik (Thailand)			Didukung			Didukung	Didukung	
Asia Pacific (Tokyo)	Didukung	Didukung	Didukung	Didukung		Didukung	Didukung	Didukung
Kanada (Pusat)	Didukung	Didukung	Didukung	Didukung		Didukung	Didukung	Didukung
Kanada Barat (Calgary)			Didukung	Didukung		Didukung	Didukung	
Eropa (Frankfurt)	Didukung	Didukung	Didukung	Didukung		Didukung	Didukung	Didukung
Europe (Zurich)		Didukung	Didukung	Didukung		Didukung	Didukung	
Eropa (Irlandia)	Didukung	Didukung	Didukung	Didukung	Didukung	Didukung	Didukung	Didukung
Eropa (London)		Didukung	Didukung	Didukung		Didukung	Didukung	Didukung
Europe (Milan)			Didukung	Didukung		Didukung	Didukung	Didukung

Region	R8G	R6GD	R6G	R5	R4	T4G	T3	Nirserver
Eropa (Paris)		Didukung	Didukung	Didukung		Didukung	Didukung	Didukung
Eropa (Spanyol)	Didukung	Didukung	Didukung	Didukung		Didukung	Didukung	Didukung
Eropa (Stockholm)	Didukung	Didukung	Didukung	Didukung		Didukung	Didukung	
Meksiko (Tengah)			Didukung			Didukung	Didukung	
Timur Tengah (UAE)			Didukung	Didukung		Didukung	Didukung	Didukung
Tiongkok (Beijing)		Didukung	Didukung	Didukung		Didukung	Didukung	Didukung
Tiongkok (Ningxia)			Didukung	Didukung		Didukung	Didukung	Didukung
Israel (Tel Aviv)			Didukung	Didukung		Didukung	Didukung	
AWS GovCloud (US-West)	Didukung	Didukung	Didukung	Didukung			Didukung	Didukung
AWS GovCloud (US-East)		Didukung	Didukung	Didukung		Didukung	Didukung	Didukung

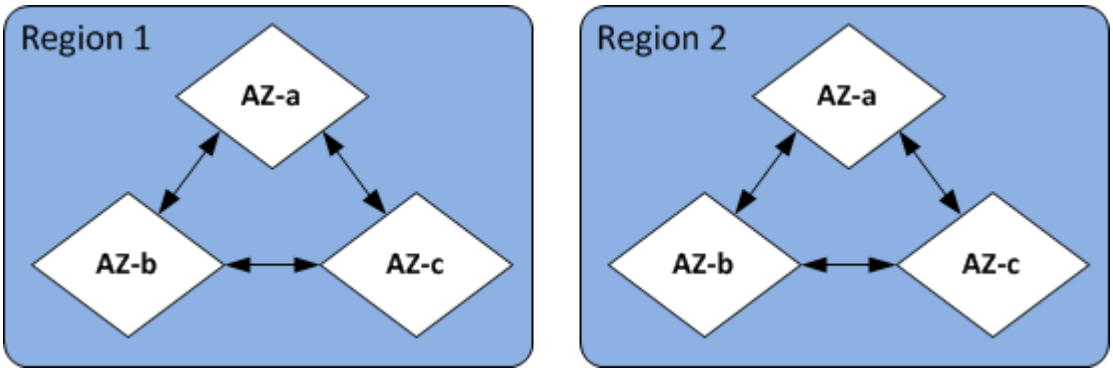
Wilayah dan zona ketersediaan

Wilayah dan Availability Zone menentukan lokasi fisik klaster dan instans Anda.

Wilayah

AWS Sumber daya komputasi awan ditempatkan di fasilitas pusat data yang sangat tersedia di berbagai wilayah di dunia (misalnya, Amerika Utara, Eropa, atau Asia). Setiap lokasi pusat data disebut Wilayah.

Setiap AWS Wilayah dirancang untuk sepenuhnya terisolasi dari AWS Wilayah lain. Di dalam setiap wilayah terdapat beberapa Zona Ketersediaan. Dengan meluncurkan simpul Anda di Zona Ketersediaan yang berbeda, Anda dapat mencapai toleransi kesalahan sebesar mungkin. Diagram berikut menunjukkan tampilan tingkat tinggi tentang cara kerja AWS Wilayah dan Zona Ketersediaan.



Zona ketersediaan

Setiap AWS Wilayah berisi beberapa lokasi berbeda yang disebut Availability Zone. Setiap Availability Zone direkayasa untuk diisolasi dari kegagalan di Availability Zone lainnya, dan untuk menyediakan konektivitas jaringan latensi rendah yang murah ke Availability Zone lainnya di Wilayah yang sama. Dengan meluncurkan instans untuk klaster tertentu di beberapa Availability Zone, Anda dapat melindungi aplikasi Anda dari peristiwa kegagalan Availability Zone yang tidak diinginkan.

Arsitektur Amazon DocumentDB memisahkan penyimpanan dan komputasi. Untuk lapisan penyimpanan, Amazon DocumentDB mereplikasi enam salinan data Anda di tiga Availability Zone. AWS Sebagai contoh, jika Anda meluncurkan klaster Amazon DocumentDB di Wilayah yang hanya mendukung dua Availability Zone, penyimpanan data Anda akan direplikasi enam cara di tiga Availability Zone tetapi instans komputasi Anda hanya akan tersedia di dua Availability Zone.

Tabel berikut mencantumkan jumlah Availability Zone yang dapat Anda gunakan dalam instance komputasi yang diberikan Wilayah AWS untuk menyediakan untuk klaster Anda.

Nama Wilayah	Region	Availability Zone (penghitungan)
AS Timur (Ohio)	`us-east-2`	3
AS Timur (Virginia Utara)	`us-east-1`	6
AS Barat (Oregon)	`us-west-2`	4

Nama Wilayah	Region	Availability Zone (penghitungan)
Africa (Cape Town)	af-south-1	3
Amerika Selatan (Sao Paulo)	sa-east-1	3
Asia Pasifik (Hong Kong)	ap-east-1	3
Asia Pasifik (Hyderabad)	ap-south-2	3
Asia Pasifik (Malaysia)	ap-southeast-5	3
Asia Pasifik (Mumbai)	ap-south-1	3
Asia Pasifik (Osaka)	ap-northeast-3	3
Asia Pasifik (Seoul)	ap-northeast-2	4
Asia Pasifik (Singapura)	ap-southeast-1	3
Asia Pasifik (Sydney)	ap-southeast-2	3
Asia Pasifik (Jakarta)	ap-southeast-3	3
Asia Pacific (Melbourne)	ap-southeast-4	3
Asia Pasifik (Thailand)	ap-southeast-7	3
Asia Pasifik (Tokyo)	ap-northeast-1	3
Kanada (Pusat)	ca-central-1	3
Kanada Barat (Calgary)	ca-west-1	3

Nama Wilayah	Region	Availability Zone (penghitungan)
Wilayah Tiongkok (Beijing)	`cn-north-1`	3
Tiongkok (Ningxia)	`cn-northwest-1`	3
Eropa (Frankfurt)	`eu-central-1`	3
Europe (Zurich)	`eu-central-2`	3
Eropa (Irlandia)	`eu-west-1`	3
Eropa (London)	`eu-west-2`	3
Europe (Milan)	`eu-south-1`	3
Eropa (Paris)	`eu-west-3`	3
Eropa (Spanyol)	`eu-south-2`	3
Eropa (Stockholm)	`eu-north-1`	3
Meksiko (Tengah)	`mx-central-1`	3
Timur Tengah (UAE)	`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

Harga Amazon DocumentDB

Cluster Amazon DocumentDB ditagih berdasarkan komponen berikut:

- Jam instans (per jam)—Berdasarkan kelas instans dari instans (sebagai contoh, `db.r5.xlarge`). Harga dicantumkan per jam, tetapi tagihan dihitung turun menjadi detik dan menunjukkan waktu dalam bentuk desimal. Penggunaan Amazon DocumentDB ditagihkan dalam kenaikan satu detik, dengan minimum 10 menit. Untuk informasi selengkapnya, lihat Mengelola kelas instance.
- I/O Permintaan (per 1 juta permintaan per bulan) — Jumlah total I/O permintaan penyimpanan yang Anda buat dalam siklus penagihan.
- Penyimpanan backup (per GiB per bulan) — Penyimpanan backup adalah penyimpanan yang berkaitan dengan backup basis data otomatis dan setiap snapshot basis data aktif yang telah Anda ambil. Meningkatkan periode retensi cadangan atau mengambil snapshot basis data tambahan akan meningkatkan penyimpanan cadangan yang digunakan oleh basis data Anda. Penyimpanan cadangan diukur GB-months dan per detik tidak berlaku. Untuk informasi selengkapnya, lihat Mencadangkan dan memulihkan di Amazon DocumentDB.
- Transfer data (per GB) — Transfer data masuk dan keluar dari instans Anda dari atau ke internet atau AWS Wilayah lain.

Untuk informasi lebih lanjut, lihat harga Amazon DocumentDB.

Uji coba gratis

Anda dapat mencoba Amazon DocumentDB secara gratis menggunakan uji coba gratis 1 bulan. Untuk informasi selengkapnya, lihat Uji coba gratis di harga Amazon DocumentDB atau lihat FAQ uji coba gratis Amazon DocumentDB.

Memantau

Terdapat beberapa cara untuk melacak performa dan kesehatan instans. Anda dapat menggunakan CloudWatch layanan Amazon gratis untuk memantau kinerja dan kesehatan suatu instans. Anda dapat menemukan grafik performa di konsol Amazon DocumentDB. Anda dapat berlangganan peristiwa Amazon DocumentDB untuk mendapatkan notifikasi tentang perubahan yang terjadi pada instans, snapshot, grup parameter, atau grup keamanan.

Untuk informasi selengkapnya, lihat berikut ini:

- Memantau Amazon DocumentDB dengan CloudWatch
- Mencatat panggilan API Amazon DocumentDB dengan AWS CloudTrail

Antarmuka

Ada beberapa cara bagi Anda untuk berinteraksi dengan Amazon DocumentDB, termasuk Konsol Manajemen AWS dan. AWS CLI

Konsol Manajemen AWS

Konsol Manajemen AWS Ini adalah antarmuka pengguna berbasis web yang sederhana. Anda dapat mengelola klaster dan instans Anda dari konsol tanpa perlu pemrograman. Untuk mengakses konsol Amazon DocumentDB, masuk ke Konsol Manajemen AWS dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb

AWS CLI

Anda dapat menggunakan AWS Command Line Interface (AWS CLI) untuk mengelola klaster dan instans Amazon DocumentDB Anda. Dengan konfigurasi minimal, Anda dapat mulai menggunakan semua fungsionalitas yang disediakan oleh konsol Amazon DocumentDB dari program terminal favorit Anda.

- Untuk menginstal AWS CLI, lihat Menginstal Antarmuka Baris AWS Perintah.
- Untuk mulai menggunakan Amazon DocumentDB AWS CLI untuk Amazon, AWS lihat Referensi Antarmuka Baris Perintah untuk Amazon DocumentDB.

Driver MongoDB

Untuk mengembangkan dan menulis aplikasi terhadap klaster Amazon DocumentDB, Anda juga dapat menggunakan driver MongoDB dengan Amazon DocumentDB. Untuk informasi selengkapnya, lihat tab shell MongoDB di atau. Menghubungkan dengan TLS diaktifkan Menghubungkan dengan TLS dinonaktifkan

Apa selanjutnya?

Bagian sebelumnya memperkenalkan Anda pada komponen infrastruktur dasar yang ditawarkan Amazon DocumentDB. Apa yang harus Anda lakukan selanjutnya? Tergantung pada keadaan Anda, lihat salah satu topik berikut untuk memulai:

- Memulai Amazon DocumentDB dengan membuat cluster dan instance menggunakan. CloudFormation Amazon DocumentDB mulai cepat menggunakan CloudFormation

- Memulai dengan Amazon DocumentDB dengan membuat klaster dan instans menggunakan petunjuk di Panduan Memulai kami.
- Mulailah dengan Amazon DocumentDB dengan membuat cluster elastis menggunakan instruksi di. Memulai dengan cluster elastis Amazon DocumentDB
- Migrasikan implementasi MongoDB Anda ke Amazon DocumentDB menggunakan panduan di Migrasi ke Amazon DocumentDB

Amazon DocumentDB: cara kerjanya

Amazon DocumentDB (dengan kompatibilitas MongoDB) adalah layanan database yang dikelola sepenuhnya. MongoDB-compatible Dengan Amazon DocumentDB, Anda dapat menjalankan kode aplikasi yang sama dan menggunakan driver dan alat yang sama yang Anda gunakan dengan MongoDB. Amazon DocumentDB kompatibel dengan MongoDB 3.6, 4.0, 5.0, dan 8.0.

Topik

- Versi Amazon DocumentDB
- Titik akhir Amazon DocumentDB
- Dukungan TLS
- Penyimpanan Amazon DocumentDB
- Replikasi Amazon DocumentDB
- Keandalan Amazon DocumentDB
- Baca opsi preferensi
- TTL menghapus
- Sumber daya yang dapat ditagih

Versi Amazon DocumentDB

> **Note**
>
> Penomoran versi minor berlaku untuk Amazon DocumentDB 5.0 dan yang lebih baru. Amazon DocumentDB 3.6 dan 4.0 hanya menggunakan versi utama dengan patch mesin.

Dimulai dengan Amazon DocumentDB 5.0, versi mesin menggunakan skema penomoran empat tingkat: major.major.minor.patch.

- Versi utama —Diidentifikasi oleh dua angka pertama dari versi (misalnya, 5.0). Versi utama mewakili tingkat kompatibilitas MongoDB. Versi utama memperkenalkan fitur baru, peningkatan kinerja, dan pembaruan kompatibilitas MongoDB. Upgrade antara versi utama (misalnya, dari 4.0 ke 5.0) memerlukan upgrade versi mayor (MVU). Untuk informasi selengkapnya, lihat Peningkatan versi utama Amazon DocumentDB di tempat.
- Versi minor —Diidentifikasi oleh nomor ketiga dari versi (misalnya, 1 di 5.0.1). Versi minor mencakup peningkatan seperti fitur baru, perbaikan keamanan, dan perbaikan bug dalam versi utama yang sama. Anda dapat meningkatkan ke versi minor baru secara manual. Untuk informasi selengkapnya, lihat Peningkatan versi minor Amazon DocumentDB.
- Versi patch —Patch mesin diterapkan dalam versi minor dan berisi perbaikan keamanan penting dan perbaikan bug. Versi patch dilacak oleh versi patch mesin (misalnya,`3.0.17983`), yang dapat Anda periksa dengan menjalankan`db.runCommand({getEngineVersion: 1})`. Patch diterapkan selama jendela pemeliharaan cluster Anda. Untuk informasi selengkapnya, lihat Memelihara Amazon DocumentDB.

Saat Anda membuat cluster baru, Anda dapat menentukan versi mesin apa pun yang tersedia. Untuk informasi tentang versi yang tersedia dan tanggal dukungan, lihatTanggal dukungan versi mesin Amazon DocumentDB.

Saat Anda menggunakan Amazon DocumentDB, Anda mulai dengan membuat klaster. Sebuah klaster terdiri dari nol atau lebih instans basis data dan volume klaster yang mengelola data untuk instans tersebut. Volume klaster Amazon DocumentDB adalah volume penyimpanan basis data virtual yang mencakup beberapa Availability Zone. Setiap Availability Zone memiliki salinan data klaster.

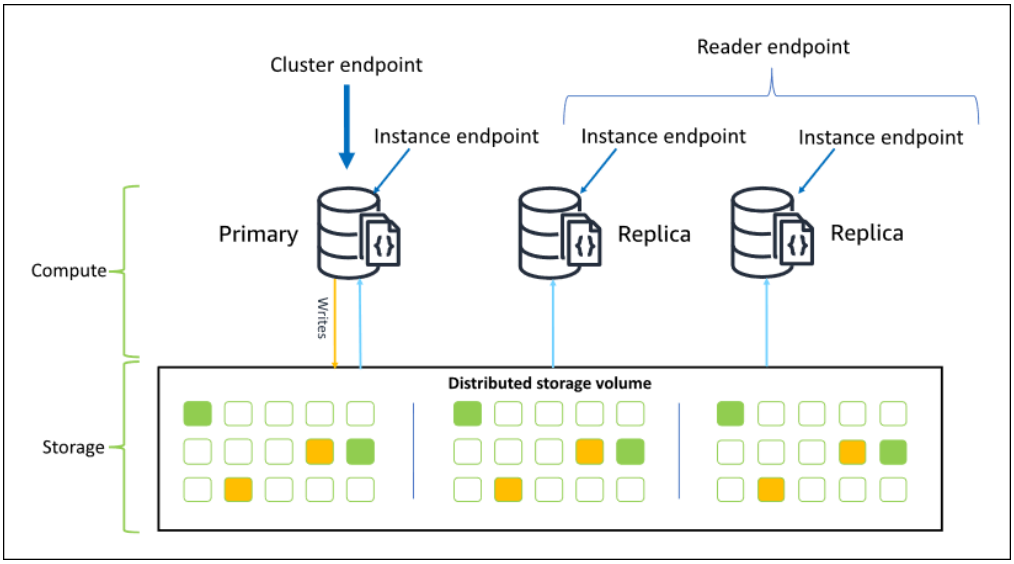
Klaster Amazon DocumentDB terdiri dari dua komponen:

- Volume klaster—Menggunakan layanan penyimpanan cloud-native untuk mereplikasi data dengan enam cara di tiga Availability Zone, menyediakan penyimpanan yang sangat berdaya tahan dan tersedia. Cluster Amazon DocumentDB memiliki tepat satu volume cluster, yang dapat menyimpan hingga 128 TiB data.
- Instans—Menyediakan kekuatan pemrosesan untuk basis data, menulis data ke, dan membaca data dari, volume penyimpanan klaster. Klaster Amazon DocumentDB dapat memiliki 0–16 instans.

Instans melayani salah satu dari dua peran:

- Instans primer—Mendukung operasi baca dan tulis, dan melakukan semua modifikasi data ke volume klaster. Setiap klaster Amazon DocumentDB memiliki satu instans primer.
- Instans replika—Mendukung hanya operasi baca. Setiap klaster Amazon DocumentDB dapat memiliki hingga 15 instans replika selain instans primer. Memiliki beberapa replika memungkinkan Anda untuk mendistribusikan beban kerja baca. Selain itu, dengan menempatkan replika di Availability Zone terpisah, Anda juga meningkatkan ketersediaan klaster.

Diagram berikut mengilustrasikan hubungan antara volume klaster, instans primer, dan replika di klaster Amazon DocumentDB:



Instans klaster tidak harus dari kelas instans yang sama, dan mereka dapat disediakan dan diakhiri sesuai yang diinginkan. Arsitektur ini memungkinkan Anda menskalakan kapasitas komputasi klaster Anda secara independen dari penyimpanannya.

Saat aplikasi Anda menulis data ke instans primer, yang primer mengeksekusi penulisan yang tahan lama ke volume klaster. Itu kemudian mereplikasi status penulisan itu (bukan data) ke setiap replika aktif. Replika Amazon DocumentDB tidak berpartisipasi dalam pemrosesan tulis, dan dengan demikian replika Amazon DocumentDB menguntungkan untuk penskalaan baca. Pembacaan dari replika Amazon DocumentDB pada akhirnya konsisten dengan jeda replika minimal —biasanya kurang dari 100 milidetik setelah instans primer menulis data. Bacaan dari replika dijamin akan dibaca sesuai urutan penulisannya ke yang primer. Jeda replika bervariasi tergantung pada kecepatan perubahan data, dan periode aktivitas tulis yang tinggi dapat meningkatkan jeda replika. Untuk informasi lebih, lihat metrik `ReplicationLag` di Metrik Amazon DocumentDB.

Titik akhir Amazon DocumentDB

Amazon DocumentDB menyediakan beberapa opsi koneksi untuk melayani berbagai kasus penggunaan. Untuk terhubung ke instans di klaster Amazon DocumentDB, Anda menentukan titik akhir instans. Sebuah titik akhir adalah alamat host dan nomor port, dipisahkan oleh titik dua.

Kami merekomendasikan Anda terhubung ke klaster menggunakan titik akhir klaster dan dalam mode set replika (lihat Menghubungkan ke Amazon DocumentDB sebagai set replika) kecuali jika Anda memiliki kasus penggunaan khusus untuk menghubungkan ke reader endpoint atau titik akhir instans. Untuk merutekan permintaan ke replika Anda, pilih pengaturan preferensi baca driver yang memaksimalkan penskalaan baca sekaligus memenuhi persyaratan konsistensi baca aplikasi Anda. Preferensi baca `secondaryPreferred` mengaktifkan pembacaan replika dan membebaskan instans primer untuk melakukan lebih banyak pekerjaan.

Titik akhir berikut tersedia dari klaster Amazon DocumentDB.

Titik Akhir klaster

Titik akhir klaster terhubung ke instans primer klaster Anda saat ini. Titik akhir klaster dapat digunakan untuk operasi baca dan tulis. Sebuah klaster Amazon DocumentDB memiliki tepat satu titik akhir klaster.

Titik akhir klaster menyediakan dukungan failover untuk koneksi baca dan tulis ke klaster. Jika instans primer klaster Anda saat ini gagal, dan klaster Anda memiliki setidaknya satu replika baca aktif, titik akhir klaster secara otomatis mengalihkan permintaan koneksi ke instans primer baru. Saat menghubungkan ke klaster Amazon DocumentDB Anda, kami merekomendasikan Anda menghubungkan ke klaster Anda menggunakan titik akhir klaster dan dalam mode set replika (lihat Menghubungkan ke Amazon DocumentDB sebagai set replika).

Berikut ini adalah contoh titik akhir klaster Amazon DocumentDB:

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

Berikut ini adalah contoh koneksi string menggunakan titik akhir klaster ini:

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

Untuk informasi tentang menemukan titik akhir klaster, lihat Menemukan titik akhir klaster.

Titik akhir pembaca

Beban reader endpoint menyeimbangkan koneksi baca-saja di semua replika yang tersedia di klaster Anda. Titik akhir pembaca cluster akan tampil sebagai titik akhir cluster jika Anda terhubung melalui `replicaSet` mode, artinya dalam string koneksi, parameter set replika adalah. `&replicaSet=rs0` Dalam hal ini, Anda akan dapat melakukan operasi penulisan pada primer. Namun, jika Anda terhubung ke klaster yang menentukan`directConnection=true`, maka mencoba melakukan operasi penulisan melalui koneksi ke titik akhir pembaca menghasilkan kesalahan. Sebuah klaster Amazon DocumentDB memiliki tepat satu reader endpoint.

Jika klaster hanya berisi satu instans (primer), reader endpoint terhubung ke instans primer. Saat Anda menambahkan instans replika ke klaster Amazon DocumentDB, reader endpoint membuka koneksi baca saja ke replika baru setelah itu aktif.

Berikut ini adalah contoh reader endpoint untuk klaster Amazon DocumentDB:

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

Berikut ini adalah contoh string koneksi menggunakan reader endpoint:

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

Beban reader endpoint menyeimbangkan koneksi baca-saja, bukan permintaan baca. Jika beberapa koneksi reader endpoint lebih banyak digunakan daripada yang lain, permintaan baca Anda mungkin tidak seimbang di antara instans di dalam klaster. Direkomendasikan untuk mendistribusikan permintaan dengan menghubungkan ke titik akhir klaster sebagai set replika dan menggunakan opsi preferensi baca secondaryPreferred.

Untuk informasi tentang menemukan titik akhir klaster, lihat Menemukan titik akhir klaster.

Titik akhir instans

Titik akhir instans terhubung ke instans tertentu dalam klaster Anda. Titik akhir instans untuk instans primer saat ini dapat digunakan untuk operasi baca dan tulis. Namun, mencoba melakukan operasi tulis ke titik akhir instans untuk replika baca akan menghasilkan kesalahan. Klaster Amazon DocumentDB memiliki satu titik akhir instans per instans aktif.

Titik akhir instans memberikan kontrol langsung atas koneksi ke instans tertentu untuk skenario di mana titik akhir klaster atau reader endpoint mungkin tidak sesuai. Contoh kasus penggunaan adalah

penyediaan beban kerja analitik baca-saja secara berkala. Anda dapat menyediakan instans replika yang lebih besar dari biasanya, menghubungkan langsung ke instans baru yang lebih besar dengan titik akhir instansnya, menjalankan kueri analitik, lalu menghentikan instans. Menggunakan titik akhir instans menjaga lalu lintas analitik agar tidak memengaruhi instans klaster lainnya.

Berikut ini adalah contoh titik akhir instans untuk instans tunggal di klaster Amazon DocumentDB:

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

Berikut ini adalah contoh string koneksi menggunakan titik akhir instans ini:

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

> **Note**
>
> Peran instans sebagai primer atau replika dapat berubah karena peristiwa failover. Aplikasi Anda tidak boleh berasumsi bahwa titik akhir instans tertentu adalah instans primer. Kami tidak menyarankan menghubungkan ke titik akhir instans untuk aplikasi produksi. Sebagai gantinya, kami merekomendasikan Anda menyambungkan ke klaster menggunakan titik akhir klaster dan dalam mode set replika (lihat Menghubungkan ke Amazon DocumentDB sebagai set replika). Untuk kendali lebih lanjut dari prioritas failover instans, lihat Memahami toleransi kesalahan klaster Amazon DocumentDB.

Untuk informasi tentang menemukan titik akhir klaster, lihat Menemukan titik akhir instance.

Mode set replika

Anda dapat terhubung ke titik akhir klaster Amazon DocumentDB Anda dalam mode set replika dengan menentukan nama set replika `rs0`. Menghubungkan dalam mode set replika menyediakan kemampuan untuk menentukan opsi Perhatian Baca, Perhatian Tulis, dan Preferensi Baca. Untuk informasi selengkapnya, lihat Konsistensi baca.

Berikut ini adalah contoh koneksi string yang menghubungkan dalam mode set replika:

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

Saat Anda terhubung dalam mode set replika, klaster Amazon DocumentDB Anda muncul ke driver dan klien Anda sebagai set replika. Instans yang ditambahkan dan dihapus dari klaster Amazon DocumentDB Anda direfleksikan secara otomatis dalam konfigurasi set replika.

Setiap klaster Amazon DocumentDB terdiri dari set replika tunggal dengan nama default `rs0`. Nama set replika tidak dapat diubah.

Menghubungkan ke titik akhir klaster dalam mode set replika adalah metode yang disarankan untuk penggunaan umum.

> **Note**
>
> Semua instans dalam klaster Amazon DocumentDB mendengarkan pada port TCP yang sama untuk koneksi.

Dukungan TLS

Untuk detail selengkapnya tentang menghubungkan ke Amazon DocumentDB menggunakan Transport Layer Security (TLS), lihat Mengenkripsi data dalam perjalanan.

Penyimpanan Amazon DocumentDB

Data Amazon DocumentDB disimpan dalam volume klaster, yang merupakan volume virtual tunggal yang menggunakan solid state drive (SSD). Volume klaster terdiri dari enam salinan data Anda, yang direplikasi secara otomatis di beberapa Availability Zone dalam satu Wilayah AWS. Replikasi ini membantu memastikan bahwa data Anda sangat berdaya tahan, dengan kemungkinan kehilangan data yang lebih kecil. Ini juga membantu memastikan bahwa klaster Anda lebih tersedia selama failover karena salinan data Anda sudah ada di Availability Zone lainnya. Salinan ini dapat terus melayani permintaan data ke instans di klaster Amazon DocumentDB Anda.

Bagaimana penyimpanan data ditagih

Amazon DocumentDB secara otomatis meningkatkan ukuran volume klaster saat jumlah data meningkat. Volume cluster Amazon DocumentDB dapat tumbuh hingga ukuran maksimum 128 TiB; Namun, Anda hanya dikenakan biaya untuk ruang yang Anda gunakan dalam volume cluster Amazon DocumentDB. Dimulai dengan Amazon DocumentDB 4.0, ketika data dihapus, seperti dengan menjatuhkan koleksi atau indeks, keseluruhan ruang yang dialokasikan berkurang dengan jumlah yang sebanding. Dengan demikian, Anda dapat mengurangi biaya penyimpanan dengan

menghapus koleksi, indeks, dan database yang tidak lagi Anda perlukan. Di Amazon DocumentDB versi 3.6, volume cluster dapat menggunakan kembali ruang yang dibebaskan saat Anda menghapus data, tetapi volume itu sendiri tidak pernah berkurang ukurannya. Akibatnya di versi 3.6, Anda mungkin tidak menyaksikan perubahan apa pun dalam penyimpanan saat Anda menjatuhkan koleksi atau indeks, meskipun ruang kosong digunakan kembali.

> **Note**
>
> Dengan Amazon DocumentDB 3.6, biaya penyimpanan didasarkan pada penyimpanan "tanda air tinggi" (jumlah maksimum yang dialokasikan untuk cluster Amazon DocumentDB kapan saja). Anda dapat mengelola biaya dengan menghindari praktik ETL yang membuat volume besar informasi sementara, atau yang memuat volume besar data baru sebelum menghapus data lama yang tidak dibutuhkan. Jika menghapus data dari klaster Amazon DocumentDB menghasilkan sejumlah besar ruang yang dialokasikan tetapi tidak digunakan, mengatur ulang tanda air yang tinggi memerlukan pembuangan data logis dan memulihkan ke klaster baru, menggunakan alat seperti `mongodump` atau `mongorestore`. Membuat dan memulihkan snapshot tidak mengurangi penyimpanan yang dialokasikan karena tata letak fisik penyimpanan yang mendasarinya tetap sama di snapshot yang dipulihkan.

> **Note**
>
> Menggunakan utilitas seperti `mongodump` dan `mongorestore` dikenakan I/O biaya berdasarkan ukuran data yang sedang dibaca dan ditulis ke volume penyimpanan.

Untuk informasi tentang penyimpanan I/O dan harga data Amazon DocumentDB, lihat FAQ harga dan Harga Amazon DocumentDB (dengan kompatibilitas MongoDB).

Replikasi Amazon DocumentDB

Dalam klaster Amazon DocumentDB, setiap instans replika menampilkan titik akhir independen. Titik akhir replika ini menyediakan akses baca-saja ke data dalam volume klaster. Mereka memungkinkan Anda untuk menskalakan beban kerja baca untuk data Anda melalui beberapa instans yang direplikasi. Mereka juga membantu meningkatkan performa pembacaan data dan meningkatkan ketersediaan data di klaster Amazon DocumentDB Anda. Replika Amazon DocumentDB juga merupakan target failover dan dengan cepat dipromosikan jika instans primer untuk klaster Amazon DocumentDB Anda gagal.

Keandalan Amazon DocumentDB

Amazon DocumentDB dirancang agar andal, tahan lama, dan toleran terhadap kesalahan. (Untuk meningkatkan ketersediaan, Anda harus mengonfigurasi klaster Amazon DocumentDB Anda sehingga memiliki beberapa instans replika di Availability Zone yang berbeda.) Amazon DocumentDB menyertakan beberapa fitur otomatis yang menjadikannya solusi basis data yang andal.

Perbaikan penyimpanan otomatis

Amazon DocumentDB menyimpan banyak salinan data Anda di tiga Availability Zone, sangat mengurangi kemungkinan kehilangan data karena kegagalan penyimpanan. Amazon DocumentDB secara otomatis mendeteksi kegagalan di dalam volume klaster. Saat segmen dari sebuah volume klaster gagal, Amazon DocumentDB segera memperbaiki segmen tersebut. Ini menggunakan data dari volume lain yang membentuk volume klaster untuk membantu memastikan bahwa data di segmen yang diperbaiki adalah yang terkini. Sebagai hasil, Amazon DocumentDB menghindari kehilangan data dan mengurangi kebutuhan untuk melakukan pemulihan tepat waktu untuk memulihkan dari kegagalan instans.

Pemanasan cache yang bisa bertahan

Amazon DocumentDB mengelola cache halamannya dalam proses terpisah dari basis data sehingga cache halaman dapat bertahan secara independen dari basis data. Jika terjadi kegagalan basis data yang tidak terduga, cache halaman tetap berada di memori. Ini memastikan bahwa kumpulan buffer dihangatkan dengan status terkini saat basis data dimulai ulang.

Pemulihan setelah crash

Amazon DocumentDB dirancang untuk pulih dari crash hampir seketika, dan untuk terus menyajikan data aplikasi Anda. Amazon DocumentDB melakukan pemulihan crash secara asinkron pada utas paralel sehingga basis data Anda terbuka dan tersedia segera setelah kerusakan.

Tata kelola sumber daya

Amazon DocumentDB melindungi sumber daya yang diperlukan untuk menjalankan proses penting dalam layanan, seperti pemeriksaan kondisi. Untuk melakukan ini, dan ketika instans mengalami tekanan memori tinggi, Amazon DocumentDB akan membatasi permintaan. Akibatnya, beberapa operasi mungkin diantrekan untuk menunggu tekanan memori mereda. Jika tekanan memori berlanjut, operasi yang diantrekan mungkin habis. Anda dapat memantau

apakah operasi pelambatan layanan karena memori rendah atau tidak dengan CloudWatch metrik berikut:`LowMemThrottleQueueDepth`,,,`LowMemThrottleMaxQueueDepth`. `LowMemNumOperationsThrottled` `LowMemNumOperationsTimedOut` Untuk informasi selengkapnya, lihat Memantau Amazon CloudWatch DocumentDB dengan. Jika Anda melihat tekanan memori berkelanjutan pada instans Anda sebagai akibat dari LowMem CloudWatch metrik, kami menyarankan Anda meningkatkan skala instance Anda untuk menyediakan memori tambahan untuk beban kerja Anda.

Baca opsi preferensi

Amazon DocumentDB menggunakan layanan penyimpanan bersama cloud-native yang mereplikasi data enam kali di tiga Availability Zone untuk memberikan tingkat ketahanan yang tinggi. Amazon DocumentDB tidak bergantung pada replikasi data ke beberapa instans untuk mencapai ketahanan. Data klaster Anda tahan lama baik berisi satu instans atau 15 instans.

Topik

- Tulis daya tahan
- Baca isolasi
- Konsistensi baca
- Ketersediaan tinggi
- Penskalaan dibaca

Tulis daya tahan

Amazon DocumentDB menggunakan sistem penyimpanan self-healing yang unik, terdistribusi, toleran terhadap kesalahan. Sistem ini mereplikasi enam salinan (V=6) data Anda di tiga AWS Availability Zone untuk memberikan ketersediaan dan daya tahan yang tinggi. Ketika menulis data, Amazon DocumentDB memastikan bahwa semua penulisan dicatat secara tahan lama di sebagian besar node sebelum menyatakan penulisan ke klien. Jika Anda menjalankan set replika MongoDB tiga simpul, menggunakan perhatian tulis `{w:3, j:true}` akan menghasilkan konfigurasi terbaik saat dibandingkan dengan Amazon DocumentDB.

Penulisan ke klaster Amazon DocumentDB harus diproses oleh instans tulis klaster. Mencoba menulis ke pembaca menghasilkan kesalahan. Tulisan yang diakui dari instance utama Amazon DocumentDB tahan lama, dan tidak dapat diputar kembali. Amazon DocumentDB sangat berdaya tahan secara default dan tidak mendukung opsi tulis yang tidak tahan lama. Anda tidak dapat

mengubah tingkat daya tahan (yaitu, perhatian tulis). Amazon DocumentDB mengabaikan w=anything dan secara efektif w: 3 dan j: true. Anda tidak dapat menguranginya.

Karena penyimpanan dan komputasi dipisahkan di dalam arsitektur Amazon DocumentDB, klaster dengan satu instans sangat berdaya tahan. Daya tahan ditangani di lapisan penyimpanan. Hasilnya, klaster Amazon DocumentDB dengan satu instans dan satu dengan tiga instans mencapai tingkat ketahanan yang sama. Anda dapat mengonfigurasi klaster Anda ke kasus penggunaan spesifik Anda sambil tetap memberikan daya tahan tinggi untuk data Anda.

Penulisan ke klaster Amazon DocumentDB bersifat atomik dalam satu dokumen.

Amazon DocumentDB tidak mendukung opsi `wtimeout` dan tidak akan mengembalikan kesalahan jika nilai ditentukan. Penulisan ke instans Amazon DocumentDB primer dijamin tidak akan diblokir tanpa batas waktu.

Baca isolasi

Pembacaan dari instans Amazon DocumentDB hanya mengembalikan data yang tahan lama sebelum kueri dimulai. Pembacaan tidak pernah mengembalikan data yang dimodifikasi setelah kueri memulai eksekusi, juga bacaan kotor tidak mungkin dilakukan dalam keadaan apa pun.

Konsistensi baca

Data yang dibaca dari klaster Amazon DocumentDB tahan lama dan tidak akan dibatalkan. Anda dapat mengubah konsistensi baca untuk pembacaan Amazon DocumentDB dengan menentukan preferensi baca untuk permintaan atau koneksi. Amazon DocumentDB tidak mendukung opsi baca yang tidak tahan lama.

Pembacaan dari instans primer klaster Amazon DocumentDB sangat konsisten dalam kondisi operasi normal dan memiliki konsistensi baca-setelah-tulis. Jika peristiwa failover terjadi antara penulisan dan pembacaan berikutnya, sistem dapat secara singkat mengembalikan pembacaan yang tidak sangat konsisten. Semua pembacaan dari replika baca pada akhirnya konsisten dan mengembalikan data dalam urutan yang sama, dan seringkali dengan jeda replika kurang dari 100 mili detik.

Preferensi baca Amazon DocumentDB

Amazon DocumentDB mendukung pengaturan opsi preferensi baca hanya saat membaca data dari titik akhir klaster dalam mode set replika. Mengatur opsi preferensi baca memengaruhi cara klien atau driver MongoDB Anda merutekan permintaan baca ke instans di klaster Amazon DocumentDB

Anda. Anda dapat mengatur opsi preferensi baca untuk kueri tertentu, atau sebagai opsi umum di driver MongoDB Anda. (Konsultasikan dokumentasi klien atau driver Anda untuk petunjuk tentang bagaimana cara mengatur opsi preferensi baca.)

Jika klien atau driver Anda tidak terhubung ke titik akhir klaster Amazon DocumentDB dalam mode set replika, hasil penentuan preferensi baca tidak ditentukan.

Amazon DocumentDB tidak mendukung pengaturan set tag sebagai preferensi baca.

Opsi Preferensi Baca Yang Didukung

- **`primary`**—Menentukan preferensi baca `primary` membantu memastikan bahwa semua pembacaan dirutekan ke instans primer klaster. Jika instans primer tidak tersedia, operasi baca gagal. Preferensi baca `primary` menghasilkan konsistensi baca-setelah-tulis dan sesuai untuk kasus penggunaan yang memprioritaskan konsistensi baca-setelah-tulis daripada ketersediaan tinggi dan penskalaan baca.

  Contoh berikut menentukan preferensi baca `primary`:

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

- **`primaryPreferred`**—Menentukan rute preferensi baca `primaryPreferred` membaca ke instans primer dalam operasi normal. Jika ada failover primer, klien merutekan permintaan ke replika. Preferensi baca `primaryPreferred` menghasilkan konsistensi baca-setelah-tulis selama operasi normal, dan pada akhirnya pembacaan yang konsisten selama peristiwa failover. Preferensi baca `primaryPreferred` sesuai untuk kasus penggunaan yang memprioritaskan konsistensi baca-setelah-tulis daripada penskalaan baca, tetapi masih memerlukan ketersediaan tinggi.

  Contoh berikut menentukan preferensi baca `primaryPreferred`:

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

- **`secondary`**—Menentukan preferensi baca `secondary` memastikan bahwa pembacaan hanya dirutekan ke replika, tidak pernah ke instans primer. Jika tidak ada instans replika dalam sebuah klaster, permintaan baca akan gagal. Preferensi baca `secondary` menghasilkan bacaan akhir

konsisten dan sesuai untuk kasus penggunaan yang memprioritaskan throughput tulis instans primer daripada ketersediaan tinggi dan konsistensi baca-setelah-tulis.

Contoh berikut menentukan preferensi baca `secondary`:

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

- **`secondaryPreferred`**—Menentukan preferensi baca `secondaryPreferred` memastikan bahwa pembacaan dirutekan ke replika baca saat satu atau beberapa replika aktif. Jika tidak ada instans replika aktif dalam sebuah klaster, permintaan baca akan dirutekan ke instans primer. Preferensi baca `secondaryPreferred` menghasilkan bacaan akhir konsisten saat pembacaan dilayani oleh replika baca. Ini menghasilkan konsistensi baca-setelah-tulis ketika pembacaan dilayani oleh instans primer (kecuali peristiwa failover). Preferensi baca `secondaryPreferred` sesuai untuk kasus penggunaan yang memprioritaskan penskalaan baca dan ketersediaan tinggi daripada konsistensi baca-setelah-tulis.

  Contoh berikut menentukan preferensi baca `secondaryPreferred`:

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

- **`nearest`**—Menentukan rute preferensi baca `nearest` membaca hanya berdasarkan latensi terukur antara klien dan semua instans di klaster Amazon DocumentDB. Preferensi baca `nearest` menghasilkan bacaan akhir konsisten saat pembacaan dilayani oleh replika baca. Ini menghasilkan konsistensi baca-setelah-tulis ketika pembacaan dilayani oleh instans primer (kecuali peristiwa failover). Preferensi baca `nearest` sesuai untuk kasus penggunaan yang memprioritaskan pencapaian latensi baca serendah mungkin dan ketersediaan tinggi dibandingkan konsistensi baca-setelah-tulis dan penskalaan baca.

  Contoh berikut menentukan preferensi baca `nearest`:

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

Ketersediaan tinggi

Amazon DocumentDB mendukung konfigurasi klaster yang sangat tersedia dengan menggunakan replika sebagai target failover untuk instans primer. Jika instans primer gagal, replika Amazon DocumentDB dipromosikan sebagai primer baru, dengan gangguan singkat selama permintaan baca dan tulis yang dibuat ke instans primer gagal dengan pengecualian.

Jika klaster Amazon DocumentDB Anda tidak menyertakan replika apa pun, instans primer akan dibuat ulang selama kegagalan. Namun, mempromosikan replika Amazon DocumentDB jauh lebih cepat daripada membuat ulang instans primer. Jadi kami merekomendasikan Anda membuat satu atau beberapa replika Amazon DocumentDB sebagai target failover.

Replika yang dimaksudkan untuk digunakan sebagai target failover harus dari kelas instans yang sama dengan instans primer. Mereka harus disediakan di Availability Zone yang berbeda dari yang primer. Anda dapat mengontrol replika mana yang lebih disukai sebagai target failover. Untuk praktik terbaik dalam mengonfigurasi Amazon DocumentDB untuk ketersediaan tinggi, lihat Memahami toleransi kesalahan klaster Amazon DocumentDB.

Penskalaan dibaca

Replika Amazon DocumentDB ideal untuk penskalaan baca. Mereka sepenuhnya didedikasikan untuk membaca operasi pada volume klaster Anda, yaitu, replika tidak memproses penulisan. Replikasi data terjadi dalam volume klaster dan bukan di antara instans. Jadi setiap sumber daya replika didedikasikan untuk memproses kueri Anda, bukan mereplikasi dan menulis data.

Jika aplikasi Anda membutuhkan lebih banyak kapasitas baca, Anda dapat menambahkan replika ke klaster Anda dengan cepat (biasanya dalam waktu kurang dari sepuluh menit). Jika persyaratan kapasitas baca Anda berkurang, Anda dapat menghapus replika yang tidak dibutuhkan. Dengan replika Amazon DocumentDB, Anda hanya membayar untuk kapasitas baca yang Anda butuhkan.

Amazon DocumentDB mendukung penskalaan baca sisi klien melalui penggunaan opsi Preferensi Baca. Untuk informasi selengkapnya, lihat Preferensi baca Amazon DocumentDB.

TTL menghapus

Penghapusan dari area indeks TTL yang dicapai melalui proses latar belakang adalah upaya terbaik dan tidak dijamin dalam jangka waktu tertentu. Faktor-faktor seperti ukuran instans, pemanfaatan sumber daya instans, ukuran dokumen, dan throughput keseluruhan dapat memengaruhi waktu penghapusan TTL.

Saat monitor TTL menghapus dokumen Anda, setiap penghapusan dikenakan biaya IO, yang akan menambah tagihan Anda. Jika throughput dan tingkat penghapusan TTL meningkat, Anda harus mengharapkan kenaikan tagihan Anda karena peningkatan penggunaan IO.

Saat Anda membuat indeks TTL pada koleksi yang ada, Anda harus menghapus semua dokumen yang kedaluwarsa sebelum membuat indeks. Implementasi TTL saat ini dioptimalkan untuk menghapus sebagian kecil dokumen dalam koleksi, yang biasa terjadi jika TTL diaktifkan pada koleksi dari awal, dan dapat menghasilkan IOPS yang lebih tinggi daripada yang diperlukan jika sejumlah besar dokumen perlu dihapus dalam sekali jalan.

Jika Anda tidak ingin membuat indeks TTL untuk menghapus dokumen, Anda dapat mengelompokkan dokumen ke dalam koleksi berdasarkan waktu, dan cukup membuang koleksi tersebut saat dokumen tidak lagi diperlukan. Misalnya: Anda dapat membuat satu koleksi per minggu dan membuangnya tanpa mengeluarkan biaya IO. Ini dapat secara signifikan menghemat biaya daripada menggunakan indeks TTL.

Sumber daya yang dapat ditagih

Mengidentifikasi sumber daya Amazon DocumentDB yang dapat ditagih

Sebagai layanan database yang dikelola sepenuhnya, Amazon DocumentDB mengenakan biaya untuk instans, penyimpanan, pencadangan I/Os, dan transfer data. Untuk informasi selengkapnya, lihat Harga Amazon DocumentDB (dengan kompatibilitas MongoDB).

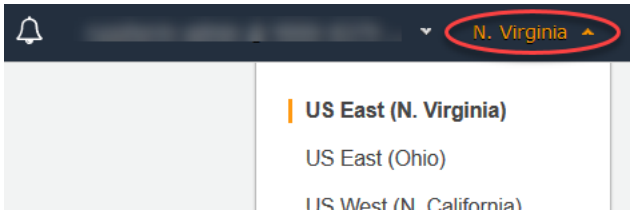
Untuk menemukan sumber daya yang dapat ditagih di akun Anda dan berpotensi menghapus sumber daya, Anda dapat menggunakan atau. Konsol Manajemen AWS AWS CLI

Menggunakan Konsol Manajemen AWS

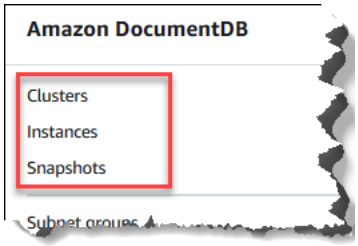
Dengan menggunakan Konsol Manajemen AWS, Anda dapat menemukan klaster Amazon DocumentDB, instance, dan snapshot yang telah Anda sediakan untuk diberikan. Wilayah AWS

Untuk menemukan klaster, instans, dan snapshot

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Untuk menemukan sumber daya yang dapat ditagih di Wilayah selain Wilayah default Anda, di sudut kanan atas layar, pilih yang ingin Anda cari. Wilayah AWS



3. Di panel navigasi, pilih tipe sumber daya yang dapat ditagih yang Anda minati: Klaster, Instans, atau Snapshot.



4. Semua klaster, instans, atau snapshot yang Anda sediakan untuk Wilayah terdaftar di panel kanan. Anda akan dikenakan biaya untuk klaster, instans, dan snapshot.

Menggunakan AWS CLI

Dengan menggunakan AWS CLI, Anda dapat menemukan klaster Amazon DocumentDB, instance, dan snapshot yang telah Anda sediakan untuk diberikan. Wilayah AWS

Untuk menemukan klaster dan instans

Kode berikut mencantumkan semua klaster dan instans Anda untuk Wilayah yang ditentukan. Jika Anda ingin mencari klaster dan instans di Wilayah default, Anda dapat menghilangkan parameter `--region`.

Example

Untuk Linux, macOS, atau Unix:

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

Untuk Windows:

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

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

Output dari operasi ini terlihat seperti berikut ini.

```
"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",
```

Untuk menemukan snapshot

Kode berikut mencantumkan semua snapshot Anda untuk Wilayah yang ditentukan. Jika Anda ingin mencari snapshot di Wilayah default Anda, Anda dapat menghilangkan parameter `--region`.

Untuk Linux, macOS, atau Unix:

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

Untuk Windows:

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

Output dari operasi ini terlihat seperti berikut ini.

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

```
        "manual"
    ]
]
```

Anda hanya perlu menghapus snapshot `manual`. Snapshot `Automated` dihapus saat Anda menghapus klaster.

Menghapus sumber daya yang tidak dapat ditagih

Untuk menghapus klaster, Anda harus menghapus semua instans di klaster terlebih dahulu.

- Untuk menghapus instans, lihat Menghapus instans Amazon DocumentDB.

> **Important**
>
> Meskipun Anda menghapus instans dalam sebuah klaster, Anda masih ditagih untuk penggunaan penyimpanan dan pencadangan yang terkait dengan klaster tersebut. Untuk menghentikan semua tagihan, Anda juga harus menghapus klaster dan snapshot manual.

- Untuk menghapus klaster, lihat Menghapus cluster Amazon DocumentDB.
- Untuk menghapus snapshot manual, lihat Menghapus snapshot cluster.

Apa itu database dokumen?

Beberapa developer tidak memikirkan model data mereka dalam hal baris dan kolom normal. Biasanya, di tingkat aplikasi, data direpresentasikan sebagai dokumen JSON karena lebih intuitif bagi developer untuk memikirkan model data mereka sebagai dokumen.

Popularitas basis data dokumen telah berkembang karena basis data dokumen ini membuat Anda dapat mempertahanan data dalam basis data dengan menggunakan format model dokumen yang sama yang Anda gunakan dalam kode aplikasi Anda. Basis data dokumen menyediakan API yang kuat dan intuitif untuk pengembangan yang fleksibel dan tangkas.

Topik

- Kasus penggunaan basis data dokumen
- Memahami dokumen
- Bekerja dengan dokumen

Kasus penggunaan basis data dokumen

Drive kasus penggunaan Anda apakah Anda memerlukan basis data dokumen atau beberapa jenis basis data lain untuk mengelola data Anda. Basis data dokumen berguna untuk beban kerja yang memerlukan skema fleksibel untuk pengembangan yang cepat dan berulang. Berikut ini adalah beberapa contoh kasus penggunaan di mana basis data dokumen dapat memberikan keuntungan yang signifikan:

Topik

- Profil pengguna
- Real-time data besar
- Manajemen konten

Profil pengguna

Karena basis data dokumen memiliki skema yang fleksibel, mereka dapat menyimpan dokumen yang memiliki atribut dan nilai data yang berbeda. Basis data dokumen adalah solusi praktis untuk profil online di mana pengguna yang berbeda menyediakan berbagai jenis informasi. Dengan menggunakan basis data dokumen, Anda dapat menyimpan profil setiap pengguna efisien dengan menyimpan hanya atribut yang spesifik untuk setiap pengguna.

Misalkan pengguna memilih untuk menambahkan atau menghapus informasi dari profil mereka. Dalam kasus ini, dokumen mereka dapat dengan mudah diganti dengan versi terbaru yang berisi atribut dan data baru apa pun yang baru ditambahkan atau menghilangkan atribut dan data baru apa pun yang dihilangkan. Basis data dokumen dengan mudah mengelola tingkat individualitas dan fluiditas ini.

Real-time data besar

Secara historis, kemampuan untuk mengekstrak informasi dari data operasional terhambat oleh fakta bahwa database operasional dan database analitis dipertahankan di lingkungan yang berbeda—operasional dan masing-masing. business/reporting Mampu mengekstrak informasi operasional secara langsung sangat penting dalam lingkungan bisnis yang sangat kompetitif. Dengan menggunakan basis data dokumen, bisnis dapat menyimpan dan mengelola data operasional dari sumber mana pun dan secara bersamaan mengumpankan data ke mesin BI pilihan untuk analisis. Tidak ada persyaratan untuk memiliki dua lingkungan.

Manajemen konten

Untuk mengelola konten secara efektif, Anda harus dapat mengumpulkan dan mengagregasikan konten dari berbagai sumber, dan kemudian mengirimkannya ke pelanggan. Akibat skema fleksibel mereka, basis data dokumen sempurna untuk mengumpulkan dan menyimpan jenis data apa pun. Anda dapat menggunakannya untuk membuat dan menggabungkan jenis konten baru, termasuk konten buatan pengguna, seperti gambar, komentar, dan video.

Memahami dokumen

Basis data dokumen digunakan untuk menyimpan data semistruktur sebagai dokumen—alih-alih menormalkan data di beberapa tabel, masing-masing dengan struktur yang unik dan tetap, seperti dalam basis data relasional. Dokumen yang disimpan dalam basis data dokumen menggunakan pasangan nilai kunci nest untuk menyediakan struktur atau skema dokumen. Namun demikian, jenis dokumen yang berbeda dapat disimpan dalam basis data dokumen yang sama, sehingga memenuhi persyaratan untuk memproses data serupa yang dalam format yang berbeda. Misalnya, karena setiap dokumen menggambarkan diri sendiri, JSON-encoded dokumen untuk toko online yang dijelaskan dalam topik Contoh dokumen dalam database dokumen dapat disimpan dalam database dokumen yang sama.

Topik

- SQL vs. terminologi non-relasional
- Dokumen sederhana
- Dokumen yang disematkan
- Contoh dokumen dalam database dokumen
- Memahami normalisasi dalam database dokumen

SQL vs. terminologi non-relasional

Tabel berikut ini membandingkan terminologi yang digunakan oleh basis data dokumen (MongoDB) dengan terminologi yang digunakan oleh basis data SQL.

SQL	MongoDB
Tabel	Koleksi
Baris	Dokumen

SQL	MongoDB
Kolom	Bidang
Kunci Primer	ObjectId
Indeks	Indeks
Lihat	Lihat
Tabel atau objek nest	Dokumen tersemat
Array	Array

Dokumen sederhana

Semua dokumen dalam basis data dokumen adalah dijelaskan sendiri. Dokumentasi ini menggunakan dokumen JSON-like yang diformat, meskipun Anda dapat menggunakan cara pengkodean lain.

Dokumen sederhana memiliki satu atau lebih bidang yang semuanya pada tingkat yang sama di dalam dokumen. Dalam contoh berikut ini, bidang `SSN`, `LName`, `FName`, `DOB`, `Street`, `City`, `State-Province`, `PostalCode`, dan`Country` semuanya adalah saudara kandung di dalam dokumen.

```
{
   "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"
}
```

Ketika informasi diatur dalam dokumen sederhana, setiap bidang terkelola secara individual. Untuk mengambil alamat seseorang, Anda harus mengambil `Street`, `City`,`State-Province`, `PostalCode`, dan `Country` sebagai item data individu.

Dokumen yang disematkan

Dokumen kompleks mengatur datanya dengan membuat dokumen tersemat di dalam dokumen. Dokumen tersemat membantu mengelola data dalam pengelompokan dan sebagai item data individual, mana yang lebih efisien dalam kasus tertentu. Dengan menggunakan contoh sebelumnya, Anda dapat menyematkan dokumen `Address` dalam dokumen utama. Melakukan hal ini akan menghasilkan struktur dokumen berikut ini:

```
{
   "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"
   }
}
```

Anda sekarang dapat mengakses data dalam dokumen sebagai bidang individu (`"SSN":`), sebagai dokumen tersemat (`"Address":`), atau sebagai anggota dokumen tersemat (`"Address": {"Street":}`).

Contoh dokumen dalam database dokumen

Seperti yang dinyatakan sebelumnya, karena setiap dokumen dalam basis data dokumen adalah dijelaskan sendiri, struktur dokumen di dalam basis data dokumen dapat berbeda satu sama lain. Dua dokumen berikut ini, satu untuk buku dan satu lagi untuk majalah, berbeda secara struktural. Namun keduanya dapat dalam basis data dokumen yang sama.

Berikut ini adalah dokumen buku sampel:

```
{
    "_id" : "9876543210123",
    "Type": "book",
    "ISBN": "987-6-543-21012-3",
    "Author":
    {
```

```
        "LName":"Roe",
        "MI": "T",
        "FName": "Richard"
    },
    "Title": "Understanding Document Databases"
}
```

Berikut ini adalah dokumen majalah sampel dengan dua artikel:

```
{
    "_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"
}
```

Bandingkan struktur kedua dokumen tersebut. Dengan basis data relasional, Anda perlu memisah tabel “majalah” dan “buku”, atau satu tabel dengan bidang yang tidak terpakai, seperti “Publikasi”, “Terbitan”, “Artikel”, dan “MI”, sebagai nilai `null`. Karena basis data dokumen adalah semistruktur, dengan setiap dokumen mendefinisikan strukturnya sendiri, kedua dokumen tersebut dapat hidup

berdampingan dalam basis data dokumen yang sama tanpa bidang `null`. Basis data dokumen baik dalam menangani data yang jarang.

Mengembangkan terhadap basis data dokumen memungkinkan pengembangan yang cepat dan berulang. Hal ini karena Anda dapat mengubah struktur data dokumen secara dinamis, tanpa harus mengubah skema untuk seluruh koleksi. Basis data dokumen sangat cocok untuk pengembangan tangkas dan secara dinamis mengubah lingkungan.

Memahami normalisasi dalam database dokumen

Basis data dokumen tidak dinormalisasi; data yang ditemukan dalam satu dokumen dapat diulang dalam dokumen lainnya. Lebih lanjut, beberapa perbedaan data dapat ada di antara dokumen. Sebagai contoh, pertimbangkan skenario di mana Anda melakukan pembelian di toko online dan semua detail pembelian Anda disimpan dalam dokumen tunggal. Dokumen mungkin terlihat seperti dokumen JSON berikut ini:

```
{
    "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"
}
```

Semua informasi ini disimpan sebagai dokumen dalam koleksi transaksi. Kemudian, Anda menyadari bahwa Anda lupa membeli satu item. Sehingga Anda kembali masuk ke toko yang sama dan melakukan pembelian lainnya, yang juga disimpan sebagai dokumen lainnya dalam koleksi transaksi.

```
{
    "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"
}
```

Perhatikan redundansi antara kedua dokumen tersebut—nama dan ID pembelanja Anda (dan, jika Anda menggunakan kartu kredit yang sama, informasi kartu kredit Anda). Tapi hal ini tidak apa-apa karena penyimpanan murah, dan setiap dokumen benar-benar mencatat transaksi tunggal yang dapat diambil dengan cepat dengan kueri nilai kunci sederhana tanpa harus bergabung.

Terdapat juga perbedaan yang jelas antara kedua dokumen—informasi kartu kredit Anda. Ini hanya perbedaan yang jelas karena kemungkinan Anda akan menggunakan kartu kredit yang berbeda untuk setiap pembelian. Setiap dokumen akurat untuk transaksi yang didokumentasikan.

Bekerja dengan dokumen

Sebagai basis data dokumen, Amazon DocumentDB membuatnya mudah untuk menyimpan, membuat kueri, dan membuat indeks data JSON. Di Amazon DocumentDB, koleksi analog dengan tabel dalam basis data relasional, kecuali tidak ada skema tunggal ditegakkan pada semua dokumen. Koleksi memungkinkan Anda mengelompokkan dokumen yang serupa bersama-sama sembari menjaga mereka semua dalam basis data yang sama, tanpa mengharuskan bahwa mereka identik dalam struktur.

Dengan menggunakan contoh dokumen dari bagian sebelumnya, kemungkinan bahwa Anda akan memiliki koleksi untuk `reading_material` dan `office_supplies`. Merupakan tanggung jawab perangkat lunak Anda untuk menegakkan koleksi yang memiliki dokumen tersebut.

Contoh berikut ini menggunakan MongoDB API untuk menunjukkan bagaimana untuk menambahkan, kueri, pembaruan, dan menghapus dokumen.

Topik

- Menambahkan dokumen
- Menanyakan dokumen
- Memperbarui dokumen
- Menghapus dokumen

Menambahkan dokumen

Di Amazon DocumentDB, basis data dibuat saat pertama kali Anda menambahkan dokumen ke koleksi. Dalam contoh ini, Anda membuat koleksi yang diberi nama `example` di basis data `test`, yang merupakan basis data default ketika Anda terhubung ke klaster. Karena koleksi secara implisit dibuat ketika dokumen pertama dimasukkan, tidak ada pemeriksaan kesalahan nama koleksi. Oleh karena itu, salah ketik dalam nama koleksi, seperti `eexample` alih-alih `example`, akan membuat dan menambahkan dokumen ke koleksi `eexample` alih-alih ke koleksi yang dimaksudkan. Pemeriksaan kesalahan harus ditangani oleh aplikasi Anda.

Contoh berikut ini menggunakan API MongoDB untuk menambahkan dokumen.

Topik

- Menambahkan satu dokumen
- Menambahkan beberapa dokumen

Menambahkan satu dokumen

Untuk menambahkan dokumen tunggal ke koleksi, gunakan operasi `insertOne( {} )` dengan dokumen yang ingin Anda tambahkan ke koleksi.

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

Keluaran dari operasi ini terlihat seperti berikut ini (format JSON).

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

Menambahkan beberapa dokumen

Untuk menambahkan beberapa dokumen ke koleksi, gunakan operasi `insertMany( [{},..., {}] )` dengan daftar dokumen yang ingin Anda tambahkan ke koleksi. Meskipun dokumen dalam daftar khusus ini memiliki skema yang berbeda, mereka semua dapat ditambahkan ke koleksi yang sama.

```
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
            }
        }
    ]
)
```

Keluaran dari operasi ini terlihat seperti berikut ini (format JSON).

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

Menanyakan dokumen

Terkadang, Anda mungkin perlu mencari inventaris toko online Anda sehingga pelanggan dapat melihat dan membeli apa yang Anda jual. Melakukan kueri sebuah koleksi relatif mudah, apakah Anda ingin semua dokumen dalam koleksi atau hanya dokumen yang memenuhi kriteria tertentu.

Untuk melakukan kueri dokumen, gunakan operasi `find()`. Perintah `find()` memiliki parameter dokumen tunggal yang mendefinisikan kriteria yang akan digunakan dalam memilih dokumen yang akan dikembalikan. Output dari `find()` adalah dokumen yang diformat sebagai satu baris teks tanpa jeda baris. Untuk memformat dokumen output agar lebih mudah dibaca, gunakan `find().pretty()`. Semua contoh dalam topik ini menggunakan `.pretty()` untuk memformat keluaran.

Gunakan empat dokumen yang Anda masukkan ke dalam koleksi `example` dalam dua latihan sebelumnya — `insertOne()` dan `insertMany()`.

Topik

- Mengambil semua dokumen dalam koleksi
- Mengambil dokumen yang cocok dengan nilai bidang
- Mengambil dokumen yang cocok dengan dokumen yang disematkan
- Mengambil dokumen yang cocok dengan nilai bidang dalam dokumen yang disematkan
- Mengambil dokumen yang cocok dengan array
- Mengambil dokumen yang cocok dengan nilai dalam array
- Mengambil dokumen menggunakan operator

Mengambil semua dokumen dalam koleksi

Untuk mengambil semua dokumen dalam koleksi Anda, gunakan operasi `find()` dengan dokumen kueri yang kosong.

Kueri berikut mengembalikan semua dokumen di dalam koleksi `example`.

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

Mengambil dokumen yang cocok dengan nilai bidang

Untuk mengambil semua dokumen yang cocok dengan bidang dan nilai, gunakan operasi `find()` dengan dokumen kueri yang mengidentifikasi bidang dan nilai untuk mencocokkan.

Menggunakan dokumen sebelumnya, kueri ini mengembalikan semua dokumen di mana bidang "Item" sama dengan "Pena".

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

Mengambil dokumen yang cocok dengan dokumen yang disematkan

Untuk menemukan semua dokumen yang cocok dengan dokumen tersemat, gunakan operasi `find()` dengan dokumen kueri yang menentukan nama dokumen tersemat dan semua bidang dan nilai untuk dokumen tersemat.

Saat mencocokkan dokumen yang ditanamkan, dokumen yang ditanamkan tersebut harus memiliki nama yang sama seperti dalam kueri. Selain itu, bidang dan nilai dalam dokumen yang ditanamkan harus cocok dengan kueri.

Kueri berikut hanya mengembalikan dokumen "Cat Poster". Ini karena "Pena" memiliki nilai yang berbeda untuk "`OnHand`" dan "`MinOnHand`", dan "Cat Semprot" memiliki satu bidang (`OrderQnty`) lebih banyak daripada dokumen kueri.

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

Mengambil dokumen yang cocok dengan nilai bidang dalam dokumen yang disematkan

Untuk menemukan semua dokumen yang cocok dengan dokumen tersemat, gunakan operasi `find()` dengan dokumen kueri yang menentukan nama dokumen tersemat dan semua bidang dan nilai untuk dokumen tersemat.

Mengingat dokumen sebelumnya, kueri berikut menggunakan "notasi titik" untuk menentukan dokumen yang ditanamkan dan bidang yang diminati. Setiap dokumen yang cocok dengan ini dikembalikan, terlepas dari apa bidang lain mungkin hadir dalam dokumen tertanam. Kueri mengembalikan "Cat Poster" dan "Cat Semprot" karena keduanya cocok dengan bidang dan nilai yang ditentukan.

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

Mengambil dokumen yang cocok dengan array

Untuk menemukan semua dokumen yang cocok dengan susunan, gunakan operasi `find()` dengan nama susunan yang menjadi perhatian Anda dan semua nilai dalam susunan tersebut. Kueri mengembalikan semua dokumen yang memiliki array dengan nama itu di mana nilai arraynya identik dengan dan dalam urutan yang sama seperti dalam kueri.

Kueri berikut hanya mengembalikan "Pena" karena "Cat Poster" memiliki warna tambahan (Putih), dan "Cat Semprot" memiliki warna dalam urutan yang berbeda.

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

Mengambil dokumen yang cocok dengan nilai dalam array

Untuk menemukan semua dokumen yang memiliki nilai susunan tertentu, gunakan operasi `find()` dengan nama susunan dan nilai yang menjadi perhatian Anda.

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

Operasi sebelumnya mengembalikan ketiga dokumen karena masing-masing memiliki array bernama `Colors` dan nilai "`Red`" di suatu tempat dalam array. Jika Anda menentukan nilai "`White`", kueri hanya akan mengembalikan "Cat Poster."

Mengambil dokumen menggunakan operator

Kueri berikut ini mengembalikan semua dokumen di mana nilai "`Inventory.OnHand`" adalah kurang dari 50.

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

Untuk daftar operator kueri yang didukung, lihat Operator kueri dan proyeksi.

Memperbarui dokumen

Biasanya, dokumen Anda tidak statis dan diperbarui sebagai bagian dari alur kerja aplikasi Anda. Contoh berikut ini menunjukkan beberapa cara yang dapat lakukan Anda untuk memperbarui dokumen.

Untuk memperbarui dokumen yang ada, gunakan operasi `update()`. Operasi `update()` memiliki dua parameter dokumen. Dokumen pertama mengidentifikasi dokumen atau dokumen-dokumen mana yang akan diperbarui. Dokumen kedua menentukan pembaruan yang akan dilakukan.

Ketika Anda memperbarui bidang yang ada — apakah bidang itu adalah bidang sederhana, susunan, atau dokumen tersemat -— Anda harus menentukan nama bidang dan nilainya. Pada akhir operasi, seolah-olah bidang dalam dokumen lama telah digantikan oleh bidang dan nilai baru.

Topik

- Memperbarui nilai bidang yang ada
- Menambahkan bidang baru
- Mengganti dokumen yang disematkan

- Memasukkan bidang baru ke dalam dokumen yang disematkan
- Menghapus bidang dari dokumen
- Menghapus bidang dari beberapa dokumen

Memperbarui nilai bidang yang ada

Gunakan empat dokumen berikut ini yang Anda tambahkan sebelumnya untuk operasi pembaruan berikut ini.

```
{
    "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
    }
```

```
}
```

Untuk memperbarui bidang sederhana

Untuk memperbarui bidang sederhana, gunakan `update()` dengan `$set` untuk menentukan nama bidang dan nilai baru. Contoh berikut ini mengubah `Item` dari “Pen” ke “Gel Pen”.

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

Hasil dari operasi ini terlihat seperti berikut ini.

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

Untuk memperbarui susunan

Contoh berikut ini menggantikan susunan warna yang ada dengan susunan baru yang mencakup `Orange` dan menghapus `White` dari daftar warna. Daftar warna baru adalah dalam urutan yang ditentukan dalam operasi `update()`.

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

Hasil dari operasi ini terlihat seperti berikut ini.

```
{
    "Item": "Poster Paint",
    "Colors": ["Red","Green","Blue","Orange","Black"],
    "Inventory": {
```

```
        "OnHand": 47,
        "MinOnHand": 50
    }
}
```

Menambahkan bidang baru

Untuk mengubah dokumen dengan menambahkan satu atau lebih bidang baru, gunakan operasi `update()` dengan dokumen kueri yang mengidentifikasi dokumen yang akan dimasukkan dan bidang dan nilai baru yang akan dimasukkan menggunakan operator `$set`.

Contoh berikut ini menambahkan bidang `UnitPrice` dengan nilai `3.99` ke dokumen Spray Paints. Perhatikan bahwa nilai `3.99` adalah numerik dan bukan string.

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

Hasil dari operasi ini terlihat seperti berikut ini (format JSON).

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

Mengganti dokumen yang disematkan

Untuk mengubah dokumen dengan mengganti dokumen tersemat, gunakan operasi `update()` dengan dokumen yang mengidentifikasi dokumen tersemat dan bidang dan nilai barunya menggunakan operator `$set`.

Mengingat dokumen berikut.

```
db.example.insert({
```

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

Untuk mengganti dokumen tersemat

Contoh berikut ini menggantikan dokumen Tanggal saat ini dengan yang dokumen baru yang hanya memiliki bidang `Month` dan `Day`; `Year` telah dihilangkan.

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

Hasil dari operasi ini terlihat seperti berikut ini.

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

Memasukkan bidang baru ke dalam dokumen yang disematkan

Untuk menambahkan bidang ke dokumen tersemat

Untuk mengubah dokumen dengan menambahkan satu atau lebih bidang baru ke dokumen tersemat, gunakan operasi `update()` dengan dokumen yang mengidentifikasi dokumen tersemat dan “notasi dot” untuk menentukan dokumen tersemat dan bidang dan nilai baru yang akan dimasukkan menggunakan operator `$set`.

Mengingat dokumen berikut ini, kode berikut ini menggunakan “notasi dot” untuk memasukkan bidang `Year` dan `DoW` ke dokumen `Date`yang disematkan, dan `Words` ke dalam dokumen induk.

```
{
```

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

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

Hasil dari operasi ini terlihat seperti berikut ini.

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

Menghapus bidang dari dokumen

Untuk mengubah dokumen dengan menghapus bidang dari dokumen, gunakan operasi `update()` dengan dokumen kueri yang mengidentifikasi dokumen yang akan dihapus bidangnya, dan operator `$unset` untuk menentukan bidang yang akan dihapus.

Contoh berikut ini akan menghapus bidang `Words`dari dokumen sebelumnya.

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

Hasil dari operasi ini terlihat seperti berikut ini.

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

Menghapus bidang dari beberapa dokumen

Untuk mengubah dokumen dengan menghapus bidang dari beberapa dokumen, gunakan operasi `update()` dengan operator `$unset` dan `multi` diatur ke `true`.

Contoh berikut ini akan menghapus bidang `Inventory` dari semua dokumen dalam koleksi contoh. Jika dokumen tidak memiliki bidang `Inventory`, tidak ada tindakan yang dilakukan pada dokumen tersebut. Jika `multi: true` dihilangkan, tindakan dilakukan hanya pada dokumen pertama yang memenuhi kriteria.

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

Menghapus dokumen

Untuk menghapus dokumen dari basis data Anda, gunakan operasi `remove()`, yang menentukan dokumen yang akan dihapus. Kode berikut ini menghapus “Gel Pen” dari koleksi `example` Anda.

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

Untuk menghapus semua dokumen dari database Anda, gunakan `remove()` operasi dengan kueri kosong.

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

Memulai Amazon DocumentDB

Ada banyak cara untuk terhubung dan memulai dengan Amazon DocumentDB. Panduan ini adalah cara tercepat, paling sederhana, dan termudah bagi pengguna untuk mulai menggunakan database dokumen kami yang kuat. Panduan ini digunakan AWS CloudShelluntuk menghubungkan dan menanyakan cluster Amazon DocumentDB Anda langsung dari. Konsol Manajemen AWS Pelanggan baru yang memenuhi syarat untuk Tingkat AWS Gratis dapat menggunakan Amazon DocumentDB CloudShell dan gratis. Jika AWS CloudShell lingkungan Anda atau klaster Amazon DocumentDB menggunakan sumber daya di luar tingkat gratis, Anda akan dikenakan tarif AWS normal untuk sumber daya tersebut. Panduan ini akan membantu Anda memulai Amazon DocumentDB dalam waktu kurang dari lima menit.

> **Note**
>
> Petunjuk dalam panduan ini khusus untuk membuat dan menghubungkan ke cluster berbasis instans Amazon DocumentDB tempat Amazon DocumentDB dan tersedia. AWS CloudShell
>
> - Jika Anda ingin membuat dan terhubung ke cluster elastis Amazon DocumentDB, lihat. Memulai dengan cluster elastis Amazon DocumentDB
> - Jika Anda berlokasi di Wilayah AWS Tiongkok, Hubungkan Amazon EC2 secara otomatis lihat.

Topik

- Prasyarat
- Langkah 1: Buat cluster
- Langkah 2: Connect ke cluster Anda
- Langkah 3: Masukkan dan kueri data
- Langkah 4: Jelajahi

Prasyarat

Sebelum membuat klaster Amazon DocumentDB pertama, Anda harus melakukan hal berikut:

Buatlah akun Amazon Web Services (AWS)

Sebelum Anda dapat mulai menggunakan Amazon DocumentDB, Anda harus memiliki akun Amazon Web Services (AWS). AWS Akun ini gratis. Anda hanya membayar untuk layanan dan sumber daya yang Anda gunakan.

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar untuk Akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.
2. Ikuti petunjuk online.

   Bagian dari prosedur pendaftaran melibatkan menerima panggilan telepon atau pesan teks dan memasukkan kode verifikasi pada keypad telepon.

   Saat Anda mendaftar untuk sebuah Akun AWS, sebuah Pengguna root akun AWSdibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya di akun. Sebagai praktik keamanan terbaik, tetapkan akses administratif ke pengguna, dan gunakan hanya pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

Siapkan izin yang diperlukan AWS Identity and Access Management (IAM).

Akses untuk mengelola sumber daya Amazon DocumentDB seperti cluster, instance, dan grup parameter cluster memerlukan kredensional yang dapat digunakan untuk AWS mengautentikasi permintaan Anda. Untuk informasi selengkapnya, lihat Identity and Access Management untuk Amazon DocumentDB.

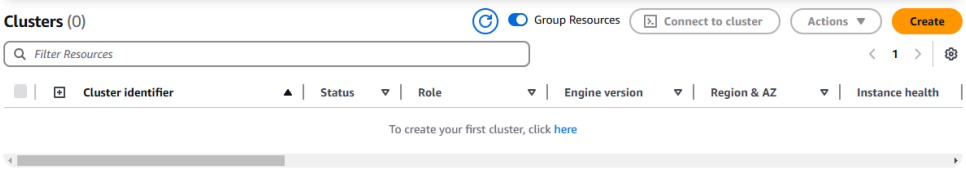
1. Di bilah pencarian Konsol Manajemen AWS, ketik IAM dan pilih IAM di menu tarik-turun yang muncul.
2. Setelah Anda berada di konsol IAM, pilih Pengguna dari panel navigasi.
3. Pilih nama pengguna Anda.
4. Pilih Tambahkan izin.
5. Pilih Lampirkan kebijakan secara langsung.
6. Ketik `AmazonDocDBFullAccess` di bilah pencarian dan pilih itu setelah muncul di hasil pencarian.
7. Pilih Berikutnya.
8. Pilih Tambahkan izin.

> **Note**
>
> AWS Akun Anda menyertakan VPC default di setiap Wilayah. Jika Anda memilih untuk menggunakan VPC Amazon, selesaikan langkah-langkah dalam topik Buat VPC Amazon di Panduan Pengguna Amazon VPC.

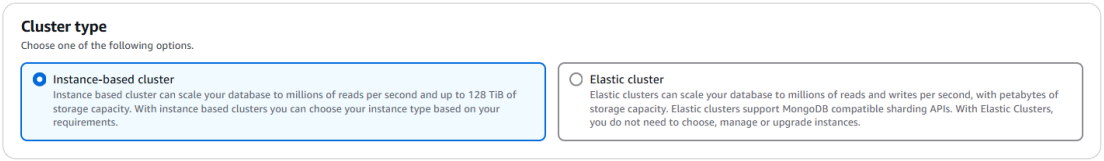
Langkah 1: Buat cluster

Pada langkah ini Anda akan membuat cluster Amazon DocumentDB.

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di konsol manajemen Amazon DocumentDB, di bawah Klaster, pilih Buat.



3. Pada halaman cluster Create Amazon DocumentDB, di bagian tipe Cluster, Instance-based pilih cluster (ini adalah opsi default).

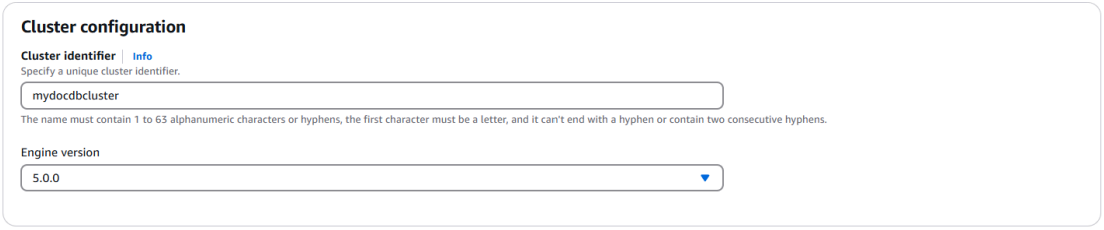


> **Note**
>
> Pilihan lain dalam kategori ini adalah cluster elastis. Untuk mempelajari selengkapnya tentang gugus elastis Amazon DocumentDB, lihat Menggunakan cluster elastis Amazon DocumentDB

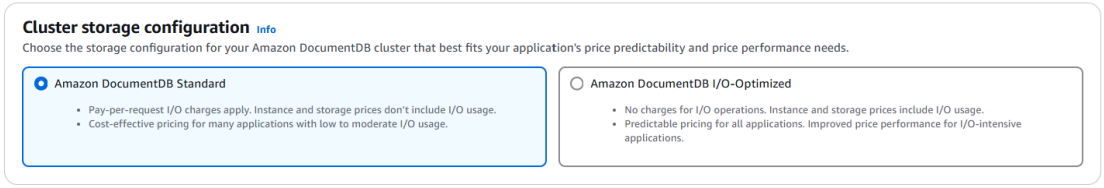
4. Di bagian konfigurasi Cluster:

   a. Untuk pengidentifikasi Cluster, masukkan nama unik, seperti**`mydocdbcluster`**. Perhatikan bahwa konsol akan mengubah semua nama cluster menjadi huruf kecil terlepas dari bagaimana mereka dimasukkan.

   b. Untuk versi Engine, pilih 5.0.0.



5. Di bagian konfigurasi penyimpanan Cluster, pilih Amazon DocumentDB Standard (ini adalah opsi default).



> **Note**
>
> Opsi lain dalam kategori ini adalah Amazon I/O-Optimized DocumentDB. Untuk mempelajari lebih lanjut tentang salah satu opsi, lihat Konfigurasi penyimpanan cluster Amazon DocumentDB

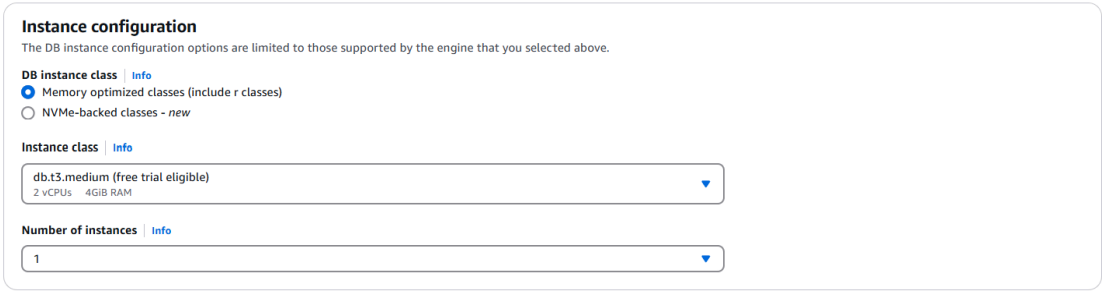
6. Di bagian konfigurasi Instans:

   a. Untuk kelas instans DB, pilih kelas yang dioptimalkan memori (termasuk kelas r) (ini default).

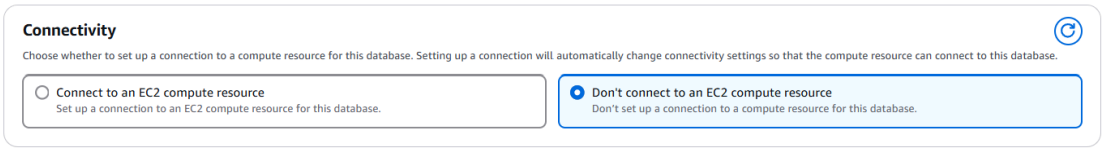
      Pilihan contoh lainnya adalah NVMe-backed kelas. Untuk mempelajari selengkapnya, lihat NVMe-backed contoh.

   b. Untuk kelas Instance, pilih db.t3.medium. Ini memenuhi syarat untuk uji coba AWS gratis.

   c. Untuk Jumlah instance, pilih 1 instance. Memilih satu instans membantu meminimalkan biaya. Jika ini adalah sistem produksi, kami sarankan Anda menyediakan tiga instans untuk ketersediaan tinggi.

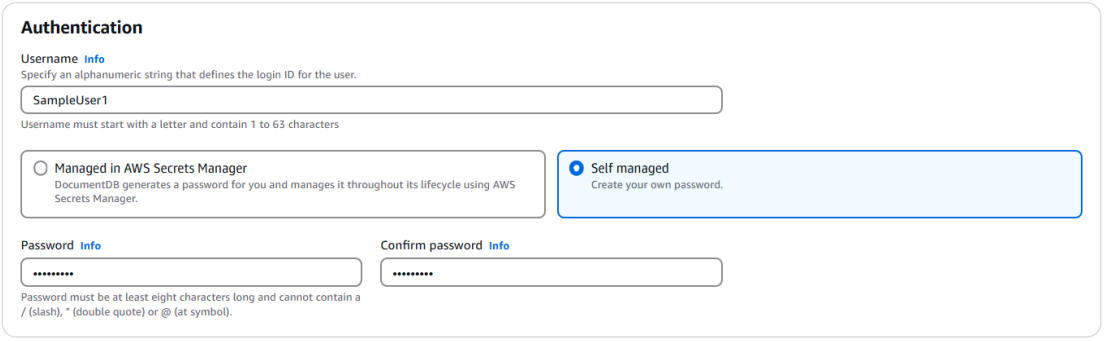


7. Di bagian Konektivitas, biarkan pengaturan default Jangan sambungkan ke sumber daya komputasi EC2.



8. Di bagian Autentikasi, masukkan nama pengguna untuk pengguna utama, lalu pilih Self managed. Masukkan kata sandi, lalu konfirmasikan.

   Jika Anda memilih Managed in AWS Secrets Manager, lihat Manajemen kata sandi dengan Amazon DocumentDB dan AWS Secrets Manager untuk informasi selengkapnya.



9. Biarkan semua opsi lain sebagai default dan pilih Buat cluster.

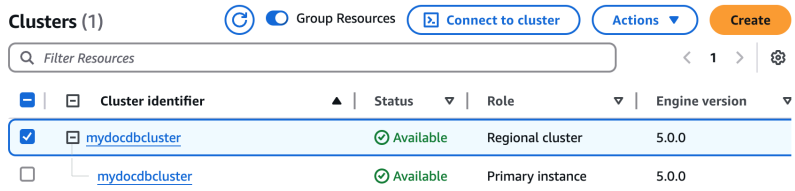
Amazon DocumentDB sekarang menyediakan klaster Anda, dan penyelesaian proses ini dapat memakan waktu hingga beberapa menit.

> **Note**
>
> Untuk informasi tentang nilai status klaster, lihat Nilai status cluster di bagian Memantau Amazon DocumentDB.

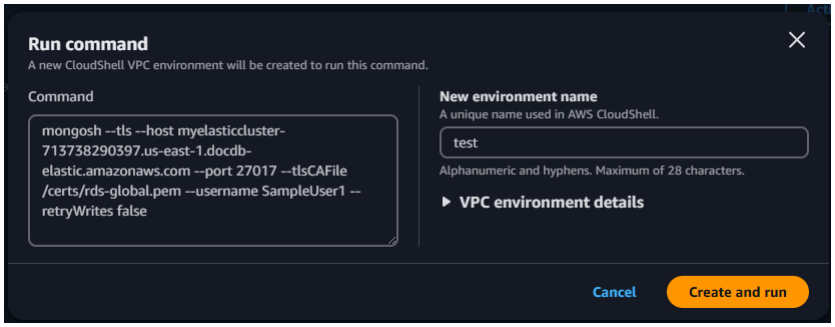
Langkah 2: Connect ke cluster Anda

Connect ke cluster Amazon DocumentDB Anda menggunakan. AWS CloudShell

1. Di konsol manajemen Amazon DocumentDB, di bawah Clusters, temukan klaster yang Anda buat. Pilih klaster Anda dengan mengklik kotak centang di sebelahnya.



2. Pilih Connect to cluster (yang berada di sebelah menu dropdown Actions). Tombol ini diaktifkan hanya setelah Anda mengklik kotak centang di sebelah klaster Anda, dan status cluster regional dan instance utama ditampilkan sebagai Tersedia. Layar perintah CloudShell Jalankan muncul.
3. Di bidang Nama lingkungan baru, masukkan nama unik, seperti “uji” dan pilih Buat dan jalankan. Detail lingkungan VPC secara otomatis dikonfigurasi untuk database Amazon DocumentDB Anda.



4. Saat diminta, masukkan kata sandi yang Anda buat di Langkah 1: Buat klaster Amazon DocumentDB (sub-langkah 7).



Setelah Anda memasukkan kata sandi dan prompt Anda menjadi`rs0 [direct: primary] <env-name>>`, Anda berhasil terhubung ke cluster Amazon DocumentDB Anda.

> **Note**
>
> Untuk informasi pemecahan masalah, lihat Memecahkan Masalah Amazon DocumentDB.

Langkah 3: Masukkan dan kueri data

Sekarang setelah Anda terhubung ke klaster, Anda dapat menjalankan beberapa kueri untuk mengenali penggunaan basis data dokumen.

1. Untuk memasukkan satu dokumen, masukkan hal berikut ini:

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

   Anda mendapatkan output berikut ini:

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

2. Anda dapat membaca dokumen yang Anda tulis dengan perintah `findOne()` (karena hanya mengembalikan satu dokumen). Masukkan yang berikut ini:

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

   Anda mendapatkan output berikut ini:

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

3. Untuk mengajukan beberapa kueri lagi, pertimbangkan kasus penggunaan profil game. Pertama, masukkan beberapa entri ke dalam koleksi berjudul `profiles`. Masukkan yang berikut ini:

   ```
   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 }
   ])
   ```

Anda mendapatkan output berikut ini:

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

4. Gunakan perintah `find()` untuk mengembalikan semua dokumen dalam koleksi profil. Masukkan berikut ini:

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

Anda akan mendapatkan output yang akan cocok dengan data yang Anda ketik di Langkah 3.

5. Gunakan kueri untuk satu dokumen dengan menggunakan filter. Masukkan yang berikut ini:

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

Anda mendapatkan output berikut ini:

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

6. Sekarang mari kita coba untuk menemukan profil dan memodifikasinya dengan menggunakan perintah `findAndModify`. Kami akan memberi pengguna Matt tambahan 10 poin dengan kode berikut:

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

Anda mendapatkan output berikut (perhatikan bahwa skornya belum meningkat):

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

7. Anda dapat memverifikasi bahwa skornya telah berubah dengan kueri berikut:

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

Anda mendapatkan output berikut ini:

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

Langkah 4: Jelajahi

Selamat! Anda telah berhasil menyelesaikan panduan Memulai untuk klaster berbasis instans Amazon DocumentDB.

Apa selanjutnya? Pelajari cara memanfaatkan basis data ini sepenuhnya dengan beberapa fitur populernya:

- Mengelola Amazon DocumentDB
- Penskalaan
- Mencadangkan dan memulihkan

> **Note**
>
> Cluster yang Anda buat dari latihan memulai ini akan terus bertambah biaya kecuali Anda menghapusnya. Untuk mengetahui petunjuknya, lihat Menghapus Klaster Amazon DocumentDB.

Fitur dan Konfigurasi Amazon DocumentDB

Tabel berikut membandingkan fitur, kemampuan, dan jenis instans yang didukung di seluruh versi engine Amazon DocumentDB dan cluster elastis. Gunakan tabel ini untuk menentukan versi mesin mana yang paling sesuai dengan kebutuhan beban kerja Anda.

Status mesin

Status mesin berdasarkan versi

Fitur/Kemampuan	v3.6	v4.0	v5.0	v8.0	
Status Support	EOL: Mar 2026	Aktif	Aktif (LTS tersedia)	Aktif (GA Nov 2025)	
Extended Support tersedia	Ya (dibayar, hingga Mar 2029)	N/A	N/A	N/A	
Rilis LTS tersedia	Tidak	Tidak	Ya (v5.0 LTS)	Tidak	
Cluster Elastis	Tidak	Tidak	Ya	Tidak	

Jenis cluster

Jenis cluster berdasarkan versi

Jenis Cluster	v3.6	v4.0	v5.0	v8.0	Cluster Elastis
Instance-based kluster	Ya	Ya	Ya	Ya	Ya
Instans tanpa server	Tidak	Tidak	Ya	Ya	Tidak

Fitur

Fitur dan Konfigurasi berdasarkan Versi Mesin

Fitur	v3.6	v4.0	v5.0	v8.0	Cluster Elastis
Klaster Global	Tidak	Ya	Ya	Ya	Tidak
Multi-AZ (3 AZ)	Ya	Ya	Ya	Ya	Ya (per pecahan)
Baca replika yang didukung	15	15	15	15	15 per pecahan
Transaksi ACID	Tidak	Ya	Ya	Ya	Tidak
Mengelola aliran	Ya	Ya	Ya	Ya	Tidak
Pencarian vektor	Tidak	Tidak	Ya	Ya	Tidak
Wawasan Performa	Ya	Ya	Ya	Ya	Tidak
In-place upgrade versi utama (MVU)	Ya (ke 5.0)	Ya (ke 5.0)	Ya (ke 8,0)	Tidak (hanya target)	Tidak
Client-side enkripsi tingkat lapangan (FLE)	Tidak	Tidak	Ya	Ya	Tidak
I/O-Optimized penyimpanan	Tidak	Tidak	Ya	Ya	Tidak
Rotasi sertifikat TLS (tidak ada reboot)	Ya (patch 1.0.208662+)	Ya (tambalan 2.0.10179+)	Ya (patch 3.0.4780+)	Ya	Ya
Cross-Region salinan snapshot	Ya	Ya	Ya	Ya	Ya

Fitur	v3.6	v4.0	v5.0	v8.0	Cluster Elastis
Cross-account berbagi snapshot	Ya	Ya	Ya	Ya	Tidak
Audit dan log kueri lambat	Ya	Ya	Ya	Ya	Ya
Dual-stack IPv4/IPv6	Tidak	Ya (tambalan 2.0.11747 +)	Ya (tambalan 3.0.15902 +)	Ya	N/A
Kolasi	Tidak	Tidak	Tidak	Ya	Tidak
Tampilan	Tidak	Tidak	Tidak	Ya	Tidak
Dukungan Kompresi	Tidak	Tidak	LZ4	LZ4/Ztsd	Tidak
Indeks Teks	Tidak	Tidak	V1	V2	Tidak
GovCloudFedRAMP ()	Tidak	Tidak	Ya	Ya	Tidak

Tipe instans

Dukungan jenis instans berdasarkan versi

Tipe Instans	v3.6	v4.0	v5.0	v8.0	
db.r4 (generasi sebelumnya)	Ya	Tidak	Tidak	Tidak	
db.r5	Ya	Ya	Ya	Ya	
db.r6g (Graviton2)	Tidak	Ya	Ya	Ya	
db.r6gd (Graviton2) NVMe-backed	Tidak	Tidak	Ya	Ya	

Tipe Instans	v3.6	v4.0	v5.0	v8.0	
db.r8g (Graviton4)	Tidak	Tidak	Ya	Ya	
db.t3.medium	Ya	Ya	Ya	Ya	
db.t4g.sedang (Graviton 2)	Tidak	Ya	Ya	Ya	
nirserver	Tidak	Tidak	Ya	Ya	

AWS integrasi

AWS dukungan integrasi berdasarkan versi

AWS Integrasi/Fitur	v3.6	v4.0	v5.0	v8.0	Cluster Elastis
AWS Backup	Ya	Ya	Ya	Ya	Tidak
AWS Secrets Manager	Ya	Ya	Ya	Ya	Tidak
AWS Lambda (ESM asli)	Tidak	Ya	Ya	Tidak	Tidak
OpenSearch Konsumsi Layanan Amazon () Zero-ETL	Tidak	Ya	Ya	Ya	Tidak
AWS DMS dukungan migrasi	Ya	Ya	Ya	Ya (dari 5.0 via AWS DMS)	Ya
AWS Identity and Access Management otentikasi	Tidak	Tidak	Ya (berbasis instans 5.0)	Ya	Tidak
AWS CloudShell	Tidak	Tidak	Ya	Ya	Tidak

AWS Integrasi/Fitur	v3.6	v4.0	v5.0	v8.0	Cluster Elastis
AWS Key Management Service	Ya	Ya	Ya	Ya	Ya
AWS CloudTrail	Ya	Ya	Ya	Ya	Ya

Amazon DocumentDB mulai cepat menggunakan CloudFormation

Bagian ini berisi langkah-langkah dan informasi lain untuk membantu Anda memulai dengan cepat menggunakan Amazon DocumentDB (dengan kompatibilitas MongoDB) menggunakan. CloudFormation Untuk informasi umum tentang Amazon DocumentDB, lihat. Apa itu Amazon DocumentDB (dengan kompatibilitas MongoDB)

Instruksi ini menggunakan AWS CloudFormation template untuk membuat cluster dan instance di Amazon VPC default Anda. Untuk petunjuk tentang membuat sumber daya ini sendiri, lihat Memulai Amazon DocumentDB.

> **Important**
>
> CloudFormation Tumpukan yang dibuat oleh template ini menciptakan beberapa sumber daya, termasuk sumber daya di Amazon DocumentDB (misalnya, cluster dan instance) dan Amazon Elastic Compute Cloud (misalnya, grup subnet).
> Beberapa sumber daya ini bukanlah sumber daya tingkat gratis. Untuk informasi selengkapnya, lihat Harga Amazon DocumentDB dan Harga Amazon EC2. Anda dapat menghapus tumpukan ketika Anda selesai dengannya untuk menghentikan biaya apa pun.

CloudFormation Tumpukan ini ditujukan untuk tujuan tutorial saja. Jika Anda menggunakan templat ini untuk lingkungan produksi, kami sarankan Anda menggunakan kebijakan IAM dan keamanan yang lebih ketat. Untuk informasi tentang mengamankan sumber daya, lihat Keamanan Amazon VPC dan Jaringan dan Keamanan Amazon EC2.

Topik

- Prasyarat
- Meluncurkan Amazon DocumentDB CloudFormation tumpukan
- Mengakses cluster Amazon DocumentDB
- Perlindungan penghentian dan perlindungan penghapusan

Prasyarat

Sebelum Anda membuat cluster Amazon DocumentDB, Anda harus memiliki yang berikut:

- Amazon VPC default
- Izin IAM yang diperlukan

Izin IAM yang Diperlukan

Izin berikut ini memungkinkan Anda untuk membuat sumber daya untuk tumpukan CloudFormation :

AWS Kebijakan Terkelola

- `AWSCloudFormationReadOnlyAccess`
- `AmazonDocDBFullAccess`

Izin IAM Tambahan

Kebijakan berikut menguraikan izin tambahan yang diperlukan untuk membuat dan menghapus tumpukan ini CloudFormation .

Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi sumber daya Anda.

Pasangan Kunci Amazon EC2

Anda harus memiliki key pair (dan file PEM) yang tersedia di Region tempat Anda akan membuat CloudFormation tumpukan. Jika Anda perlu membuat key pair, lihat Membuat Pasangan Kunci Menggunakan Amazon EC2 di Panduan Pengguna Amazon EC2.

Meluncurkan Amazon DocumentDB CloudFormation tumpukan

Bagian ini menguraikan cara untuk memulai dan mengonfigurasi tumpukan CloudFormation Amazon DocumentDB.

1. Masuk ke Konsol Manajemen AWS athttps://console.aws.amazon.com/.
2. Tabel berikut mencantumkan templat tumpukan Amazon DocumentDB untuk setiap Wilayah AWS. Pilih Launch Stack untuk tempat Wilayah AWS Anda ingin meluncurkan tumpukan Anda.

Region	Lihat Templat	Lihat di Perancang	Luncurkan
AS Timur (Ohio)	Lihat Template	Lihat di Desainer	Launch Stack
US East (N. Virginia)	Lihat Template	Lihat di Desainer	Launch Stack
US West (Oregon)	Lihat Template	Lihat di Desainer	Launch Stack
Asia Pasifik (Mumbai)	Lihat Template	Lihat di Desainer	Launch Stack
Asia Pasifik (Seoul)	Lihat Template	Lihat di Desainer	Launch Stack
Asia Pacific (Singapore)	Lihat Template	Lihat di Desainer	Launch Stack
Asia Pacific (Sydney)	Lihat Template	Lihat di Desainer	Launch Stack
Asia Pacific (Tokyo)	Lihat Template	Lihat di Desainer	Launch Stack
Kanada (Pusat)	Lihat Template	Lihat di Desainer	Launch Stack
Eropa (Frankfurt)	Lihat Template	Lihat di Desainer	Launch Stack
Eropa (Irlandia)	Lihat Template	Lihat di Desainer	Launch Stack
Eropa (London)	Lihat Template	Lihat di Desainer	Launch Stack
Eropa (Paris)	Lihat Template	Lihat di Desainer	Launch Stack

3. Buat tumpukan — Menguraikan templat Amazon DocumentDB yang Anda pilih. Setiap tumpukan didasarkan pada template - file JSON atau YAML - yang berisi konfigurasi tentang AWS sumber daya yang ingin Anda sertakan dalam tumpukan. Karena Anda memilih untuk meluncurkan

tumpukan dari template yang disediakan di atas, template Anda telah dikonfigurasi untuk membuat tumpukan Amazon DocumentDB untuk Wilayah AWS yang Anda pilih.

Saat Anda meluncurkan CloudFormation tumpukan, perlindungan penghapusan untuk klaster Amazon DocumentDB Anda dinonaktifkan secara default. Jika Anda ingin mengaktifkan perlindungan penghapusan untuk klaster Anda, selesaikan langkah-langkah berikut ini. Jika tidak, pilih Selanjutnya untuk melanjutkan ke langkah berikutnya.

Untuk mengaktifkan perlindungan penghapusan untuk klaster Amazon DocumentDB Anda:

1. Pilih Lihat di Perancang dari sudut kanan bawah halaman Buat tumpukan.
2. Ubah template menggunakan editor JSON dan YAMM terintegrasi di halaman CloudFormation Desainer konsol yang dihasilkan. Gulir ke bagian `Resources` dan ubah untuk menyertakan `DeletionProtection`, sebagai berikut. Untuk informasi selengkapnya tentang menggunakan CloudFormation Desainer, lihat Apa Itu CloudFormation Desainer? .

   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. Pilih Buat Tumpukan (



)

dari pojok kiri atas halaman untuk menyimpan perubahan Anda dan buat tumpukan dengan perubahan tersebut diaktifkan.

4. Setelah menyimpan perubahan Anda, Anda akan diarahkan ke halaman Buat tumpukan.
5. Pilih Next untuk melanjutkan.

4. Tentukan detail Tumpukan — Masukkan nama tumpukan dan parameter untuk templat Anda. Parameter ditetapkan dalam templat Anda dan mengizinkan Anda untuk memasukkan nilai kustom ketika Anda membuat atau memperbarui tumpukan.

- Di bawah Nama Tumpukan, masukkan nama untuk tumpukan Anda atau terima nama yang disediakan. Nama tumpukan dapat mencakup huruf (A—Z dan a—z), angka (0—9), dan tanda hubung (—).
- Di bawah Parameter, masukkan detail berikut ini:
  - DBClusterName— Masukkan nama untuk cluster Amazon DocumentDB Anda atau terima nama yang disediakan.

    Aturan penamaan klaster:
    - Panjangnya adalah [1—63] huruf, angka, atau tanda hubung.
    - Karakter pertama harus berupa huruf.
    - Tidak dapat diakhiri dengan tanda hubung atau berisi dua tanda hubung berurutan.
    - Harus unik untuk semua cluster di Amazon RDS, Neptunus, dan Amazon DocumentDB per, per Wilayah. Akun AWS
  - DBInstanceClass— Dari daftar drop-down, pilih kelas instance untuk cluster Amazon DocumentDB Anda.
  - DBInstanceName— Masukkan nama untuk instance Amazon DocumentDB Anda atau terima nama yang diberikan.

    Kendala penamaan instans:

- Panjangnya adalah [1—63] huruf, angka, atau tanda hubung.
- Karakter pertama harus berupa huruf.
- Tidak dapat diakhiri dengan tanda hubung atau berisi dua tanda hubung berurutan.
- Harus unik untuk semua instans di Amazon RDS, Neptunus, dan Amazon DocumentDB per, per Wilayah. Akun AWS

- MasterPassword— Kata sandi akun admin database.
- MasterUser— Nama pengguna akun admin database. MasterUser Harus dimulai dengan huruf dan hanya dapat berisi karakter alfanumerik.

Pilih Selanjutnya untuk menyimpan perubahan Anda dan melanjutkan.

5. Konfigurasi opsi tumpukan — Konfigurasi tag, izin, dan opsi tambahan tumpukan Anda.

- Tag — Tentukan pasangan tag (nilai kunci) untuk diterapkan ke sumber daya Anda di tumpukan Anda. Anda dapat menambahkan hingga 50 tag unik untuk setiap tumpukan.
- Izin — Opsional. Pilih peran IAM untuk secara eksplisit menentukan cara CloudFormation membuat, memodifikasi, atau menghapus sumber daya dalam tumpukan. Jika Anda tidak memilih peran, CloudFormation gunakan izin berdasarkan kredensyal pengguna Anda. Sebelum Anda menentukan peran layanan, pastikan bahwa Anda memiliki izin untuk memberikannya (`iam:PassRole`). Izin `iam:PassRole` menentukan peran yang dapat Anda gunakan.

  > **Note**
  >
  > Saat Anda menentukan peran layanan, CloudFormation selalu gunakan peran itu untuk semua operasi yang dilakukan pada tumpukan tersebut. Pengguna lainnya yang memiliki izin untuk melakukan operasi pada tumpukan ini akan dapat menggunakan peran ini, meskipun mereka tidak memiliki izin untuk meneruskannya. Jika peran tersebut mencakup izin yang seharusnya tidak dimiliki pengguna, Anda dapat secara tidak sengaja meningkatkan izin pengguna. Pastikan bahwa peran tersebut memberikan hak istimewa paling sedikit.

- Opsi lanjutan — Anda dapat mengatur opsi lanjutan berikut ini:

  - Kebijakan tumpukan — Opsional. Menetapkan sumber daya yang Anda ingin lindungi dari pembaruan yang tidak disengaja selama pembaruan tumpukan. Secara default, semua sumber daya dapat diperbarui selama pembaruan tumpukan.

Anda dapat memasukkan kebijakan tumpukan secara langsung sebagai JSON, atau unggah file JSON yang berisi kebijakan tumpukan. Untuk informasi selengkapnya, lihat Mencegah Pembaruan untuk Sumber Daya Tumpukan.

- Konfigurasi rollback — Opsional. Tentukan alarm CloudWatch Log CloudFormation untuk dipantau saat membuat dan memperbarui tumpukan. Jika operasi melanggar ambang alarm, putar CloudFormation kembali.
- Opsi notifikasi — Opsional. Tentukan topik untuk Simple Notification System (SNS).
- Opsi pembuatan tumpukan — Opsional. Anda dapat menentukan opsi berikut ini:
  - Rollback pada kegagalan — Apakah tumpukan harus dikembalikan jika pembuatan tumpukan gagal.
  - Waktu habis —Jumlah menit sebelum waktu pembuatan tumpukan habis.
  - Perlindungan penghentian — Mencegah tumpukan tidak sengaja dihapus.

> **Note**
>
> CloudFormation perlindungan terminasi berbeda dari konsep perlindungan penghapusan Amazon DocumentDB. Untuk informasi selengkapnya, lihat Perlindungan penghentian dan perlindungan penghapusan.

Pilih Next untuk melanjutkan.

6. Tinjauan <nama–tumpukan> — Tinjau templat, detail, dan opsi konfigurasi tumpukan Anda. Anda juga dapat membuka tautan buat cepat pada bagian bawah halaman untuk membuat tumpukan dengan konfigurasi dasar yang sama seperti yang satu ini.

   - Pilih Buat untuk membuat tumpukan.
   - Sebagai alternatif, Anda dapat memilih Buat set perubahan. Set perubahan adalah pratinjua tentang bagaimana tumpukan ini akan dikonfigurasi sebelum membuat tumpukan. Hal ini mengizinkan Anda untuk memeriksa berbagai konfigurasi sebelum menjalankan set perubahan.

Mengakses cluster Amazon DocumentDB

Setelah CloudFormation tumpukan selesai, Anda dapat menggunakan instans Amazon EC2 untuk terhubung ke cluster Amazon DocumentDB Anda. Untuk informasi tentang menghubungkan

ke instans Amazon EC2 menggunakan SSH, lihat Connect to Your Linux Instance di Panduan Pengguna Amazon EC2.

Setelah Anda terhubung, lihat bagian berikut ini, yang berisi informasi tentang menggunakan Amazon DocumentDB.

- Langkah 3: Masukkan dan kueri data
- Mengelola sumber daya Amazon DocumentDB
- Memantau Amazon DocumentDB

Perlindungan penghentian dan perlindungan penghapusan

Ini adalah praktik terbaik Amazon DocumentDB untuk mengaktifkan perlindungan penghapusan dan perlindungan penghentian. CloudFormation perlindungan terminasi adalah fitur yang sangat berbeda dari fitur perlindungan penghapusan Amazon DocumentDB.

- Perlindungan terminasi — Anda dapat mencegah tumpukan dihapus secara tidak sengaja dengan mengaktifkan perlindungan terminasi untuk CloudFormation tumpukan Anda. Jika pengguna mencoba untuk menghapus tumpukan dengan penghentian perlindungan yang diaktifkan padanya, penghapusan gagal dan tumpukan tetap tidak berubah. Perlindungan penghentian dinonaktifkan secara default saat Anda membuat tumpukan menggunakan CloudFormation. Anda dapat mengaktifkan perlindungan terminasi di tumpukan saat membuatnya. Untuk informasi selengkapnya, lihat Mengatur Opsi CloudFormation Tumpukan.
- Perlindungan penghapusan — Amazon DocumentDB juga menyediakan kemampuan untuk mengaktifkan perlindungan penghapusan untuk klaster. Jika pengguna mencoba untuk menghapus klaster Amazon DocumentDB dengan perlindungan penghapusan yang diaktifkan padanya, penghapusan gagal dan klaster tetap tidak berubah. Perlindungan penghapusan, saat diaktifkan, melindungi terhadap penghapusan yang tidak disengaja dari Amazon DocumentDB,, dan. Konsol Manajemen AWS AWS CLI CloudFormation Untuk informasi lebih lanjut tentang mengaktifkan dan menonaktifkan perlindungan penghapusan untuk klaster Amazon DocumentDB, lihat Perlindungan penghapusan.

Kompatibilitas Amazon DocumentDB dengan MongoDB

Amazon DocumentDB mendukung kompatibilitas MongoDB termasuk MongoDB 4.0, MongoDB 5.0, dan MongoDB 8.0. Kompatibilitas MongoDB berarti bahwa sebagian besar aplikasi, driver, dan alat yang sudah Anda gunakan hari ini dengan database MongoDB Anda dapat digunakan dengan Amazon DocumentDB dengan sedikit atau tanpa perubahan. Bagian ini menjelaskan semua yang perlu Anda ketahui tentang kompatibilitas Amazon DocumentDB dengan MongoDB termasuk kemampuan dan fitur baru, memulai, jalur migrasi, dan perbedaan fungsional.

Topik

- MongoDB 8.0 kompatibilitas
- Kompatibilitas MongoDB 5.0
- Kompatibilitas MongoDB 4.0

MongoDB 8.0 kompatibilitas

Topik

- Apa yang baru di Amazon DocumentDB 8.0
- Memulai Amazon DocumentDB 8.0
- Tingkatkan atau migrasi ke Amazon DocumentDB 5.0 atau 8.0
- Perbedaan fungsional

Apa yang baru di Amazon DocumentDB 8.0

Amazon DocumentDB 8.0 meningkatkan kinerja kueri hingga 7x, dan meningkatkan rasio kompresi hingga 5x, memungkinkan Anda membangun aplikasi berkinerja tinggi dengan biaya lebih rendah. Amazon DocumentDB 8.0 memiliki kompatibilitas protokol kawat penuh dengan MongoDB 8.0. Ringkasan di bawah ini memperkenalkan beberapa fitur utama yang diperkenalkan di Amazon DocumentDB 8.0. Untuk melihat daftar lengkap kemampuan baru, lihat Catatan rilis.

- Menyediakan kompatibilitas dengan MongoDB 8.0 dengan menambahkan dukungan untuk driver API MongoDB 8.0. Juga mendukung aplikasi yang dibangun menggunakan MongoDB API versi 6.0 dan 7.0.

- Perencana kueri baru di Amazon DocumentDB 8.0 memperluas peningkatan kinerja ke operator tahap agregasi, bersama dengan mendukung pengoptimalan saluran agregasi dan perintah yang berbeda.
- Amazon DocumentDB 8.0 mendukung pemeriksaan dan tampilan.
- Menawarkan 6 tahap agregasi baru: $replaceWith, $vectorSearch, $merge, $set, $unset, $bucket, dan 3 operator agregasi baru $pow, $rand, $dateTrunc.
- Versi baru indeks teks: Indeks teks v2 di Amazon DocumentDB 8.0 memperkenalkan token tambahan, meningkatkan kemampuan pencarian teks.
- Melalui pembuatan indeks vektor paralel, Amazon DocumentDB 8.0 mengurangi waktu pembuatan indeks hingga 30x.

Memulai Amazon DocumentDB 8.0

Untuk memulai Amazon DocumentDB 8.0, lihat Panduan Memulai. Anda dapat membuat klaster Amazon DocumentDB 8.0 baru menggunakan atau SDK Konsol Manajemen AWS ,, atau AWS . AWS CLI CloudFormation Saat menghubungkan ke Amazon DocumentDB, Anda harus menggunakan driver MongoDB atau utilitas yang kompatibel dengan MongoDB 5.0 atau lebih tinggi.

> **Note**
>
> Saat menggunakan AWS SDK,, atau AWS CLI CloudFormation, versi mesin akan default ke 5.0.0. Anda harus secara eksplisit menentukan parameter `engineVersion = 8.0.0` untuk membuat cluster Amazon DocumentDB 8.0 baru atau untuk `engineVersion = 4.0.0` membuat cluster Amazon DocumentDB 4.0 baru atau `engineVersion = 3.6.0` untuk membuat cluster Amazon DocumentDB 3.6 baru. Untuk klaster Amazon DocumentDB tertentu, Anda dapat menentukan versi cluster menggunakan to `describe-db-clusters` call atau menggunakan AWS CLI konsol manajemen Amazon DocumentDB untuk melihat nomor versi engine untuk klaster tertentu.

Amazon DocumentDB 5.0 dan 8.0 mendukung prosesor Amazon EC2 Graviton4 dan prosesor `r8g` Graviton2 `r6g` seperti dan jenis instans untuk cluster Anda dan tersedia di semua wilayah yang `t4.medium` didukung (lihat). Contoh Untuk informasi selengkapnya tentang harga, lihat Harga Amazon DocumentDB (dengan kompatibilitas MongoDB).

Tingkatkan atau migrasi ke Amazon DocumentDB 5.0 atau 8.0

Anda dapat bermigrasi dari MongoDB 3.6 atau MongoDB 4.0 ke Amazon DocumentDB 5.0 atau Amazon DocumentDB 8.0 menggunakan atau utilitas seperti,,, dan. AWS DMS`mongodump` `mongorestore` `mongoimport` `mongoexport` Untuk petunjuk tentang cara bermigrasi, lihat Memutakhirkan cluster Amazon DocumentDB Anda menggunakan AWS Database Migration Service.

Perbedaan fungsional

Perbedaan fungsional antara Amazon DocumentDB 5.0 dan 8.0

Dengan dirilisnya Amazon DocumentDB 8.0, ada perbedaan fungsional antara Amazon DocumentDB 5.0 dan Amazon DocumentDB 8.0:

- Planner v1 adalah perencana kueri default di Amazon DocumentDB 5.0, sedangkan Planner v3 yang lebih berkinerja adalah default di Amazon DocumentDB 8.0.
- Fitur baru di Amazon DocumentDB 8.0 termasuk Views, Collation, dan operator seperti $merge hanya kompatibel dengan Planner v3.
- Kompresi diaktifkan secara default di Amazon DocumentDB 8.0 dan diatur untuk menggunakan algoritma Zstandard. Selanjutnya, di Amazon DocumentDB 8.0 'diaktifkan' bukan lagi pilihan yang valid; Anda dapat memilih dari Zstd, LZ4 dan tidak ada.
- Amazon DocumentDB 8.0 hanya mendukung TLS 1.2 dan lebih tinggi untuk koneksi database.

> **Important**
>
> Amazon DocumentDB 8.0 membutuhkan TLS 1.2 atau lebih tinggi. TLS 1.0 dan TLS 1.1 tidak lagi didukung.

Kompatibilitas MongoDB 5.0

Topik

- Apa yang baru di Amazon DocumentDB 5.0
- Apa yang baru di Amazon DocumentDB 5.0.1
- Memulai Amazon DocumentDB 5.0

- Tingkatkan atau migrasi ke Amazon DocumentDB 5.0
- Perbedaan fungsional

Apa yang baru di Amazon DocumentDB 5.0

Amazon DocumentDB 5.0 memperkenalkan fitur dan kemampuan baru yang mencakup batas penyimpanan dan enkripsi tingkat bidang sisi klien. Ringkasan di bawah ini memperkenalkan beberapa fitur utama yang diperkenalkan di Amazon DocumentDB 5.0. Untuk melihat daftar lengkap kemampuan baru, lihat Catatan rilis.

- Peningkatan batas penyimpanan menjadi 128 TiB untuk semua cluster Amazon DocumentDB berbasis instans dan cluster elastis berbasis shard.
- Memperkenalkan Amazon DocumentDB 5.0 Engine Versi 3.0.775)
  - Dukungan untuk driver MongoDB 5.0 API
  - Support untuk Client-side Field Level Encryption (FLE) Anda sekarang dapat mengenkripsi bidang di sisi klien sebelum menulis data ke cluster Amazon DocumentDB. Untuk informasi selengkapnya, lihat enkripsi tingkat Client-side bidang.
  - Operator agregasi baru:`$dateAdd`, `$dateSubtract`
  - Mendukung indeks dengan `$elemMatch` operator. Akibatnya, kueri yang memiliki `$elemMatch` akan menghasilkan pemindaian indeks.

Amazon DocumentDB tidak mendukung setiap fitur MongoDB 5.0. Ketika kami membangun Amazon DocumentDB 5.0, kami bekerja mundur dari fitur dan kemampuan yang diminta pelanggan kami untuk membangun paling banyak. Kami akan terus menambahkan kemampuan MongoDB 5.0 tambahan berdasarkan apa yang diminta pelanggan untuk kami bangun. Untuk daftar terbaru API yang didukung, lihatAPI MongoDB yang didukung, operasi, dan tipe data di Amazon DocumentDB.

Apa yang baru di Amazon DocumentDB 5.0.1

Amazon DocumentDB 5.0.1 adalah rilis versi minor pertama untuk Amazon DocumentDB 5.0 versi utama. Versi minor mencakup peningkatan seperti fitur baru, perbaikan keamanan, dan perbaikan bug. Anda dapat meningkatkan ke 5.0.1 dari 5.0.0 menggunakan upgrade versi minor. Untuk informasi selengkapnya, lihat Peningkatan versi minor Amazon DocumentDB.

- Peningkatan kemampuan agregasi dengan operator baru (`$rand`,, `$pow$dateToParts`,`$dateFromParts`).

- Metrik koneksi aktif untuk memantau instance. Batasannya dapat ditemukan di halaman Kuota dan batas Amazon DocumentDB kami.
- Metrik kinerja tingkat perintah granular CloudWatch untuk operasi CRUD (`find`,,,,,`insert`,`findAndModify`,`update`,`delete`,`aggregate`,`count`,`distinct`,, `getMore` `abortTransaction` `AvgDurationP100Duration`, dan `commitTransaction` `MaxConcurrent` dimensi untuk pemantauan operasional yang lebih baik.
- Amazon DocumentDB 5.0.1 hanya mendukung TLS 1.2 dan lebih tinggi untuk koneksi database.
- Ditambahkan`gcRuntimeStats`,`documentFragmentStats`, dan pemecahan metrik tingkat pengumpulan ke dalam `storageSegmentBase` dan ke `storageSegmentExtended` dalam `collstats` untuk memunculkan lebih banyak informasi ke dalam kesehatan keseluruhan dari proses pengumpulan sampah.
- Meningkatkan ketahanan lintas wilayah Global Cluster dengan menghilangkan reboot cluster sekunder dengan Global Survivable Read Only.

Memulai Amazon DocumentDB 5.0

Untuk memulai Amazon DocumentDB 5.0, lihat Panduan Memulai. Anda dapat membuat klaster Amazon DocumentDB 5.0 baru menggunakan atau SDK Konsol Manajemen AWS AWS CLI, AWS , atau. CloudFormation Saat menghubungkan ke Amazon DocumentDB, Anda harus menggunakan driver MongoDB atau utilitas yang kompatibel dengan MongoDB 5.0 atau lebih tinggi.

> **Note**
>
> Saat menggunakan AWS SDK,, atau AWS CLI CloudFormation, versi mesin akan default ke 5.0.0. Anda harus secara eksplisit menentukan parameter `engineVersion = 4.0.0` untuk membuat cluster Amazon DocumentDB 4.0 baru `engineVersion = 3.6.0` atau untuk membuat cluster Amazon DocumentDB 3.6 baru. Untuk klaster Amazon DocumentDB tertentu, Anda dapat menentukan versi cluster menggunakan to `describe-db-clusters` call atau menggunakan AWS CLI konsol manajemen Amazon DocumentDB untuk melihat nomor versi engine untuk klaster tertentu.

Amazon DocumentDB 5.0 dan 8.0 mendukung prosesor Amazon EC2 Graviton4 dan prosesor `r8g` Graviton2 `r6g` seperti dan jenis instans untuk cluster Anda dan tersedia di semua wilayah yang `t4.medium` didukung (lihat). Contoh Untuk informasi selengkapnya tentang harga, lihat Harga Amazon DocumentDB (dengan kompatibilitas MongoDB).

Tingkatkan atau migrasi ke Amazon DocumentDB 5.0

Anda dapat bermigrasi dari MongoDB 3.6 atau MongoDB 4.0 ke Amazon DocumentDB 5.0 menggunakan atau utilitas seperti,,, dan. AWS DMS`mongodump` `mongorestore` `mongoimport` `mongoexport` Untuk petunjuk tentang cara bermigrasi, lihat Memutakhirkan cluster Amazon DocumentDB Anda menggunakan AWS Database Migration Service.

Perbedaan fungsional

Perbedaan fungsional antara Amazon DocumentDB 4.0 dan 5.0

Dengan dirilisnya Amazon DocumentDB 5.0, ada perbedaan fungsional antara Amazon DocumentDB 4.0 dan Amazon DocumentDB 5.0:

- Peran bawaan cadangan sekarang mendukung`serverStatus`. Tindakan - Pengembang dan aplikasi dengan peran cadangan dapat mengumpulkan statistik tentang keadaan cluster Amazon DocumentDB.
- `secondaryDelaySecs`Bidang menggantikan `slaveDelay` `replSetGetConfig` output.
- helloPerintah menggantikan `isMaster` - hello mengembalikan dokumen yang menjelaskan peran cluster Amazon DocumentDB.
- Amazon DocumentDB 5.0 sekarang mendukung pemindaian indeks dengan operator di tingkat `$elemMatch` bersarang pertama. Pemindaian indeks didukung ketika filter kueri hanya memiliki satu tingkat `$elemMatch` filter tetapi tidak didukung jika `$elemMatch` kueri bersarang disertakan.

  Misalnya, di Amazon DocumentDB 5.0, jika Anda menyertakan `$elemMatch` operator di level bersarang, itu tidak akan mengembalikan nilai seperti yang terjadi di 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 ] } ] ] }
```

- Proyeksi “$” di Amazon DocumentDB 4.0 mengembalikan semua dokumen dengan semua bidang. Dengan Amazon DocumentDB 5.0, find perintah dengan proyeksi “$” mengembalikan dokumen yang cocok dengan parameter kueri yang hanya berisi bidang yang cocok dengan proyeksi “$”.
- Di Amazon DocumentDB 5.0, find perintah `$regex` dengan `$options` dan parameter kueri mengembalikan kesalahan: “Tidak dapat mengatur opsi di `$regex` keduanya dan”. `$options`
- Dengan Amazon DocumentDB 5.0`$indexOfCP`, sekarang mengembalikan “-1" ketika:
  - substring tidak ditemukan dalam ekspresi string, atau
  - awal adalah angka yang lebih besar dari akhir, atau
  - start adalah angka yang lebih besar dari panjang byte string.
- Di Amazon DocumentDB 4.0`$indexOfCP`, mengembalikan “0" ketika posisi awal adalah angka yang lebih besar dari akhir atau panjang byte string.
- Dengan Amazon DocumentDB 5.0, operasi proyeksi `_id fields` di, `{"_id.nestedField" : 1}` misalnya, mengembalikan dokumen yang hanya menyertakan bidang yang diproyeksikan. Sedangkan di Amazon DocumentDB 4.0, perintah proyeksi bidang bersarang tidak memfilter dokumen apa pun.

Kompatibilitas MongoDB 4.0

Topik

- Fitur Amazon DocumentDB 4.0
- Memulai Amazon DocumentDB 4.0
- Tingkatkan atau migrasi ke Amazon DocumentDB 4.0
- Perbedaan fungsional

Fitur Amazon DocumentDB 4.0

Amazon DocumentDB 4.0 memperkenalkan banyak fitur dan kemampuan baru yang mencakup transaksi ACID dan peningkatan untuk mengubah aliran. Ringkasan di bawah ini menunjukkan beberapa fitur utama yang diperkenalkan di Amazon DocumentDB 4.0. Untuk melihat daftar lengkap kemampuan, lihatCatatan rilis.

- Transaksi ACID: Amazon DocumentDB sekarang mendukung kemampuan untuk melakukan transaksi di beberapa dokumen, pernyataan, koleksi, dan basis data. Transaksi menyederhanakan pengembangan aplikasi dengan memungkinkan Anda untuk melakukan operasi atomik, konsisten, terisolasi, dan tahan lama (ACID) di satu atau lebih dokumen dalam klaster Amazon DocumentDB. Untuk informasi selengkapnya, lihat Transaksi di Amazon DocumentDB.
- Aliran perubahan: Anda sekarang memiliki kemampuan untuk membuka aliran perubahan di tingkat klaster (`client.watch()` atau `mongo.watch()`) dan basis data (`db.watch()`), Anda dapat menentukan `startAtOperationTime` untuk membuka kursor aliran perubahan, dan terakhir Anda sekarang dapat memperpanjang periode retensi aliran perubahan menjadi 7 hari (sebelumnya 24 jam). Untuk informasi selengkapnya, lihat Menggunakan aliran perubahan dengan Amazon DocumentDB.
- AWS Database Migration Service(AWS DMS): Anda sekarang dapat menggunakan AWS DMS untuk memigrasikan beban kerja MongoDB 4.0 Anda ke Amazon DocumentDB. AWS DMS sekarang mendukung sumber MongoDB 4.0, target Amazon DocumentDB 4.0, dan sumber Amazon DocumentDB 3.6 untuk melakukan peningkatan antara Amazon DocumentDB 3.6 dan 4.0. Untuk informasi selengkapnya, lihat Dokumentasi AWS DMS .
- Performa dan pengindeksan: Anda sekarang dapat memanfaatkan indeks dengan `$lookup`, menemukan kueri dengan proyeksi yang berisi satu bidang atau satu bidang dan bidang `_id` dapat dilayani langsung dari indeks dan tanpa perlu membaca dari koleksi (kueri tercakup), kemampuan untuk `hint()` dengan `findAndModify`, optimasi performa untuk `$addToSet`, dan peningkatan untuk mengurangi ukuran indeks secara keseluruhan. Untuk informasi selengkapnya, lihat Catatan rilis.
- Operator: Amazon DocumentDB 4.0 sekarang mendukung sejumlah operator agregasi baru: `$ifNull`, `$replaceRoot`, `$setIsSubset`, `$setIntersection`, `$setUnion`, `$setEquals`. Anda dapat melihat semua API, Operasi, dan Tipe Data MongoDB yang kami dukung di API MongoDB yang didukung, operasi, dan tipe data di Amazon DocumentDB.
- Kontrol akses berbasis peran (RBAC): Dengan perintah `ListCollection` dan `ListDatabase` Anda sekarang dapat secara opsional menggunakan parameter `authorizedCollections` dan `authorizedDatabases` untuk memungkinkan pengguna membuat daftar koleksi dan basis

data yang memiliki izin untuk mereka akses tanpa memerlukan peran `listCollections` dan `listDatabase`, masing-masing. Anda juga memiliki kemampuan untuk menghilangkan kursor Anda sendiri tanpa memerlukan peran `KillCursor`.

Amazon DocumentDB tidak mendukung setiap fitur MongoDB 4.0. Ketika kami membangun Amazon DocumentDB 4.0, kami bekerja ke belakang dari fitur dan kemampuan yang paling banyak diminta pelanggan kami untuk kami bangun. Kami akan terus menambahkan kemampuan MongoDB 4.0 tambahan berdasarkan apa yang diminta pelanggan untuk kami bangun. Misalnya, Amazon DocumentDB 4.0 saat ini tidak mendukung operator konversi tipe atau operator string yang diperkenalkan di MongoDB 4.0. Untuk daftar terbaru API yang didukung, lihat[API MongoDB yang didukung, operasi, dan tipe data di Amazon DocumentDB](#).

Memulai Amazon DocumentDB 4.0

Untuk memulai Amazon DocumentDB 4.0, lihat Panduan Memulai. Anda dapat membuat klaster Amazon DocumentDB 4.0 baru menggunakan atau SDK Konsol Manajemen AWS AWS CLI, AWS , atau. CloudFormation Saat menghubungkan ke Amazon DocumentDB, Anda harus menggunakan driver atau utilitas MongoDB yang kompatibel dengan MongoDB 4.0 atau yang lebih tinggi.

> **Note**
>
> Saat menggunakan AWS SDK,, atau AWS CLI CloudFormation, versi mesin akan default ke 5.0.0. Anda harus secara eksplisit menentukan parameter `engineVersion = 4.0.0` untuk membuat cluster Amazon DocumentDB 4.0 baru `engineVersion = 3.6.0` atau untuk membuat cluster Amazon DocumentDB 3.6 baru. Untuk klaster Amazon DocumentDB tertentu, Anda dapat menentukan versi cluster menggunakan to `describe-db-clusters` call atau menggunakan AWS CLI konsol manajemen Amazon DocumentDB untuk melihat nomor versi engine untuk klaster tertentu.

Amazon DocumentDB 4.0 `r5` mendukung`r6g`,,`t3.medium`, `t4g.medium` dan jenis instans untuk cluster Anda dan tersedia di semua wilayah yang didukung. Tidak ada biaya tambahan untuk menggunakan Amazon DocumentDB 4.0. Untuk informasi selengkapnya tentang harga, lihat Harga Amazon DocumentDB (dengan kompatibilitas MongoDB).

Tingkatkan atau migrasi ke Amazon DocumentDB 4.0

Anda dapat bermigrasi dari MongoDB 3.6 atau MongoDB 4.0 ke Amazon DocumentDB 4.0 memanfaatkan AWS DMS atau utilitas seperti `mongodump`, `mongorestore`, `mongoimport`, dan `mongoexport`. Demikian pula, Anda dapat menggunakan alat yang sama untuk meningkatkan dari Amazon DocumentDB 3.6 ke Amazon DocumentDB 4.0. Untuk petunjuk tentang cara bermigrasi, lihat Memutakhirkan cluster Amazon DocumentDB Anda menggunakan AWS Database Migration Service.

Perbedaan fungsional

Perbedaan fungsional antara Amazon DocumentDB 3.6 dan 4.0

Dengan dirilisnya Amazon DocumentDB 4.0, ada perbedaan fungsional antara Amazon DocumentDB 3.6 dan Amazon DocumentDB 4.0:

- Proyeksi untuk dokumen bersarang: Amazon DocumentDB 3.6 menganggap bidang pertama dalam dokumen bersarang ketika menerapkan proyeksi. Namun, Amazon DocumentDB 4.0 akan mengurai sub dokumen dan juga menerapkan proyeksi untuk setiap sub dokumen. Misalnya: jika proyeksinya adalah `"a.b.c": 1`, maka perilaku di kedua versi itu identik. Namun, jika proyeksinya adalah `{a:{b:{c:1}}}` maka Amazon DocumentDB 3.6 hanya akan menerapkan proyeksi ke 'a' dan bukan 'b' atau 'c'.
- Perilaku untuk **`minKey`**, **`maxKey`**: Di Amazon DocumentDB 4.0, perilaku untuk `{x: {$gt:MaxKey}}` tidak mengembalikan apa-apa, dan untuk `{x:{$lt:MaxKey}}` mengembalikan segalanya.
- Perbedaan perbandingan dokumen: Membandingkan nilai-nilai numerik dari berbagai jenis (ganda, int, panjang) di sub dokumen (misalnya, b dalam `{"_id" :1, "a" :{"b":1}}`) sekarang menyediakan output yang konsisten di seluruh tipe data numerik dan untuk setiap tingkat dokumen.

Perbedaan fungsional antara Amazon DocumentDB 4.0 dan MongoDB 4.0

Di bawah ini adalah perbedaan fungsional antara Amazon DocumentDB 4.0 dan MongoDB 4.0.

- Pencarian dengan kunci kosong di jalur: Ketika koleksi berisi dokumen dengan kunci kosong di dalam array (misalnya `{"x" : [ { "" : 10 }, { "b" : 20 } ]}`), dan ketika kunci yang digunakan dalam kueri berakhir dalam string kosong (misalnya `x.`), maka Amazon DocumentDB akan mengembalikan dokumen itu karena melintasi semua dokumen dalam array sedangkan MongoDB tidak akan mengembalikan dokumen itu.

- **`$setOnInsert`** bersama dengan **`$`** di jalur: Operator bidang `$setOnInsert` tidak akan bekerja dalam kombinasi dengan $ di jalur di Amazon DocumentDB, yang juga konsisten dengan MongoDB 4.0.

Transaksi di Amazon DocumentDB

Amazon DocumentDB (dengan kompatibilitas MongoDB) mendukung transaksi di Amazon DocumentDB 4.0 dan yang lebih baru. Anda dapat melakukan transaksi di beberapa dokumen, pernyataan, koleksi, dan basis data. Transaksi menyederhanakan pengembangan aplikasi dengan memungkinkan Anda untuk melakukan operasi atomik, konsisten, terisolasi, dan tahan lama (ACID) di satu atau lebih dokumen dalam klaster Amazon DocumentDB. Kasus penggunaan umum untuk transaksi termasuk pemrosesan keuangan, pemenuhan dan pengelolaan pesanan, dan membangun permainan multi-pemain.

Tidak ada biaya tambahan untuk transaksi. Anda hanya membayar untuk IO baca dan tulis yang Anda gunakan sebagai bagian dari transaksi.

Topik

- Persyaratan
- Praktik terbaik
- Batasan
- Pemantauan dan diagnostik
- Tingkat isolasi transaksi
- Kasus penggunaan
- Perintah yang Didukung
- Kemampuan yang tidak didukung
- Sesi
- Kesalahan transaksi

Persyaratan

Untuk menggunakan fitur transaksi, Anda perlu memenuhi persyaratan berikut ini:

- Anda harus menggunakan Amazon DocumentDB 4.0 atau yang lebih baru.
- Anda harus menggunakan driver yang kompatibel dengan MongoDB 4.0 atau yang lebih baru.

Praktik terbaik

Berikut ini adalah beberapa praktik terbaik sehingga Anda bisa mendapatkan yang paling baik menggunakan transaksi dengan Amazon DocumentDB.

- Selalu lanjutkan atau batalkan transaksi setelah selesai. Meninggalkan transaksi dalam keadaan tidak lengkap akan mengikat sumber daya basis data dan dapat menyebabkan konflik tulis.
- Disarankan untuk menyimpan transaksi ke jumlah terkecil perintah yang dibutuhkan. Jika Anda memiliki transaksi dengan beberapa pernyataan yang dapat dibagi menjadi beberapa transaksi yang lebih kecil, disarankan untuk melakukannya, untuk mengurangi kemungkinan waktu habis. Selalu bertujuan untuk membuat transaksi pendek, pembacaan yang tidak berjalan lama.

Batasan

- Amazon DocumentDB tidak mendukung kursor di dalam transaksi.
- Amazon DocumentDB tidak dapat membuat koleksi baru dalam transaksi dan tidak dapat melawan koleksi yang query/update tidak ada.
- Document-level kunci tulis tunduk pada batas waktu 1 menit, yang tidak dapat dikonfigurasi oleh pengguna.
- Perintah retryable write, retryable commit, dan retryable abort tidak didukung di Amazon DocumentDB. Jika Anda menggunakan shell mongo lama (bukan mongosh), jangan sertakan `retryWrites=false` perintah dalam string kode apa pun. Secara default, penulisan yang dapat dicoba ulang dinonaktifkan. Termasuk `retryWrites=false` dapat menyebabkan kegagalan dalam perintah baca normal.
- Setiap instans Amazon DocumentDB memiliki batas batas atas pada jumlah transaksi bersamaan yang dibuka pada instans pada satu waktu. Untuk batasnya, lihatBatas instans.
- Untuk transaksi tertentu, ukuran log transaksi harus kurang dari 32MB.
- Amazon DocumentDB tidak mendukung `count()` di dalam transaksi, tetapi tidak semua driver mendukung kemampuan ini. Alternatifnya adalah dengan menggunakan `countDocuments()` API, yang menerjemahkan kueri hitungan ke kueri agregasi pada sisi klien.
- Transaksi memiliki batas eksekusi satu menit dan sesi memiliki batas waktu 30 menit. Jika waktu transaksi habis, maka akan dibatalkan, dan perintah berikutnya dikeluarkan di dalam sesi untuk transaksi yang ada yang akan menghasilkan kesalahan berikut ini:

```
WriteCommandError({
```

```
"ok" : 0,
"operationTime" : Timestamp(1603491424, 627726),
"code" : 251,
"errmsg" : "Given transaction number 0 does not match any in-progress transactions."
}
```

Pemantauan dan diagnostik

Dengan dukungan untuk transaksi di Amazon DocumentDB 4.0, metrik CloudWatch tambahan ditambahkan untuk membantu Anda memantau transaksi Anda.

CloudWatch Metrik Baru

- `DatabaseTransactions`: Jumlah transaksi terbuka yang diambil pada periode satu menit.
- `DatabaseTransactionsAborted`: Jumlah transaksi yang dibatalkan yang diambil pada periode satu menit.
- `DatabaseTransactionsMax`: Jumlah maksimum transaksi terbuka dalam periode satu menit.
- `TransactionsAborted`: Jumlah transaksi yang dibatalkan pada instans dalam periode satu menit.
- `TransactionsCommitted`: Jumlah transaksi yang dilakukan pada instans dalam periode satu menit.
- `TransactionsOpen`: Jumlah transaksi terbuka pada instans yang diambil pada periode satu menit.
- `TransactionsOpenMax`: Jumlah maksimum transaksi terbuka pada instans dalam periode satu menit.
- `TransactionsStarted`: Jumlah transaksi yang dimulai pada instans dalam periode satu menit.

> **Note**
>
> Untuk CloudWatch metrik lainnya untuk Amazon DocumentDB, buka. Memantau Amazon DocumentDB dengan CloudWatch

Selain itu, bidang baru ditambahkan ke `currentOp` `lsid`, `transactionThreadId`, dan keadaan baru untuk transaksi "`idle transaction`" dan `serverStatus`: `currentActive`, `currentInactive`, `currentOpen`, `totalAborted`, `totalCommitted`, dan `totalStarted`.

Tingkat isolasi transaksi

Saat memulai transaksi, Anda memiliki kemampuan untuk menentukan keduanya `readConcern` dan `writeConcern` seperti yang ditunjukkan pada contoh berikut:

```
mySession.startTransaction({readConcern: {level: 'snapshot'}, writeConcern: {w: 'majority'}});
```

Untuk `readConcern`, Amazon DocumentDB mendukung isolasi snapshot secara default. Jika `readConcern` lokal, tersedia, atau mayoritas ditentukan, Amazon DocumentDB akan meng-upgrade tingkat `readConcern` untuk snapshot. Amazon DocumentDB tidak mendukung `readConcern` yang dapat dilinierkan dan menentukan kekhawatiran baca semacam itu akan mengakibatkan kesalahan.

Untuk `writeConcern`, Amazon DocumentDB mendukung mayoritas secara default dan kuorum tulis dicapai ketika empat salinan data bertahan di tiga AZ. Jika `writeConcern` lebih rendah ditentukan, Amazon DocumentDB akan meng-upgrade `writeConcern` untuk mayoritas. Selanjutnya, semua penulisan Amazon DocumentDB dijurnalkan dan penjurnalan tidak dapat dinonaktifkan.

Kasus penggunaan

Dalam bagian ini, kita akan berjalan melalui dua kasus penggunaan untuk transaksi: multi-pernyataan dan multi-koleksi.

Multi-Statement Transaksi

Transaksi Amazon DocumentDB adalah multi-pernyataan, yang berarti Anda dapat menulis transaksi yang mencakup beberapa pernyataan dengan penyimpanan atau rollback eksplisit. Anda dapat mengelompokkan tindakan `insert`, `update`, `delete`, dan `findAndModify` sebagai operasi atomik tunggal.

Kasus penggunaan umum untuk transaksi multi-pernyataan adalah transaksi debet-kredit. Sebagai contoh: Anda berutang kepada uang teman untuk membeli pakaian. Dengan demikian, Anda perlu mendebet (menarik) $500 dari akun Anda dan kredit $500 (deposit) ke akun teman Anda. Untuk melakukan operasi tersebut, Anda melakukan operasi debit dan kredit dalam satu transaksi untuk memastikan atomisitas. Melakukan hal ini akan mencegah skenario di mana $500 didebet dari akun Anda, tetapi tidak dikreditkan ke akun teman Anda. Seperti inilah kasus penggunaan ini akan terlihat:

```
// *** 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 transaksi

Transaksi kami juga bersifat multi-koleksi, yang berarti teransaksi tersebut dapat digunakan untuk melakukan beberapa operasi di dalam satu transaksi dan di beberapa koleksi. Hal ini menyediakan tampilan data yang konsisten dan menjaga integritas data Anda. Ketika Anda melakukan perintah sebagai <> tunggal, transaksi adalah eksekusi semua-atau-tidak sama sekali—dalam hal itu, transaksi semuanya berhasil maupun semuanya gagal.

Berikut adalah contoh transaksi multi-koleksi, menggunakan skenario dan data yang sama dari contoh untuk transaksi multi-pernyataan.

```
// *** 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
```

Contoh API transaksi untuk API callback

API panggilan balik hanya tersedia untuk driver 4.2+.

Javascript

Kode berikut ini menunjukkan cara untuk memanfaatkan API transaksi Amazon DocumentDB dengan 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

Kode berikut menunjukkan cara memanfaatkan API transaksi Amazon DocumentDB dengan. 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#

Kode berikut ini menunjukkan cara untuk memanfaatkan API transaksi Amazon DocumentDB dengan 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

Kode berikut ini menunjukkan cara untuk memanfaatkan API transaksi Amazon DocumentDB API dengan 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

Kode berikut menunjukkan cara untuk memanfaatkan API transaksi Amazon DocumentDB dengan 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

Kode berikut ini menunjukkan cara untuk memanfaatkan API transaksi Amazon DocumentDB dengan 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

Kode berikut ini menunjukkan cara untuk memanfaatkan API transaksi Amazon DocumentDB dengan 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

Kode berikut ini menunjukkan cara untuk memanfaatkan API transaksi Amazon DocumentDB dengan 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
```

Contoh API transaksi untuk API inti

Javascript

Kode berikut ini menunjukkan cara untuk memanfaatkan API transaksi Amazon DocumentDB dengan 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#

Kode berikut ini menunjukkan cara untuk memanfaatkan API transaksi Amazon DocumentDB dengan 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

Kode berikut ini menunjukkan cara untuk memanfaatkan API transaksi Amazon DocumentDB API dengan 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

Kode berikut menunjukkan cara untuk memanfaatkan API transaksi Amazon DocumentDB dengan 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

Kode berikut ini menunjukkan cara untuk memanfaatkan API transaksi Amazon DocumentDB dengan 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

Kode berikut ini menunjukkan cara untuk memanfaatkan API transaksi Amazon DocumentDB dengan 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

Kode berikut ini menunjukkan cara untuk memanfaatkan API transaksi Amazon DocumentDB dengan 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

Kode berikut ini menunjukkan cara untuk memanfaatkan API transaksi Amazon DocumentDB dengan 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
```

Perintah yang Didukung

Perintah	Didukung
`abortTransaction`	Ya
`commitTransaction`	Ya
`endSessions`	Ya
`killSession`	Ya
`killAllSession`	Ya
`killAllSessionsByPattern`	Tidak
`refreshSessions`	Tidak
`startSession`	Ya

Kemampuan yang tidak didukung

Metode	Tahapan atau Perintah

`db.collection.aggregate()`	`$collStats`
	`$currentOp`
	`$indexStats`
	`$listSessions`
	`$out`

`db.collection.count()`
`db.collection.countDocuments()`
	`$where`
	`$near`
	`$nearSphere`

`db.collection.insert()`	`insert` tidak didukung jika tidak dijalankan terhadap koleksi yang ada. Metode ini didukung jika menargetkan koleksi yang sudah ada sebelumnya.

Sesi

Sesi MongoDB adalah kerangka kerja yang digunakan untuk mendukung operasi penulisan yang dapat dicoba lagi, konsistensi kausal, transaksi, dan pengelolaan di seluruh basis data. Ketika sesi dibuat, pengidentifikasi sesi logis (lsid) dihasilkan oleh klien dan digunakan untuk menandai semua operasi di dalam sesi tersebut ketika mengirim perintah ke server.

Amazon DocumentDB mendukung penggunaan sesi untuk mengaktifkan transaksi, tetapi tidak mendukung konsistensi kausal atau penulisan yang dapat dicoba lagi.

Ketika memanfaatkan transaksi di dalam Amazon DocumentDB, transaksi akan dimulai dari dalam sesi menggunakan `session.startTransaction()` API dan sesi mendukung transaksi tunggal pada satu waktu. Serupa dengan itu, transaksi diselesaikan menggunakan penyimpanan (`session.commitTransaction()`) maupun pembatalan (`session.abortTransaction()`) API.

Konsistensi kausal

Konsistensi kausal menjamin bahwa dalam satu sesi klien, klien akan mengamati konsistensi baca-setelah-tulis, monatomik reads/writes, dan penulisan akan mengikuti pembacaan dan jaminan ini berlaku di semua instance dalam sebuah cluster, bukan hanya yang utama. Amazon DocumentDB tidak mendukung konsistensi kausal dan pernyataan berikut ini akan mengakibatkan kesalahan.

```
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()
```

Anda dapat menonaktifkan konsistensi kausal di dalam sesi. Perhatikan bahwa melakukan hal itu akan memungkinkan Anda untuk menggunakan kerangka sesi, tetapi tidak akan memberikan jaminan konsistensi kausal untuk pembacaan. Ketika menggunakan Amazon DocumentDB, pembacaan dari primer akan berupa konsisten baca-setelah-tulis dan pembacaan dari instans replika pada akhirnya akan konsisten. Transaksi adalah kasus penggunaan primer untuk memanfaatkan sesi.

```
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 }
```

Penulisan yang dapat dicoba lagi

Penulisan yang dapat dicoba lagi adalah kemampuan di mana klien akan berupaya untuk mencoba lagi menulis operasi, satu kali, ketika kesalahan jaringan terjadi atau jika klien tidak dapat menemukan primer. Di Amazon DocumentDB, penulisan yang dapat dicoba lagi tidak didukung dan harus dinonaktifkan. Anda dapat menonaktifkannya dengan perintah (`retryWrites=false`) di string koneksi.

> **Note**
>
> Jika Anda menggunakan shell mongo lama (bukan mongosh), jangan sertakan `retryWrites=false` perintah dalam string kode apa pun. Secara default, penulisan yang dapat dicoba ulang dinonaktifkan. Termasuk `retryWrites=false` dapat menyebabkan kegagalan dalam perintah baca normal.

Kesalahan transaksi

Saat menggunakan transaksi, ada skenario yang dapat menghasilkan kesalahan yang menyatakan bahwa nomor transaksi tidak cocok dengan transaksi yang sedang berlangsung.

Kesalahan dapat dihasilkan dalam setidaknya dua skenario yang berbeda:

- Setelah batas waktu transaksi satu menit.
- Setelah instance restart (karena patching, crash recovery, dll.), Dimungkinkan untuk menerima kesalahan ini bahkan dalam kasus di mana transaksi berhasil dilakukan. Selama instance restart, database tidak dapat membedakan antara transaksi yang berhasil diselesaikan versus transaksi yang dibatalkan. Dengan kata lain, status penyelesaian transaksi bersifat ambigu.

Cara terbaik untuk menangani kesalahan ini adalah dengan membuat pembaruan transaksional menjadi idempoten -- misalnya, dengan menggunakan `$set` mutator alih-alih operasi. increment/ decrement Lihat di bawah ini:

```
{ "ok" : 0,
"operationTime" : Timestamp(1603938167, 1),
"code" : 251,
"errmsg" : "Given transaction number 1 does not match any in-progress transactions."
}
```

Praktik terbaik untuk Amazon DocumentDB

Pelajari praktik terbaik untuk bekerja dengan Amazon DocumentDB (dengan kompatibilitas MongoDB). Bagian ini terus diperbarui saat praktik terbaik baru diidentifikasi.

Topik

- Pedoman operasional dasar
- Ukuran instans
- Bekerja dengan indeks
- Praktik terbaik keamanan
- Optimalisasi biaya
- Menggunakan metrik untuk mengidentifikasi masalah performa
- TTL dan beban kerja time series
- Migrasi
- Bekerja dengan kelompok parameter cluster
- Kueri pipa agregasi
- BatchInsert dan BatchUpdate

Pedoman operasional dasar

Berikut ini adalah panduan operasional dasar yang harus diikuti setiap orang saat bekerja dengan Amazon DocumentDB. Perjanjian Tingkat Layanan Amazon DocumentDB mewajibkan Anda untuk mengikuti panduan berikut ini.

- Menerapkan klaster yang terdiri dari dua atau lebih instans AWS Amazon DocumentDB di dua Availability Zone. Untuk beban kerja produksi, kami merekomendasikan men-deploy klaster yang terdiri dari tiga atau lebih instans Amazon DocumentDB di tiga Availability Zone.
- Gunakan layanan dalam kuota layanan yang dinyatakan. Untuk informasi selengkapnya, lihat Kuota dan batas Amazon DocumentDB.
- Memantau memori, CPU, koneksi, dan penggunaan penyimpanan Anda. Untuk membantu Anda mempertahankan performa dan ketersediaan sistem, siapkan Amazon CloudWatch untuk memberi tahu Anda saat pola penggunaan berubah atau saat Anda mendekati kapasitas penerapan.

- Menaikkan skala instans Anda saat Anda mendekati batas kapasitas. Instans Anda harus disediakan dengan sumber daya komputasi yang cukup (yaitu, RAM, CPU) untuk mengakomodasi peningkatan permintaan yang tidak terduga dari aplikasi Anda.
- Atur periode retensi cadangan agar selaras dengan tujuan titik pemulihan Anda.
- Tes failover untuk klaster Anda untuk memahami berapa lama proses untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat Failover Amazon DocumentDB.
- Hubungkan ke klaster Amazon DocumentDB Anda dengan titik akhir klaster (lihat Titik akhir Amazon DocumentDB) dan dalam mode set replika (lihat Menghubungkan ke Amazon DocumentDB sebagai set replika) untuk meminimalkan dampak failover pada aplikasi Anda.
- Pilih pengaturan preferensi baca driver yang memaksimalkan penskalaan baca sekaligus memenuhi persyaratan konsistensi baca aplikasi Anda. Preferensi baca `secondaryPreferred` mengaktifkan pembacaan replika dan membebaskan instans primer untuk melakukan lebih banyak pekerjaan. Untuk informasi selengkapnya, lihat Baca opsi preferensi.
- Rancang aplikasi Anda agar menjadi tangguh jika terjadi peristiwa kesalahan jaringan dan basis data. Gunakan mekanisme kesalahan driver Anda untuk membedakan antara kesalahan sementara dan kesalahan terus-menerus. Coba lagi kesalahan sementara menggunakan mekanisme backoff eksponensial bila sesuai. Pastikan bahwa aplikasi Anda mempertimbangkan konsistensi data ketika menerapkan logika coba lagi.
- Aktifkan perlindungan penghapusan klaster untuk semua klaster produksi, atau klaster mana pun yang memiliki data berharga. Sebelum menghapus klaster Amazon DocumentDB, ambil snapshot akhir. Jika Anda menggunakan sumber daya dengan CloudFormation, aktifkan perlindungan penghentian. Untuk informasi selengkapnya, lihat Perlindungan penghentian dan perlindungan penghapusan.
- Saat membuat cluster Amazon DocumentDB, `--engine-version` ini adalah parameter opsional yang default ke versi mesin utama terbaru. Versi mesin default saat ini adalah 5.0.0 (catatan: Amazon DocumentDB 8.0 tersedia tetapi harus ditentukan secara eksplisit). Ketika versi mesin utama baru dirilis, versi mesin default untuk `--engine-version` akan diperbarui untuk mencerminkan versi mesin utama terakhir. Akibatnya, untuk beban kerja produksi, dan terutama yang bergantung pada skrip, otomatisasi, atau CloudFormation templat, kami menyarankan Anda secara eksplisit menentukan `--engine-version` ke versi utama yang dimaksud.

Ukuran instans

Salah satu aspek terpenting dalam memilih ukuran instans di Amazon DocumentDB adalah jumlah RAM untuk cache Anda. Amazon DocumentDB mencadangkan sepertiga RAM untuk layanannya

sendiri, artinya hanya dua pertiga dari RAM instans yang tersedia untuk cache. Dengan demikian, itu adalah praktik terbaik Amazon DocumentDB untuk memilih tipe instans dengan RAM yang cukup agar sesuai dengan set kerja Anda (yaitu, data dan indeks) dalam memori. Memiliki instance berukuran tepat akan membantu mengoptimalkan kinerja secara keseluruhan dan berpotensi meminimalkan I/O biaya. Anda dapat menggunakan kalkulator ukuran Amazon DocumentDB pihak ketiga untuk memperkirakan ukuran instans untuk beban kerja tertentu.

Untuk menentukan apakah set kerja aplikasi Anda cocok dengan memori, pantau `BufferCacheHitRatio` penggunaan Amazon CloudWatch untuk setiap instance dalam klaster yang sedang dimuat.

`BufferCacheHitRatio` CloudWatch Metrik mengukur persentase data dan indeks yang disajikan dari cache memori instance (versus volume penyimpanan). Secara umum, nilai `BufferCacheHitRatio` harus setinggi mungkin, karena membaca data dari memori set kerja lebih cepat dan lebih hemat biaya daripada membaca dari volume penyimpanan. Selagi itu diinginkan untuk menjaga `BufferCacheHitRatio` sedekat mungkin dengan 100%, nilai terbaik yang dapat dicapai akan tergantung pada pola akses dan persyaratan performa aplikasi Anda. Untuk mempertahankan `BufferCacheHitRatio` setinggi mungkin, dianjurkan agar instans di klaster Anda disediakan dengan RAM yang cukup untuk dapat menyesuaikan indeks dan set data kerja dalam memori.

Jika indeks Anda tidak sesuai dengan memori, Anda akan melihat `BufferCacheHitRatio` yang lebih rendah. Terus membaca dari disk menimbulkan I/O biaya tambahan dan tidak lebih baik daripada membaca dari memori. Jika rasio `BufferCacheHitRatio` Anda lebih rendah dari yang diharapkan, tingkatkan ukuran instans untuk klaster Anda guna menyediakan lebih banyak RAM untuk menyesuaikan data set kerja dalam memori. Jika menskalakan ke atas kelas instans menghasilkan peningkatan dramatis dalam `BufferCacheHitRatio`, maka set kerja aplikasi anda tidak sesuai di memori. Lanjutkan untuk menaikkan skala sampai `BufferCacheHitRatio` tidak lagi meningkat secara dramatis setelah operasi penskalaan. Untuk informasi tentang pemantauan metrik instans, lihat Metrik Amazon DocumentDB.

Tergantung pada beban kerja dan persyaratan latensi Anda, itu mungkin dapat diterima untuk aplikasi Anda untuk memiliki nilai `BufferCacheHitRatio` yang lebih tinggi selama penggunaan kondisi stabil, tetapi memiliki penurunan `BufferCacheHitRatio` secara berkala arena kueri analitik yang perlu memindai seluruh koleksi dijalankan pada instans. Penurunan periodik dalam `BufferCacheHitRatio` dapat bermanifestasi sebagai latensi yang lebih tinggi untuk kueri berikutnya yang perlu untuk mengisi ulang data set kerja dari volume penyimpanan kembali ke cache buffer. Kami menyarankan Anda menguji beban kerja Anda di lingkungan pra-produksi dengan

beban kerja produksi yang representatif terlebih dahulu untuk memahami karakteristik kinerja dan **`BufferCacheHitRatio`** sebelum menerapkan beban kerja ke produksi.

`BufferCacheHitRatio` adalah metrik khusus instans, jadi instans yang berbeda dalam klaster yang sama mungkin memiliki nilai-nilai `BufferCacheHitRatio` yang berbeda tergantung pada cara pembacaan didistribusikan di antara instans primer dan replika. Jika beban kerja operasional Anda tidak dapat menangani peningkatan periodik latensi dari mengisi ulang cache set kerja setelah menjalankan kueri analitik, Anda harus mencoba untuk mengisolasi cache buffer beban kerja reguler dari kueri analitik. Anda dapat mencapai isolasi `BufferCacheHitRatio` lengkap dengan mengarahkan kueri operasional ke instans primer dan kueri analitik hanya ke instans replika. Anda juga dapat mencapai isolasi parsial dengan mengarahkan kueri analitik ke instans replika tertentu dengan pemahaman bahwa beberapa persentase kueri reguler juga akan berjalan pada replika tersebut dan berpotensi terpengaruh.

Nilai `BufferCacheHitRatio` yang sesuai tergantung pada kasus penggunaan dan persyaratan aplikasi Anda. Tidak ada satu nilai terbaik atau minimum untuk metrik ini; hanya Anda yang dapat memutuskan apakah tradeoff dari `BufferCacheHitRatio` yang sementara lebih rendah dapat diterima dari perspektif biaya dan performa.

Bekerja dengan indeks

Membangun Indeks

Ketika mengimpor data ke Amazon DocumentDB, Anda harus membuat indeks Anda sebelum mengimpor set data besar. Anda dapat menggunakan Alat Indeks Amazon DocumentDB untuk mengekstrak indeks dari instans MongoDB atau direktori `mongodump` yang sedang berjalan, dan membuat indeks tersebut di klaster Amazon DocumentDB. Untuk panduan lebih lanjut tentang migrasi, lihat Migrasi ke Amazon DocumentDB.

Selektivitas indeks

Kami merekomendasikan Anda membatasi pembuatan indeks ke bidang di mana jumlah nilai duplikat kurang dari 1% dari jumlah total dokumen dalam koleksi. Sebagai contoh, jika pengumpulan Anda berisi 100.000 dokumen, buat indeks hanya pada bidang tempat nilai yang sama muncul 1000 kali atau kurang.

Memilih indeks dengan jumlah nilai unik yang tinggi (yaitu, kardinalitas tinggi) memastikan bahwa operasi filter mengembalikan sejumlah kecil dokumen, sehingga menghasilkan performa yang baik

selama pemindaian indeks. Contoh indeks kardinalitas tinggi adalah indeks unik, yang menjamin bahwa predikat kesetaraan mengembalikan paling banyak satu dokumen. Contoh kardinalitas rendah termasuk indeks di atas bidang Boolean dan indeks di atas hari dalam seminggu. Karena performanya yang buruk, indeks kardinalitas rendah tidak mungkin dipilih oleh pengoptimal kueri basis data. Pada saat yang sama, indeks kardinalitas rendah terus mengkonsumsi sumber daya seperti ruang disk dan. I/Os Sebagai aturan praktis, Anda harus menargetkan indeks pada bidang di mana frekuensi nilai khas adalah 1% dari total ukuran koleksi atau kurang.

Selain itu, direkomendasikan untuk hanya membuat indeks pada bidang yang umumnya digunakan sebagai filter dan secara teratur mencari indeks yang tidak terpakai. Untuk informasi selengkapnya, lihat Bagaimana cara menganalisis penggunaan indeks dan mengidentifikasi indeks yang tidak digunakan?.

Dampak indeks pada penulisan data

Meskipun indeks dapat meningkatkan performa kueri dengan menghindari kebutuhan untuk memindai setiap dokumen dalam koleksi, peningkatan ini disertai dengan tradeoff. Untuk setiap indeks pada koleksi, setiap kali dokumen dimasukkan, diperbarui, atau dihapus, basis data harus memperbarui koleksi dan menulis bidang untuk masing-masing indeks untuk koleksi. Sebagai contoh, jika koleksi memiliki sembilan indeks, basis data harus melakukan sepuluh penulisan sebelum mengakui operasi ke klien. Dengan demikian, setiap indeks tambahan menimbulkan latensi tulis tambahan, I/O's, dan peningkatan penyimpanan yang digunakan secara keseluruhan.

Instans klaster harus berukuran tepat untuk menjaga semua memori set kerja. Ini menghindari kebutuhan untuk terus membaca halaman indeks dari volume penyimpanan, yang berdampak negatif pada kinerja dan menghasilkan I/O biaya yang lebih tinggi. Untuk informasi selengkapnya, lihat Ukuran instans.

Untuk performa terbaik, meminimalkan jumlah indeks dalam koleksi Anda, tambahkan hanya indeks yang diperlukan untuk meningkatkan performa untuk kueri umum. Meskipun beban kerja bervariasi, pedoman yang baik adalah untuk menjaga jumlah indeks per koleksi menjadi lima atau lebih sedikit.

Mengidentifikasi indeks yang hilang

Mengidentifikasi indeks yang hilang adalah praktik terbaik yang kami rekomendasikan untuk dilakukan secara teratur. Untuk informasi selengkapnya, lihat Bagaimana cara mengidentifikasi indeks yang hilang?.

Mengidentifikasi indeks yang tidak digunakan

Mengidentifikasi dan menghapus indeks yang tidak terpakai adalah praktik terbaik yang kami rekomendasikan untuk dilakukan secara teratur. Untuk informasi selengkapnya, lihat Bagaimana cara menganalisis penggunaan indeks dan mengidentifikasi indeks yang tidak digunakan?.

Praktik terbaik keamanan

Untuk praktik terbaik keamanan, Anda harus menggunakan akun AWS Identity and Access Management (IAM) untuk mengontrol akses ke operasi Amazon DocumentDB API, terutama operasi yang membuat, memodifikasi, atau menghapus sumber daya Amazon DocumentDB. Sumber daya tersebut termasuk klaster, grup keamanan, dan grup parameter. Anda juga harus menggunakan IAM untuk mengontrol tindakan yang melakukan tindakan administrasi umum seperti mencadangkan pemulihan klaster. Saat membuat IAM role, terapkan prinsip hak istimewa terendah.

- Terapkan hak istimewa terendah dengan kontrol akses berbasis peran.
- Tetapkan akun IAM individu untuk setiap orang yang mengelola sumber daya Amazon DocumentDB. Jangan gunakan pengguna Akun AWS root untuk mengelola sumber daya Amazon DocumentDB. Buat pengguna IAM untuk semua orang, termasuk Anda sendiri.
- Berikan setiap pengguna IAM set izin minimum yang diperlukan untuk melakukan tugas mereka.
- Gunakan grup IAM untuk mengelola izin secara efektif bagi beberapa pengguna. Untuk informasi lebih lanjut tentang IAM, lihat Panduan Pengguna IAM. Untuk informasi tentang praktik terbaik IAM, lihat Praktik Terbaik IAM.
- Secara rutin putar kredensial IAM Anda.
- Konfigurasikan AWS Secrets Manager untuk secara otomatis memutar rahasia untuk Amazon DocumentDB. Untuk informasi selengkapnya, lihat Memutar AWS Rahasia Secrets Manager Anda dan Rotating Secrets untuk Amazon DocumentDB di Panduan Pengguna AWS Secrets Manager.
- Berikan setiap pengguna Amazon DocumentDB set izin minimum yang diperlukan untuk melakukan tugas mereka. Untuk informasi selengkapnya, lihat Akses database menggunakan Role-Based Access Control.
- Gunakan Transport Layer Security (TLS) untuk mengenkripsi data Anda dalam perjalanan dan AWS KMS mengenkripsi data Anda saat istirahat.

Optimalisasi biaya

Praktik terbaik berikut dapat membantu Anda mengelola dan meminimalkan biaya Anda ketika menggunakan Amazon DocumentDB. Untuk informasi harga, lihat Harga Amazon DocumentDB (dengan kompatibilitas MongoDB) dan FAQ Amazon DocumentDB (dengan kompatibilitas MongoDB).

- Buat pemberitahuan penagihan pada ambang batas 50 persen dan 75 persen dari tagihan yang Anda harapkan untuk bulan tersebut. Untuk informasi selengkapnya tentang membuat pemberitahuan penagihan, lihat Membuat Alarm Penagihan.
- Arsitektur Amazon DocumentDB memisahkan penyimpanan dan komputasi, sehingga klaster instans tunggal pun sangat berdaya tahan. Volume penyimpanan klaster mereplikasi data enam cara di tiga Availability Zone, menyediakan daya tahan yang sangat tinggi terlepas dari jumlah instans dalam klaster. Klaster produksi khas memiliki tiga atau lebih instans untuk menyediakan ketersediaan tinggi. Namun, Anda dapat mengoptimalkan biaya dengan menggunakan klaster pengembangan instans tunggal ketika ketersediaan tinggi tidak diperlukan.
- Untuk skenario pengembangan dan pengujian, hentikan klaster ketika tidak lagi diperlukan dan mulai klaster ketika pengembangan dilanjutkan. Untuk informasi selengkapnya, lihat Menghentikan dan memulai cluster Amazon DocumentDB.
- Baik TTL dan aliran perubahan terjadi I/O saat data ditulis, dibaca, dan dihapus. Jika Anda telah mengaktifkan fitur ini tetapi tidak memanfaatkannya dalam aplikasi Anda, menonaktifkan fitur tersebut dapat membantu mengurangi biaya.

Menggunakan metrik untuk mengidentifikasi masalah performa

Topik

- Melihat metrik performa
- Mengatur CloudWatch alarm
- Mengevaluasi metrik performa
- Mengevaluasi penggunaan instans Amazon DocumentDB dengan metrik CloudWatch
- Menyetel kueri

Untuk mengidentifikasi masalah performa yang disebabkan sumber daya yang tidak mencukupi dan kemacetan umum lainnya, Anda dapat memantau metrik yang tersedia untuk klaster Amazon DocumentDB.

Melihat metrik performa

Pantau metrik kinerja secara rutin untuk melihat nilai rata-rata, maksimum, dan minimum untuk berbagai rentang waktu. Hal ini membantu Anda mengidentifikasi saat performa menurun. Anda juga dapat mengatur CloudWatch alarm Amazon untuk ambang metrik tertentu sehingga Anda diberi tahu jika mereka tercapai.

Untuk memecahkan masalah kinerja, penting untuk memahami kinerja dasar sistem. Setelah Anda menyiapkan klaster baru dan menjalankannya dengan beban kerja tipikal, tangkap nilai rata-rata, maksimum, dan minimum dari semua metrik kinerja pada interval yang berbeda (misalnya, 1 jam, 24 jam, 1 minggu, 2 minggu). Ini memberi Anda gambaran tentang apa yang normal. Sehingga membantu mendapatkan perbandingan untuk jam sibuk dan tidak sibuk. Kemudian, Anda dapat menggunakan informasi ini untuk mengidentifikasi saat performa turun di bawah tingkat standar.

Anda dapat melihat metrik kinerja menggunakan Konsol Manajemen AWS atau AWS CLI. Untuk informasi selengkapnya, lihat Melihat CloudWatch data.

Mengatur CloudWatch alarm

Untuk menyetel CloudWatch alarm, lihat Menggunakan CloudWatch Alarm Amazon di Panduan CloudWatch Pengguna Amazon.

Mengevaluasi metrik performa

Instans memiliki beberapa kategori metrik yang berbeda. Cara Anda menentukan nilai yang dapat diterima tergantung pada metrik.

CPU

- Pemanfaatan CPU — Persentase kapasitas pemrosesan komputer yang digunakan.

Memori

- Memori yang Bisa Dibebaskan — Berapa banyak RAM yang tersedia pada instans.
- Penggunaan Swap — Berapa banyak ruang tukar yang digunakan oleh instans, dalam megabyte.

Input/output operasi

- IOPS Baca, IOPS Tulis — Jumlah rata-rata operasi baca atau tulis disk per detik.
- Latensi Baca, Latensi Tulis — Waktu rata-rata untuk operasi baca atau tulis dalam milidetik.
- Throughput Baca, Throughput Tulis — Rata-rata jumlah megabyte yang dibaca dari atau ditulis ke disk per detik.
- Kedalaman Antrian Disk — Jumlah I/O operasi yang menunggu untuk ditulis atau dibaca dari disk.

Lalu lintas jaringan

- Throughput Penerimaan Jaringan, Throughput Pengiriman Jaringan — Tingkat lalu lintas jaringan ke dan dari instans dalam megabyte per detik.

Koneksi basis data

- Koneksi DB — Jumlah sesi klien yang terhubung ke instans.

Secara umum, nilai yang dapat diterima untuk metrik performa bergantung pada seperti apa garis dasar Anda dan apa yang dilakukan aplikasi Anda. Periksa varian yang konsisten atau sedang tren dari garis dasar Anda.

Berikut ini adalah rekomendasi dan saran tentang jenis metrik tertentu:

- Konsumsi CPU tinggi — Nilai tinggi untuk konsumsi CPU mungkin sesuai, jika mereka menjaga tujuan untuk aplikasi Anda (seperti throughput atau konkurensi) dan diharapkan. Jika konsumsi CPU Anda secara konsisten lebih dari 80 persen, pertimbangkan untuk menskalakan ke atas instans Anda.
- Konsumsi RAM tinggi — Jika `FreeableMemory` metrik Anda sering turun di bawah 10% dari total memori instans, pertimbangkan untuk meningkatkan instans Anda. Untuk informasi selengkapnya tentang apa yang terjadi ketika instans DocumentDB Anda mengalami tekanan memori tinggi, lihat Tata Kelola Sumber Daya Amazon DocumentDB.
- Penggunaan swap — Metrik ini harus tetap pada atau mendekati nol. Jika penggunaan swap Anda signifikan, pertimbangkan untuk menskalakan ke atas instans Anda.
- Lalu lintas jaringan — Untuk lalu lintas jaringan, bicarakan dengan administrator sistem Anda untuk memahami throughput yang diharapkan untuk jaringan domain dan koneksi internet Anda. Selidiki lalu lintas jaringan jika throughput selalu di bawah yang diharapkan.

- Koneksi basis data — Pertimbangkan untuk membatasi koneksi basis data jika Anda melihat jumlah koneksi pengguna yang tinggi bersamaan dengan penurunan performa instans dan waktu respons. Jumlah koneksi pengguna terbaik untuk instans Anda bervariasi berdasarkan kelas instans Anda dan kerumitan operasi yang sedang dilakukan. Untuk masalah dengan metrik kinerja apa pun, salah satu hal pertama yang dapat Anda lakukan untuk meningkatkan perfroma adalah menyesuaikan kueri yang paling banyak digunakan dan paling mahal untuk melihat apakah hal tersebut menurunkan tekanan pada sumber daya sistem.

Jika kueri Anda disetel dan masalah tetap ada, pertimbangkan untuk memutakhirkan kelas instans Amazon DocumentDB Anda ke kelas dengan lebih banyak sumber daya (CPU, RAM, ruang disk, bandwidth jaringan, I/O kapasitas) yang terkait dengan masalah yang Anda alami.

Mengevaluasi penggunaan instans Amazon DocumentDB dengan metrik CloudWatch

Anda dapat menggunakan CloudWatch metrik untuk melihat throughput instans dan mengetahui apakah kelas instans menyediakan sumber daya yang memadai untuk aplikasi Anda. Untuk informasi tentang batas kelas instans Anda, lihat Batas instans dan temukan spesifikasi untuk kelas instans Anda untuk menemukan kinerja jaringan Anda.

Jika penggunaan instance Anda mendekati batas kelas instance, maka kinerja mungkin mulai melambat. CloudWatch Metrik dapat mengonfirmasi situasi ini sehingga Anda dapat merencanakan untuk meningkatkan skala secara manual ke kelas instance yang lebih besar.

Gabungkan nilai CloudWatch metrik berikut untuk mengetahui apakah Anda mendekati batas kelas instance:

- NetworkThroughputJumlah throughput jaringan yang diterima dan ditransmisikan oleh klien untuk setiap instans di cluster Amazon DocumentDB. Nilai throughput ini tidak termasuk lalu lintas jaringan antara instance di cluster dan volume penyimpanan cluster.
- StorageNetworkThroughputJumlah throughput jaringan yang diterima dan dikirim ke volume penyimpanan cluster Amazon DocumentDB oleh setiap instance di cluster Amazon DocumentDB.

Tambahkan NetworkThroughputke untuk menemukan throughput jaringan yang diterima dari dan dikirim ke volume penyimpanan klaster Amazon DocumentDB oleh setiap instance di cluster Amazon DocumentDB Anda. StorageNetworkThroughput Batas kelas instans untuk instans Anda harus lebih besar dari jumlah kedua metrik gabungan ini.

Anda dapat menggunakan metrik berikut untuk meninjau detail tambahan lalu lintas jaringan dari aplikasi klien Anda saat mengirim dan menerima:

- NetworkReceiveThroughputJumlah throughput jaringan yang diterima dari klien oleh setiap instans di cluster Amazon DocumentDB. Throughput ini tidak termasuk lalu lintas jaringan antara instance di cluster dan volume penyimpanan cluster.
- NetworkTransmitThroughputJumlah throughput jaringan yang dikirim ke klien oleh setiap instans di cluster Amazon DocumentDB. Throughput ini tidak termasuk lalu lintas jaringan antara instance di cluster dan volume penyimpanan cluster.
- StorageNetworkReceiveThroughputJumlah throughput jaringan yang diterima dari volume penyimpanan cluster Amazon DocumentDB oleh setiap instance di cluster.
- StorageNetworkTransmitThroughput—Jumlah throughput jaringan yang dikirim ke volume penyimpanan cluster Amazon DocumentDB oleh setiap instance di cluster.

Tambahkan semua metrik ini bersama-sama untuk mengevaluasi bagaimana penggunaan jaringan Anda dibandingkan dengan batas kelas instans. Batas kelas instans harus lebih besar dari jumlah metrik gabungan ini.

Batas jaringan dan pemanfaatan CPU oleh sebuah instance saling menguntungkan. Ketika throughput jaringan meningkat, maka pemanfaatan CPU juga meningkat. Pemantauan penggunaan CPU dan jaringan memberikan informasi tentang bagaimana dan mengapa sumber daya habis.

Untuk membantu meminimalkan penggunaan jaringan, Anda dapat mempertimbangkan:

- Menggunakan kelas instans yang lebih besar.
- Membagi permintaan tulis dalam batch untuk mengurangi keseluruhan transaksi.
- Mengarahkan beban kerja hanya-baca ke instans hanya-baca.
- Menghapus indeks yang tidak digunakan.

Menyetel kueri

Salah satu cara terbaik untuk meningkatkan kinerja klaster adalah dengan menyetel kueri yang paling sering digunakan dan paling intensif sumber daya untuk membuatnya lebih murah untuk dijalankan.

Anda dapat menggunakan profiler (lihat Membuat profil operasi Amazon DocumentDB) untuk mencatat waktu eksekusi dan detail operasi yang dilakukan di klaster Anda. Profiler berguna untuk

memantau operasi paling lambat di klaster Anda untuk membantu Anda meningkatkan performa kueri individual dan performa klaster secara keseluruhan.

Anda juga dapat menggunakan perintah `explain` untuk mempelajari cara menganalisis rencana kueri untuk kueri tertentu. Gunakan informasi ini untuk memodifikasi kueri atau koleksi yang mendasari untuk meningkatkan kinerja kueri Anda (misalnya, menambahkan indeks).

TTL dan beban kerja time series

Penghapusan dokumen yang dihasilkan dari kedaluwarsanya indeks TTL merupakan proses upaya terbaik. Dokumen tidak dijamin akan dihapus dalam periode tertentu. Faktor-faktor seperti ukuran instans, pemanfaatan sumber daya instans, ukuran dokumen, throughput keseluruhan, jumlah indeks, dan apakah indeks dan set kerja sesuai dalam memori, semuanya dapat memengaruhi waktu ketika dokumen yang kedaluwarsa dihapus oleh proses TTL.

Ketika monitor TTL menghapus dokumen Anda, setiap penghapusan menimbulkan I/O biaya, yang meningkatkan tagihan Anda. Jika tingkat throughput dan penghapusan TTL meningkat, Anda harus mengharapkan tagihan yang lebih tinggi karena peningkatan I/O penggunaan. Namun, jika Anda tidak membuat indeks TTL untuk menghapus dokumen, melainkan mengelompokkan dokumen ke dalam koleksi berdasarkan waktu, dan cukup membuang koleksi tersebut saat tidak lagi diperlukan, Anda tidak akan dikenakan biaya IO apa pun. Ini dapat secara signifikan menghemat biaya daripada menggunakan indeks TTL.

Untuk beban kerja deret waktu, Anda dapat mempertimbangkan untuk membuat koleksi bergulir alih-alih indeks TTL karena koleksi bergulir dapat menjadi cara yang lebih baik untuk menghapus data dan tidak terlalu intensif. I/O Jika Anda memiliki koleksi besar (terutama koleksi di atas 1TB) atau I/O biaya penghapusan TTL menjadi perhatian, kami sarankan Anda mempartisi dokumen ke dalam koleksi berdasarkan waktu, dan menjatuhkan koleksi ketika dokumen tidak lagi diperlukan. Anda dapat membuat satu koleksi per hari atau satu per minggu, tergantung pada tingkat menyerap data Anda. Meskipun persyaratan akan bervariasi tergantung pada aplikasi Anda, aturan praktis yang baik adalah memiliki lebih banyak koleksi yang lebih kecil daripada beberapa koleksi besar. Menjatuhkan koleksi ini tidak menimbulkan I/O biaya, dan bisa lebih cepat dan lebih hemat biaya daripada menggunakan indeks TTL.

Migrasi

Sebagai praktik terbaik, kami merekomendasikan bahwa ketika memigrasikan data ke Amazon DocumentDB, Anda terlebih dahulu membuat indeks Anda di Amazon DocumentDB sebelum

memigrasikan data. Membuat indeks terlebih dahulu dapat mengurangi waktu keseluruhan dan meningkatkan kecepatan migrasi. Untuk melakukannya, Anda dapat menggunakan Alat Indeks Amazon DocumentDB. Untuk informasi lebih lanjut tentang migrasi, lihat Panduan migrasi Amazon DocumentDB.

Kami juga merekomendasikan bahwa sebelum Anda memigrasikan basis data produksi Anda, praktik terbaik adalah untuk sepenuhnya menguji aplikasi Anda di Amazon DocumentDB, dengan mempertimbangkan fungsionalitas, kinerja, operasi, dan biaya.

Bekerja dengan kelompok parameter cluster

Kami menyarankan agar Anda mencoba perubahan grup parameter klaster pada klaster uji sebelum menerapkan perubahan ke klaster produksi Anda. Untuk informasi tentang pencadangan kluster Anda, lihat Mencadangkan dan memulihkan di Amazon DocumentDB.

Kueri pipa agregasi

Ketika membuat kueri alur agregasi dengan beberapa tahap dan mengevaluasi hanya sebagian dari data dalam kueri, gunakan tahap `$match` sebagai tahap pertama atau di awal alur. Menggunakan `$match` terlebih dahulu akan mengurangi jumlah dokumen yang perlu diproses pada tahap selanjutnya dalam kueri alur agregasi, sehingga meningkatkan kinerja kueri Anda.

`BatchInsert` dan `BatchUpdate`

Saat melakukan tingkat `batchInsert` and/or `batchUpdate` operasi bersamaan yang tinggi, dan jumlah `FreeableMemory` (CloudWatch Metrik) menjadi nol pada instans utama Anda, Anda dapat mengurangi konkurensi penyisipan batch atau memperbarui beban kerja atau, jika konkurensi beban kerja tidak dapat dikurangi, tingkatkan ukuran instans untuk menambah jumlah. `FreeableMemory`

Perbedaan fungsional: Amazon DocumentDB dan MongoDB

Berikut ini adalah perbedaan fungsional antara Amazon DocumentDB (dengan kompatibilitas MongoDB) dan MongoDB.

Topik

- Manfaat fungsional Amazon DocumentDB
- Perbedaan fungsional yang diperbarui
- Perbedaan fungsional dengan MongoDB

Manfaat fungsional Amazon DocumentDB

Transaksi implisit

Di Amazon DocumentDB, semua pernyataan CRUD (`findAndModify`, `update`, `insert`, `delete`) menjamin atomisitas dan konsistensi, bahkan untuk operasi yang memodifikasi beberapa dokumen. Dengan peluncuran Amazon DocumentDB 4.0, transaksi eksplisit yang menyediakan properti ACID untuk operasi multi-pernyataan dan multi-koleksi kini didukung. Untuk informasi lebih lanjut tentang menggunakan transaksi di Amazon DocumentDB, lihat. Transaksi di Amazon DocumentDB

Berikut ini adalah contoh operasi di Amazon DocumentDB yang memodifikasi beberapa dokumen yang memenuhi perilaku atomik dan konsisten.

```
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 }]
})
```

Operasi individu yang menyusun operasi massal seperti `updateMany` dan `deleteMany` bersifat atomik tetapi keseluruhan operasi massal tidak bersifat atomik. Misalnya, keseluruhan operasi `insertMany` bersifat atomik jika operasi penyisipan individual berhasil dijalankan tanpa kesalahan. Jika kesalahan ditemukan pada operasi `insertMany`, setiap pernyataan penyisipan individu dalam operasi `insertMany` akan dijalankan sebagai operasi atom. Jika Anda memerlukan properti ACID untuk `insertMany``updateMany`,, dan `deleteMany` operasi, disarankan untuk menggunakan transaksi.

Perbedaan fungsional yang diperbarui

Amazon DocumentDB terus meningkatkan kompatibilitas pada MongoDB dengan merujuk kembali pada kemampuan yang perlu kami bangun sesuai permintaan pelanggan. Bagian ini berisi perbedaan fungsional yang telah kami hilangkan dari Amazon DocumentDB untuk mempermudah migrasi dan pembangunan aplikasi bagi pelanggan kami.

Topik

- Pengindeksan array

- Multi-key indeks
- Karakter nol dalam string
- Role-based kontrol akses
- $regex pengindeksan
- Proyeksi untuk dokumen bersarang

Pengindeksan array

Mulai 23 April 2020, Amazon DocumentDB kini mendukung kemampuan untuk mengindeks array yang lebih besar daripada 2.048 byte. Batas untuk setiap item individual dalam array masih tetap 2.048 byte, dan ini konsisten dengan MongoDB.

Jika Anda membuat indeks baru, tidak ada tindakan yang diperlukan untuk memanfaatkan fungsionalitas yang ditingkatkan. Jika Anda sudah memiliki indeks, Anda dapat memanfaatkan peningkatan fungsionalitas dengan menghapus indeks tersebut dan membuatnya lagi. Versi indeks saat ini dengan kemampuan yang ditingkatkan adalah `"v" : 3`.

> **Note**
>
> Untuk klaster produksi, penghapusan indeks mungkin berdampak pada performa aplikasi Anda. Kami menyarankan Anda untuk terlebih dahulu menguji dan melanjutkan dengan hati-hati ketika membuat perubahan pada sistem produksi. Selain itu, waktu yang dibutuhkan untuk menciptakan indeks akan menjadi fungsi ukuran data secara keseluruhan pada pengumpulan.

Anda dapat mengajikan kueri untuk versi indeks Anda dengan menggunakan perintah berikut.

```
db.collection.getIndexes()
```

Output dari operasi ini terlihat seperti berikut ini. Dalam output ini, versi indeks adalah `"v" : 3`, yang merupakan versi indeks terbaru.

```
[
    {
        "v" : 3,
        "key" : {
```

```
        "_id" : 1
        },
        "name" : "_id_",
        "ns" : "test.test"
    }
]
```

Multi-key indeks

Mulai 23 April 2020, Amazon DocumentDB sekarang mendukung kemampuan untuk membuat indeks gabungan dengan beberapa kunci dalam array yang sama.

Jika Anda membuat indeks baru, tidak ada tindakan yang diperlukan untuk memanfaatkan fungsionalitas yang ditingkatkan. Jika Anda sudah memiliki indeks, Anda dapat memanfaatkan peningkatan fungsionalitas dengan menghapus indeks tersebut dan membuatnya lagi. Versi indeks saat ini dengan kemampuan yang ditingkatkan adalah `"v" : 3`.

> **Note**
>
> Untuk klaster produksi, penghapusan indeks mungkin berdampak pada performa aplikasi Anda. Kami menyarankan Anda untuk terlebih dahulu menguji dan melanjutkan dengan hati-hati ketika membuat perubahan pada sistem produksi. Selain itu, waktu yang dibutuhkan untuk menciptakan indeks akan menjadi fungsi ukuran data secara keseluruhan pada pengumpulan.

Anda dapat mengajikan kueri untuk versi indeks Anda dengan menggunakan perintah berikut.

```
db.collection.getIndexes()
```

Output dari operasi ini terlihat seperti berikut ini. Dalam output ini, versi indeks adalah `"v" : 3`, yang merupakan versi indeks terbaru.

```
[
    {
        "v" : 3,
        "key" : {
            "_id" : 1
        },
        "name" : "_id_",
```

```
        "ns" : "test.test"
    }
]
```

Karakter nol dalam string

Mulai 22 Juni 2020, Amazon DocumentDB sekarang mendukung karakter null (`'\0'`) dalam string.

Role-based kontrol akses

Mulai 26 Maret 2020, Amazon DocumentDB mendukung kontrol akses berbasis peran (RBAC) untuk peran bawaan. Untuk mempelajari selengkapnya, lihat Role-Based Kontrol Akses.

`$regex` pengindeksan

Mulai 22 Juni 2020, Amazon DocumentDB kini mendukung kemampuan operator `$regex` untuk menggunakan indeks.

Untuk menggunakan indeks dengan operator `$regex`, Anda harus menggunakan perintah `hint()`. Saat menggunakan `hint()`, Anda harus menentukan nama bidang tempat Anda menerapkan `$regex`. Misalnya, jika Anda memiliki indeks pada bidang `product` dengan nama indeks sebagai `p_1`, `db.foo.find({product: /^x.*/}).hint({product:1})` akan menggunakan indeks `p_1`, tetapi `db.foo.find({product: /^x.*/}).hint("p_1")` tidak akan menggunakan indeks. Anda dapat memverifikasi apakah suatu indeks dipilih dengan menggunakan perintah `explain()` atau menggunakan profiler untuk mencatat kueri lambat. Misalnya, `db.foo.find({product: /^x.*/}).hint("p_1").explain()`.

> **Note**
>
> Metode `hint()` ini hanya dapat digunakan dengan satu indeks pada satu waktu.

Penggunaan indeks untuk kueri `$regex` dioptimalkan untuk kueri ekspresi reguler yang menggunakan prefiks dan tidak menentukan opsi ekspresi reguler `i`, `m`, atau `o`.

Saat menggunakan indeks dengan `$regex`, Anda disarankan untuk membuat indeks pada bidang yang sangat selektif yang memuat jumlah nilai duplikat kurang dari 1% dari jumlah total dokumen dalam koleksi. Sebagai contoh, jika pengumpulan Anda berisi 100.000 dokumen, buat indeks hanya pada bidang tempat nilai yang sama muncul 1000 kali atau kurang.

Proyeksi untuk dokumen bersarang

Pada operator `$project`, terdapat perbedaan fungsional antara Amazon DocumentDB dan MongoDB di versi 3.6 yang telah diselesaikan di Amazon DocumentDB 4.0 tetapi akan tetap tidak didukung di Amazon DocumentDB 3.6.

Amazon DocumentDB 3.6 hanya mempertimbangkan bidang pertama dalam dokumen bersarang saat menerapkan proyeksi, sedangkan MongoDB 3.6 akan mengurai subdokumen dan menerapkan proyeksi ke setiap sub dokumen juga.

Misalnya: jika proyeksinya adalah `“a.b.c”: 1`, maka perilaku tersebut berfungsi seperti yang diharapkan di Amazon DocumentDB dan MongoDB. Namun, jika proyeksinya adalah `{a:{b:{c:1}}}` maka Amazon DocumentDB 3.6 hanya akan menerapkan proyeksi ke `a` dan bukan `b` atau `c`. Di Amazon DocumentDB 4.0, proyeksi `{a:{b:{c:1}}}` akan diterapkan ke `a`, `b`, dan `c`.

Perbedaan fungsional dengan MongoDB

Topik

- $ VectorSearch operator
- OpCountersCommand
- Database dan koleksi admin
- CursorMaxTimems
- jelaskan()
- Indeks membangun
- Pencarian dengan kunci kosong di jalur
- MongoDB API, operasi, dan tipe data
- utilitas mongodump dan mongorestore
- Pemesanan hasil
- Penulisan yang dapat dicoba lagi
- Indeks jarang
- Menggunakan $elemMatch dalam ekspresi $all
- Dolar ($) dan titik (.) dalam nama bidang
- $lookup
- $ penyortiran alami dan terbalik

`$ VectorSearch` operator

Amazon DocumentDB tidak `$vectorSearch` mendukung sebagai operator independen. Sebaliknya kami mendukung, `vectorSearch` di dalam `$search` operator. Untuk informasi selengkapnya, lihat Pencarian vektor untuk Amazon DocumentDB.

`OpCountersCommand`

`OpCountersCommand`Perilaku Amazon DocumentDB menyimpang dari MongoDB sebagai berikut: `opcounters.command`

- MongoDB `opcounters.command` menghitung semua perintah kecuali menyisipkan, memperbarui, dan menghapus sementara Amazon DocumentDB juga `OpCountersCommand` mengecualikan perintah. `find`
- Amazon DocumentDB menghitung beberapa perintah internal ke arah. `OpCountersCommand`

Database dan koleksi admin

Amazon DocumentDB tidak mendukung basis data admin atau lokal atau koleksi MongoDB `system.*` atau `startup_log` secara berurutan.

`CursorMaxTimems`

Di Amazon DocumentDB, `cursor.maxTimeMS` menyetel ulang penghitung untuk setiap permintaan `getMore`. Jadi, jika `maxTimeMS` 3000MS ditentukan, kueri membutuhkan 2800MS, dan setiap permintaan `getMore` berikutnya membutuhkan 300MS, maka kursor tidak akan kehabisan waktu. Kursor hanya akan kehabisan waktu jika satu operasi, baik kueri atau permintaan individu `getMore`, membutuhkan lebih dari `maxTimeMS` yang ditentukan. Selanjutnya, penyapu yang memeriksa waktu eksekusi kursor berjalan pada granularitas lima (5) menit.

jelaskan()

Amazon DocumentDB mengemulasi API MongoDB 3.6, 4.0, 5.0, dan 8.0 pada mesin database yang dibuat khusus yang menggunakan sistem penyimpanan self-healing yang terdistribusi, toleran kesalahan, dan self-healing. Akibatnya, rencana kueri dan keluaran dari `explain()` mungkin berbeda antara Amazon DocumentDB dan MongoDB. Pelanggan yang ingin kontrol atas rencana kueri mereka dapat menggunakan operator `$hint` untuk memaksa pemilihan indeks yang diutamakan.

Indeks membangun

Amazon DocumentDB memungkinkan hanya satu build indeks terjadi pada koleksi pada waktu tertentu. Baik di latar depan atau latar belakang. Jika operasi seperti `createIndex()` atau `dropIndex()` terjadi pada koleksi yang sama saat pembuatan indeks sedang berlangsung, operasi yang baru dicoba akan gagal.

Secara default, build indeks di Amazon DocumentDB dan MongoDB versi 4.0 terjadi di latar depan. MongoDB versi 4.2, dan yang lebih baru mengabaikan opsi pembuatan indeks latar belakang jika ditentukan ke createIndexes atau pembantu shell-nya dan. `createIndex()` `createIndexes()`

Indeks Time to Live (TTL) mulai kedaluwarsa dokumen setelah pembuatan indeks selesai.

Pencarian dengan kunci kosong di jalur

Saat Anda melakukan pencarian dengan kunci yang menyertakan string kosong sebagai bagian dari jalur (mis. `x.`, `x..b`), dan objek memiliki jalur kunci string kosong (mis. `{"x" : [ { "" : 10 }, { "b" : 20 } ]}`) di dalam array, Amazon DocumentDB akan mengembalikan hasil yang berbeda dibandingkan jika Anda menjalankan pencarian yang sama di MongoDB.

Di MongoDB, pencarian jalur kunci kosong di dalam array berfungsi seperti yang diharapkan ketika kunci string kosong tidak berada di ujung pencarian jalur. Namun, ketika berada di ujung pencarian jalur, kunci string kosong tidak melihat ke dalam array.

Namun di Amazon DocumentDB, hanya elemen pertama dalam array yang dibaca, karena `getArrayIndexFromKeyString` mengonversi string kosong menjadi `0`, sehingga pencarian kunci string diperlakukan sebagai pencarian indeks array.

MongoDB API, operasi, dan tipe data

Amazon DocumentDB kompatibel dengan API MongoDB 3.6, 4.0, 5.0, dan 8.0. Untuk daftar terbaru fungsionalitas yang didukung, lihat API MongoDB yang didukung, operasi, dan tipe data di Amazon DocumentDB.

`utilitas mongodump dan mongorestore`

Amazon DocumentDB tidak mendukung basis data admin dan dengan demikian tidak membuang atau memulihkan bssis data admin saat menggunakan `mongodump` atau utilitas `mongorestore`.

Saat Anda membuat basis data baru di Amazon DocumentDB dengan menggunakan `mongorestore`, Anda perlu membuat ulang peran pengguna selain operasi pemulihan.

> **Note**
>
> Kami merekomendasikan MongoDB Database Tools hingga dan termasuk versi 100.6.1 untuk Amazon DocumentDB. Anda dapat mengakses unduhan Alat Database MongoDB di sini.

Pemesanan hasil

Amazon DocumentDB tidak menjamin pengurutan kategori hasil secara impilisit dari kumpulan hasil. Untuk memastikan pengurutan kumpulan hasil, tentukan pengurutan kategori secara eksplisit dengan menggunakan `sort()`.

Contoh berikut mengategorikan item dalam koleksi inventaris dalam urutan menurun berdasarkan bidang stok.

```
db.inventory.find().sort({ stock: -1 })
```

Saat menggunakan tahap agregasi `$sort`, urutan kategori tidak dipertahankan kecuali jika tahap `$sort` tersebut adalah tahap terakhir dalam alur agregasi. Saat menggunakan tahap agregasi `$sort` bersama dengan tahap agregasi `$group`, tahap agregasi `$sort` hanya diterapkan pada akumulator `$first` dan `$last`. Di Amazon DocumentDB 4.0, dukungan ditambahkan untuk `$push` sesuai pengurutan kategori dari tahap `$sort` sebelumnya.

Penulisan yang dapat dicoba lagi

Dimulai dengan driver yang kompatibel dengan MongoDB 4.2, penulisan yang dapat dicoba ulang diaktifkan secara default. Namun, Amazon DocumentDB saat ini tidak mendukung penulisan yang dapat dicoba lagi. Perbedaan fungsional akan memanifestasikan dirinya dalam pesan kesalahan yang mirip dengan hal berikut ini.

```
{"ok":0,"errmsg":"Unrecognized field: 'txnNumber'","code":9,"name":"MongoError"}
```

Tulisan yang dapat dicoba ulang dapat dinonaktifkan melalui string koneksi (misalnya,`MongoClient("mongodb://my.mongodb.cluster/db?retryWrites=false")`)

atau argumen kata kunci MongoClient konstruktor (misalnya,). `MongoClient("mongodb://my.mongodb.cluster/db", retryWrites=False)`

Berikut ini adalah contoh Python yang menonaktifkan penulisan yang dapat dicoba lagi dalam string koneksi.

```
client =
 pymongo.MongoClient('mongodb://
<username>:<password>@docdb-2019-03-17-16-49-12.cluster-ccuszbx3pn5e.us-
east-1.docdb.amazonaws.com:27017/?
replicaSet=rs0',w='majority',j=True,retryWrites=False)
```

Indeks jarang

Untuk menggunakan indeks jarang yang telah Anda buat dalam kueri, Anda harus menggunakan klausa `$exists` pada bidang yang mencakup indeks tersebut. Jika Anda menghilangkan`$exists`, Amazon DocumentDB tidak akan menggunakan indeks jarang.

Berikut ini adalah contoh.

```
db.inventory.count({ "stock": { $exists: true }})
```

Untuk indeks multi-kunci yang jarang, Amazon DocumentDB tidak mendukung batasan kunci unik jika pencarian dokumen menghasilkan sekumpulan nilai dan hanya sebagian dari bidang yang diindeks yang hilang. Misalnya, `createIndex({"a.b" : 1 }, { unique : true, sparse :true })` tidak didukung, mengingat input `"a" : [ { "b" : 2 }, { "c" : 1 } ]`, seperti `"a.c"` yang disimpan dalam indeks.

Menggunakan $elemMatch dalam ekspresi $all

Amazon DocumentDB saat ini tidak mendukung penggunaan operator `$elemMatch` dalam ekspresi `$all`. Sebagai solusinya, Anda dapat menggunakan operator `$and` dengan `$elemMatch` sebagai berikut.

Operasi asli:

```
db.col.find({
  qty: {
    $all: [
```

```
      { "$elemMatch": { part: "xyz", qty: { $lt: 11 } } },
      { "$elemMatch": { num: 40, size: "XL" } }
    ]
  }
})
```

Operasi yang diperbarui:

```
db.col.find({
  $and: [
    { qty: { "$elemMatch": { part: "xyz", qty: { $lt: 11 } } } },
    { qty: { "$elemMatch": { qty: 40, size: "XL" } } }
  ]
})
```

Dolar ($) dan titik (.) dalam nama bidang

Amazon DocumentDB tidak mendukung kueri bidang awalan Dollar ($) di $in, $nin dan $all di objek bersarang. Misalnya, kueri berikut tidak valid di Amazon DocumentDB:

```
coll.find({"field": {"$all": [{ "$a": 1 }]}})
```

$lookup

Amazon DocumentDB mendukung kemampuan untuk melakukan kecocokan kesetaraan (misalnya, gabungan luar kiri) dan juga mendukung subkueri yang tidak berkorelasi, tetapi tidak mendukung subkueri yang berkorelasi.

Memanfaatkan indeks dengan $lookup

Anda sekarang dapat menggunakan indeks dengan operator tahapan `$lookup`. Berdasarkan kasus penggunaan Anda, ada beberapa algoritme pengindeksan yang dapat Anda gunakan untuk mengoptimalkan kinerja. Bagian ini akan menjelaskan berbagai algoritme pengindeksan untuk `$lookup` dan membantu Anda memilih yang terbaik untuk beban kerja Anda.

Secara default, Amazon DocumentDB akan menggunakan algoritme hash saat `allowDiskUse:false` digunakan dan mengurutkan gabungan saat `allowDiskUse:true` digunakan.

> **Note**
>
> `allowDiskUse`Opsi saat ini tidak didukung untuk `find` perintah. Opsi ini hanya didukung sebagai bagian dari agregasi. Sebaiknya gunakan kerangka agregasi `allowDiskUse:true` untuk menangani kueri besar yang mungkin melebihi batas memori.

Untuk beberapa kasus penggunaan, akan lebih bermanfaat jika pengoptimal kueri didesak untuk menggunakan algoritme yang berbeda. Di bawah ini adalah algoritme pengindeksan berbeda yang dapat digunakan operator `$lookup` agregasi:

- Nested loop: Rencana loop bersarang biasanya bermanfaat untuk beban kerja jika koleksi asing <1 GB dan bidang dalam koleksi asing memiliki indeks. Jika algoritma loop bersarang sedang digunakan, rencana penjelasan akan menunjukkan tahap sebagai`NESTED_LOOP_LOOKUP`.
- Penggabungan pengurutan: Rencana penggabungan pengurutan biasanya bermanfaat untuk beban kerja jika koleksi asing tidak memiliki indeks pada bidang yang digunakan dalam pencarian dan kumpulan data yang berfungsi tidak sesuai dengan memori. Jika algoritma penggabungan pengurutan sedang digunakan, rencana penjelasan akan menampilkan tahapan sebagai`SORT_LOOKUP`.
- Hash: Paket hash biasanya bermanfaat untuk beban kerja jika koleksi asing < 1 GB dan kumpulan data yang berfungsi sesuai dengan memori. Jika algoritma hash sedang digunakan, rencana penjelasan akan menunjukkan tahap sebagai`HASH_LOOKUP`.

Anda dapat mengidentifikasi algoritma pengindeksan yang sedang digunakan untuk `$lookup` operator dengan menggunakan `explain` pada query. Di bawah ini adalah contoh:

```
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
}
```

Sebagai alternatif untuk menggunakan `explain()` metode ini, Anda dapat menggunakan profiler untuk meninjau algoritma yang sedang digunakan dengan penggunaan operator Anda. `$lookup` Untuk informasi selengkapnya tentang profiler, lihatMembuat profil operasi Amazon DocumentDB.

Menggunakan **`PlanHint`**

Jika Anda ingin memaksa pengoptimal kueri untuk menggunakan algoritme pengindeksan yang berbeda dengan `$lookup`, Anda dapat menggunakan `planHint`. Untuk melakukannya, gunakan komentar dalam opsi tahap agregasi untuk memaksa rencana yang berbeda. Di bawah ini adalah contoh sintaks untuk komentar:

```
comment : {
    comment :  "<string>",
    lookupStage : { planHint : "SORT" | "HASH" | "NESTED_LOOP" }
}
```

Di bawah ini adalah contoh penggunaan `planHint` untuk memaksa pengoptimal kueri untuk menggunakan algoritme pengindeksan `HASH`:

```
db.foo.aggregate(
   [
      {
         $lookup:
            {
               from: "foo",
               localField: "_id",
               foreignField: "_id",
               as: "joined"
            },
      }
   ]
),
{
   comment : "{ \"lookupStage\" : { \"planHint\": \"HASH\" }}"
```

Untuk menguji algoritme mana yang terbaik untuk beban kerja Anda, Anda dapat menggunakan parameter `executionStats` dari metode `explain` untuk mengukur waktu eksekusi tahapan `$lookup` saat memodifikasi algoritme pengindeksan (yaitu, `HASH`/`SORT`/`NESTED_LOOP`).

Contoh berikut menunjukkan cara menggunakan `executionStats` untuk mengukur waktu eksekusi tahap `$lookup` dengan menggunakan algoritme `SORT`.

```
db.foo.explain("executionStats").aggregate(
   [
      {
         $lookup:
            {
               from: "foo",
               localField: "_id",
               foreignField: "_id",
               as: "joined"
            },
      }
   ]
),
{
   comment : "{ \"lookupStage\" : { \"planHint\": \"SORT\" }}"
```

`$ penyortiran alami` dan terbalik

Amazon DocumentDB hanya `$natural` mendukung pemindaian koleksi maju. Pemindaian koleksi terbalik (`{$natural: -1}`) akan mengarah ke a`MongoServerError`.

API MongoDB yang didukung, operasi, dan tipe data di Amazon DocumentDB

Amazon DocumentDB (dengan kompatibilitas MongoDB) adalah layanan basis data dokumen yang cepat, dapat diskalakan, sangat tersedia, dan terkelola penuh yang mendukung beban kerja MongoDB. Amazon DocumentDB kompatibel dengan API MongoDB 3.6, 4.0, 5.0, dan 8.0. Bagian ini mencantumkan fungsionalitas yang didukung. Untuk dukungan menggunakan API dan driver MongoDB, lihat Forum Komunitas MongoDB. Untuk dukungan menggunakan layanan Amazon DocumentDB, hubungi tim dukungan yang AWS sesuai. Untuk perbedaan fungsional antara Amazon DocumentDB dan MongoDB, lihat. Perbedaan fungsional: Amazon DocumentDB dan MongoDB

Perintah dan operator MongoDB yang hanya berlaku untuk internal atau tidak berlaku untuk layanan terkelola penuh tidak didukung dan tidak termasuk dalam daftar fungsi yang didukung.

Kami telah menambahkan lebih dari 50+ kemampuan tambahan sejak diluncurkan, dan akan terus bekerja mundur dari pelanggan kami untuk memberikan kemampuan yang mereka butuhkan. Untuk informasi tentang peluncuran terbaru, lihat Pengumuman Amazon DocumentDB.

Jika ada fitur yang tidak didukung yang Anda ingin kami bangun, beri tahu kami dengan mengirimkan email dengan accountID Anda, fitur yang diminta, dan kasus penggunaan ke tim layanan Amazon DocumentDB.

Topik

- Perintah basis data
- Operator kueri dan proyeksi
- Perbarui operator
- Geospasial
- Metode kursor
- Operator pipa agregasi
- Jenis Data
- Indeks

Perintah basis data

Topik

- Perintah administratif
- Agregasi
- Autentikasi
- Perintah diagnostik
- Operasi kueri dan tulis
- Perintah manajemen peran
- Perintah sesi
- Manajemen pengguna
- Perintah sharding

Perintah administratif

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
Koleksi Terbatas	Tidak	Tidak	Tidak	Tidak	Tidak
klon CollectionAsCapped	Tidak	Tidak	Tidak	Tidak	Tidak
collMod	Sebagian	Sebagian	Sebagian	Sebagian	Sebagian
CollMod: kedaluwarsaAfterSeconds	Ya	Ya	Ya	Ya	Ya
mengkonversi ToCapped	Tidak	Tidak	Tidak	Tidak	Tidak
copydb	Tidak	Tidak	Tidak	Tidak	Tidak

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
Buat	Ya	Ya	Ya	Ya	Ya
createView	Tidak	Tidak	Tidak	Ya	Tidak
createIndex	Ya	Ya	Ya	Ya	Ya
currentOp	Ya	Ya	Ya	Ya	Ya
drop	Ya	Ya	Ya	Ya	Ya
dropDatabase	Ya	Ya	Ya	Ya	Ya
dropIndexes	Ya	Ya	Ya	Ya	Ya
filemd5	Tidak	Tidak	Tidak	Tidak	Tidak
mendapatkanAuditConfig	Tidak	Ya	Ya	Ya	Tidak
killCursors	Ya	Ya	Ya	Ya	Ya
killOp	Ya	Ya	Ya	Ya	Ya
ListCollection*	Ya	Ya	Ya	Ya	Ya
listDatabases	Ya	Ya	Ya	Ya	Ya
listIndexes	Ya	Ya	Ya	Ya	Ya
reIndex	Tidak	Tidak	Ya	Ya	Tidak
renameCollection	Ya	Ya	Ya	Ya	Tidak
set AuditConfig	Tidak	Ya	Ya	Ya	Tidak

* `type` Kunci dalam opsi filter tidak didukung.

Agregasi

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
agregasi	Ya	Ya	Ya	Ya	Ya
hitungan	Ya	Ya	Ya	Ya	Ya
berbeda	Ya	Ya	Ya	Ya	Ya
mapReduce	Tidak	Tidak	Tidak	Ya	Tidak

Autentikasi

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
diautentikasi	Ya	Ya	Ya	Ya	Ya
logout	Ya	Ya	Ya	Ya	Ya

Perintah diagnostik

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
buildInfo	Ya	Ya	Ya	Ya	Ya
collStats	Ya	Ya	Ya	Ya	Ya
conn PoolStats	Tidak	Tidak	Tidak	Tidak	Tidak

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
connectionStatus	Ya	Ya	Ya	Ya	Ya
dataSize	Ya	Ya	Ya	Ya	Ya
dbHash	Tidak	Tidak	Tidak	Tidak	Tidak
dbStats	Ya	Ya	Ya	Ya	Ya
jelaskan	Ya	Ya	Ya	Ya	Ya
jelaskan: executionStats	Ya	Ya	Ya	Ya	Ya
fitur	Tidak	Tidak	Tidak	Tidak	Tidak
hostInfo	Ya	Ya	Ya	Ya	Ya
listCommands	Ya	Ya	Ya	Ya	Ya
profiler	Ya	Ya	Ya	Ya	Tidak
serverStatus	Ya	Ya	Ya	Ya	Ya
atas	Ya	Ya	Ya	Ya	Ya

Operasi kueri dan tulis

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
Mengelola aliran	Ya	Ya	Ya	Ya	Tidak

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
hapus	Ya	Ya	Ya	Ya	Ya
cari	Ya	Ya	Ya	Ya	Ya
menemukanAndModify	Ya	Ya	Ya	Ya	Ya
mendapatkanLastError	Tidak	Tidak	Tidak	Tidak	Tidak
getMore	Ya	Ya	Ya	Ya	Ya
mendapatkanPrevError	Tidak	Tidak	Tidak	Tidak	Tidak
GridFS	Ya	Ya	Ya	Ya	Tidak
sisipkan	Ya	Ya	Ya	Ya	Ya
paralelCollectionScan	Tidak	Tidak	Tidak	Tidak	Tidak
resetError	Tidak	Tidak	Tidak	Tidak	Tidak
perbarui	Ya	Ya	Ya	Ya	Ya
ReplaceOne	Ya	Ya	Ya	Ya	Ya

Perintah manajemen peran

F	3	4	5	8	C e
cYɛYɛYɛYɛYɛ e					idak

F	3	4	5	8	C e
j A r s	Ya	Ya	Ya	Ya	Tidak
c	Ya	Ya	Ya	Ya	Tidak
h F l	Ya	Ya	Ya	Ya	Tidak
n F F	Ya	Ya	Ya	Ya	Tidak
n F s	Ya	Ya	Ya	Ya	Tidak
n	Ya	Ya	Ya	Ya	Tidak
u e	Ya	Ya	Ya	Ya	Tidak

Perintah sesi

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
abortTransaction	Tidak	Ya	Ya	Ya	Tidak
commitTransaction	Tidak	Ya	Ya	Ya	Tidak

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
endSessions	Tidak	Tidak	Tidak	Tidak	Tidak
membunuh AllSessions	Tidak	Ya	Ya	Ya	Tidak
membunuh AllSessio nsByPattern	Tidak	Tidak	Tidak	Tidak	Tidak
killSessions	Tidak	Ya	Ya	Ya	Tidak
refreshse ssions	Tidak	Tidak	Tidak	Tidak	Tidak
startSession	Tidak	Ya	Ya	Ya	Tidak

Manajemen pengguna

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
createUser	Ya	Ya	Ya	Ya	Ya
jatuhkan AllUsersF romDatabase	Ya	Ya	Ya	Ya	Ya
dropUser	Ya	Ya	Ya	Ya	Ya
hibah RolesToUser	Ya	Ya	Ya	Ya	Ya
mencabut RolesFrom User	Ya	Ya	Ya	Ya	Ya

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
updateUser	Ya	Ya	Ya	Ya	Ya
UsersInfo	Ya	Ya	Ya	Ya	Ya

Perintah sharding

Perintah	Cluster elastis
menggugurkan ReshardCollection	Tidak
AddsHard	Tidak
menambahkan ShardToZone	Tidak
penyeimbang CollectionStatus	Tidak
BalancerStart	Tidak
BalancerStatus	Tidak
BalancerSTOP	Tidak
periksa ShardingIndex	Tidak
jelas JumboFlag	Tidak
cleanupOrphaned	Tidak
pembersihan ReshardCollection	Tidak
berkomitmen ReshardCollection	Tidak
Aktifkan Sharding	Ya
siram RouterConfig	Tidak
mendapatkan ShardMap	Tidak

Perintah	Cluster elastis
mendapatkan ShardVersion	Tidak
ISDBgrid	Tidak
ListShards	Tidak
MedianKey	Tidak
MoveChunk	Tidak
MovePrimary	Tidak
MergeChunks	Tidak
menyempurnakan CollectionShardKey	Tidak
MenghapusHard	Tidak
menghapus ShardFromZone	Tidak
ResHardCollection	Tidak
set AllowMigrations	Tidak
set ShardVersion	Tidak
ShardCollection	Ya
ShardingState	Tidak
terbelah	Tidak
SplitVector	Tidak
UnsetSharding	Tidak
perbarui ZoneKeyRange	Tidak

Operator kueri dan proyeksi

Topik

- Operator Array
- Operator Bitwise
- Operator komentar
- Operator perbandingan
- Operator elemen
- Operator kueri evaluasi
- Operator logis
- Operator proyeksi

Operator Array

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$ semua	Ya	Ya	Ya	Ya	Ya
$elemMatch	Ya	Ya	Ya	Ya	Ya
$size	Ya	Ya	Ya	Ya	Ya

Operator Bitwise

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$ bit AllSet	Ya	Ya	Ya	Ya	Ya
$ bit AnySet	Ya	Ya	Ya	Ya	Ya
$ bit AllClear	Ya	Ya	Ya	Ya	Ya

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$ bit AnyClear	Ya	Ya	Ya	Ya	Ya

Operator komentar

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$comment	Ya	Ya	Ya	Ya	Ya

Operator perbandingan

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$eq	Ya	Ya	Ya	Ya	Ya
$gt	Ya	Ya	Ya	Ya	Ya
$gte	Ya	Ya	Ya	Ya	Ya
$in	Ya	Ya	Ya	Ya	Ya
$lt	Ya	Ya	Ya	Ya	Ya
$lte	Ya	Ya	Ya	Ya	Ya
$ne	Ya	Ya	Ya	Ya	Ya
$nin	Ya	Ya	Ya	Ya	Ya

Operator elemen

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$exists	Ya	Ya	Ya	Ya	Ya
$type	Ya	Ya	Ya	Ya	Ya

Operator kueri evaluasi

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$expr	Tidak	Ya	Ya	Ya	Tidak
$jsonSchema	Tidak	Ya	Ya	Ya	Tidak
$mod	Ya	Ya	Ya	Ya	Ya
$regex	Ya	Ya	Ya	Ya	Ya
$text	Tidak	Tidak	Ya	Ya	Tidak
$where	Tidak	Tidak	Tidak	Tidak	Tidak

Operator logis

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$and	Ya	Ya	Ya	Ya	Ya
$nor	Ya	Ya	Ya	Ya	Ya
$not	Ya	Ya	Ya	Ya	Ya

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$or	Ya	Ya	Ya	Ya	Ya

Operator proyeksi

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$	Ya	Ya	Ya	Ya	Ya
$elemMatch	Ya	Ya	Ya	Ya	Ya
$meta	Tidak	Tidak	Ya	Ya	Tidak
$slice	Ya	Ya	Ya	Ya	Ya

Perbarui operator

Topik

- Operator array
- Operator Bitwise
- Operator lapangan
- Perbarui pengubah

Operator array

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$	Ya	Ya	Ya	Ya	Ya
$[]	Ya	Ya	Ya	Ya	Ya

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$ [<pengidentifikasi>]	Ya	Ya	Ya	Ya	Ya
$ tambahkanToSet	Ya	Ya	Ya	Ya	Ya
$ pop	Ya	Ya	Ya	Ya	Ya
$ PullAll	Ya	Ya	Ya	Ya	Ya
$ tarik	Ya	Ya	Ya	Ya	Ya
$ dorong	Ya	Ya	Ya	Ya	Ya

Operator Bitwise

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$ bit	Ya	Ya	Ya	Ya	Ya

Operator lapangan

Operator	3.6	4.0	5.0	8.0	Cluster elastis
$CurrentDate	Ya	Ya	Ya	Ya	Ya
$ inc	Ya	Ya	Ya	Ya	Ya
$max	Ya	Ya	Ya	Ya	Ya
$min	Ya	Ya	Ya	Ya	Ya

Operator	3.6	4.0	5.0	8.0	Cluster elastis
$ mul	Ya	Ya	Ya	Ya	Ya
$ ganti nama	Ya	Ya	Ya	Ya	Ya
$set	Ya	Ya	Ya	Ya	Ya
$ set OnInsert	Ya	Ya	Ya	Ya	Ya
$unset	Ya	Ya	Ya	Ya	Ya

Perbarui pengubah

Operator	3.6	4.0	5.0	8.0	Cluster elastis
$ masing-masing	Ya	Ya	Ya	Ya	Ya
$ posisi	Ya	Ya	Ya	Ya	Ya
$slice	Ya	Ya	Ya	Ya	Ya
$sort	Ya	Ya	Ya	Ya	Ya

Geospasial

Penentu geometri

Pemilih Kueri	3.6	4.0	5.0	8.0	Cluster elastis
$box	Tidak	Tidak	Tidak	Tidak	Tidak

Pemilih Kueri	3.6	4.0	5.0	8.0	Cluster elastis
$center	Tidak	Tidak	Tidak	Tidak	Tidak
$centerSphere	Tidak	Tidak	Tidak	Tidak	Tidak
$geometry	Ya	Ya	Ya	Ya	Ya
$maxDistance	Ya	Ya	Ya	Ya	Ya
$minDistance	Ya	Ya	Ya	Ya	Ya
$nearSphere	Ya	Ya	Ya	Ya	Ya
$polygon	Tidak	Tidak	Tidak	Tidak	Tidak
$uniqueDocs	Tidak	Tidak	Tidak	Tidak	Tidak

Penyeleksi kueri

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$geointersects	Ya	Ya	Ya	Ya	Ya
$geowithin	Ya	Ya	Ya	Ya	Ya
$near	Ya	Ya	Ya	Ya	Ya
$nearSphere	Ya	Ya	Ya	Ya	Ya
$polygon	Tidak	Tidak	Tidak	Tidak	Tidak
$uniqueDocs	Tidak	Tidak	Tidak	Tidak	Tidak

Metode kursor

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
cursor.batchsize ()	Ya	Ya	Ya	Ya	Ya
cursor.close ()	Ya	Ya	Ya	Ya	Ya
cursor.collation()	Tidak	Tidak	Tidak	Ya	Tidak
cursor.comment ()	Ya	Ya	Ya	Ya	Ya
cursor.count ()	Ya	Ya	Ya	Ya	Ya
cursor.explain ()	Ya	Ya	Ya	Ya	Tidak
cursor.foreach ()	Ya	Ya	Ya	Ya	Ya
cursor.hasNext ()	Ya	Ya	Ya	Ya	Ya
cursor.hint ()	Ya	Ya	Ya	Ya	Ya*
cursor.isClosed ()	Ya	Ya	Ya	Ya	Ya
cursor.isExhausted ()	Ya	Ya	Ya	Ya	Tidak
cursor.itcount ()	Ya	Ya	Ya	Ya	Tidak

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
cursor.limit ()	Ya	Ya	Ya	Ya	Tidak
cursor.map ()	Ya	Ya	Ya	Ya	Tidak
cursor.max ()	Tidak	Tidak	Tidak	Tidak	Tidak
cursor.maxScan ()	Ya	Ya	Ya	Ya	Tidak
cursor.maxTimes ()	Ya	Ya	Ya	Ya	Tidak
cursor.min ()	Tidak	Tidak	Tidak	Tidak	Tidak
cursor.next ()	Ya	Ya	Ya	Ya	Ya
kursor.noCursorTimeout ()	Tidak	Tidak	Tidak	Tidak	Tidak
kursor.objsLeftInBatch ()	Ya	Ya	Ya	Ya	Tidak
cursor.pretty ()	Ya	Ya	Ya	Ya	Tidak
cursor.readconcern ()	Ya	Ya	Ya	Ya	Tidak
cursor.readPref ()	Ya	Ya	Ya	Ya	Tidak
cursor.returnKey ()	Tidak	Tidak	Tidak	Tidak	Tidak
kursor.showRecordId ()	Tidak	Tidak	Tidak	Tidak	Tidak

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
cursor.size ()	Ya	Ya	Ya	Ya	Tidak
cursor.skip ()	Ya	Ya	Ya	Ya	Tidak
cursor.sort ()	Ya	Ya	Ya	Ya	Tidak
cursor.ta ilable ()	Tidak	Tidak	Tidak	Tidak	Tidak
cursor.to Array ()	Ya	Ya	Ya	Ya	Tidak

* Indeks `hint` didukung dengan ekspresi indeks. Misalnya, `db.foo.find().hint({x:1})`.

Operator pipa agregasi

Topik

- Ekspresi akumulator
- Operator aritmatika
- Operator array
- Operator Boolean
- Operator perbandingan
- Operator ekspresi bersyarat
- Operator tipe data
- Operator ukuran data
- Operator tanggal
- Operator literal
- Gabungkan operator
- Operator alami
- Tetapkan operator
- Operator panggung

- Operator String
- Variabel sistem
- Operator pencarian teks
- Jenis operator konversi
- Operator variabel
- Operator lain-lain

Ekspresi akumulator

Ekspresi	3.6	4.0	5.0	8.0	Cluster elastis
$ akumulator	-	-	Tidak	Tidak	Tidak
$ tambahkan ToSet	Ya	Ya	Ya	Ya	Ya
$avg	Ya	Ya	Ya	Ya	Ya
$count	-	-	Tidak	Tidak	Tidak
$ KovariancePOP	Tidak	Tidak	Tidak	Tidak	Tidak
$ kovarianceSamp	Tidak	Tidak	Tidak	Tidak	Tidak
$ DenseRank	Tidak	Tidak	Tidak	Tidak	Tidak
$ turunan	Tidak	Tidak	Tidak	Tidak	Tidak
$DocumentNumber	Tidak	Tidak	Tidak	Tidak	Tidak
$ expMovingAvg	Tidak	Tidak	Tidak	Tidak	Tidak
$first	Ya	Ya	Ya	Ya	Ya

Ekspresi	3.6	4.0	5.0	8.0	Cluster elastis
$ integral	Tidak	Tidak	Tidak	Tidak	Tidak
$last	Ya	Ya	Ya	Ya	Ya
$max	Ya	Ya	Ya	Ya	Ya
$min	Ya	Ya	Ya	Ya	Ya
$push	Ya	Ya	Ya	Ya	Ya
$ peringkat	Tidak	Tidak	Tidak	Tidak	Tidak
$ shift	Tidak	Tidak	Tidak	Tidak	Tidak
$ std DevPop	Tidak	Tidak	Tidak	Tidak	Tidak
$ std DevSamp	Tidak	Tidak	Tidak	Tidak	Tidak
$sum	Ya	Ya	Ya	Ya	Ya

Operator aritmatika

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$abs	Ya	Ya	Ya	Ya	Ya
$add	Ya	Ya	Ya	Ya	Ya
$ceil	Tidak	Ya	Ya	Ya	Ya
$divide	Ya	Ya	Ya	Ya	Ya
$exp	Tidak	Ya	Ya	Ya	Ya
$floor	Tidak	Ya	Ya	Ya	Ya

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$ln	Tidak	Ya	Ya	Ya	Ya
$log	Tidak	Ya	Ya	Ya	Ya
$log10	Tidak	Ya	Ya	Ya	Ya
$mod	Ya	Ya	Ya	Ya	Ya
$multiply	Ya	Ya	Ya	Ya	Ya
$pow	Tidak	Tidak	Ya (5.0.1+)	Ya	Tidak
$ bulat	-	-	Tidak	Tidak	Tidak
$sqrt	Tidak	Ya	Ya	Ya	Ya
$subtract	Ya	Ya	Ya	Ya	Ya
$trunc	Tidak	Tidak	Tidak	Tidak	Tidak

Operator array

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$ array ElemAt	Ya	Ya	Ya	Ya	Ya
$ array ToObject	Ya	Ya	Ya	Ya	Ya
$concatArrays	Ya	Ya	Ya	Ya	Ya
$filter	Ya	Ya	Ya	Ya	Ya

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$first	-	-	Ya	Ya	Tidak
$in	Ya	Ya	Ya	Ya	Ya
$ indeks OfArray	Ya	Ya	Ya	Ya	Ya
$isArray	Ya	Ya	Ya	Ya	Ya
$last	-	-	Ya	Ya	Tidak
$ objek ToArray	Ya	Ya	Ya	Ya	Ya
$range	Ya	Ya	Ya	Ya	Ya
$reverseA rray	Ya	Ya	Ya	Ya	Ya
$reduce	Ya	Ya	Ya	Ya	Ya
$size	Ya	Ya	Ya	Ya	Ya
$slice	Ya	Ya	Ya	Ya	Ya
$zip	Ya	Ya	Ya	Ya	Ya

Operator Boolean

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$and	Ya	Ya	Ya	Ya	Ya
$not	Ya	Ya	Ya	Ya	Ya

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$or	Ya	Ya	Ya	Ya	Ya

Operator perbandingan

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$cmp	Ya	Ya	Ya	Ya	Ya
$eq	Ya	Ya	Ya	Ya	Ya
$gt	Ya	Ya	Ya	Ya	Ya
$gte	Ya	Ya	Ya	Ya	Ya
$lt	Ya	Ya	Ya	Ya	Ya
$lte	Ya	Ya	Ya	Ya	Ya
$ne	Ya	Ya	Ya	Ya	Ya

Operator ekspresi bersyarat

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$cond	Ya	Ya	Ya	Ya	Ya
$ifNull	Ya	Ya	Ya	Ya	Ya
$switch	Tidak	Ya	Ya	Ya	Tidak

Operator tipe data

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$type	Ya	Ya	Ya	Ya	Ya

Operator ukuran data

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$ BinarySize	-	-	Tidak	Tidak	Tidak
$ BSONSize	-	-	Tidak	Tidak	Tidak

Operator tanggal

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$ DateAdd	Tidak	Tidak	Ya	Ya	Ya
$ DateDiff	-	-	Ya	Ya	Tidak
$ tanggal FromParts	Tidak	Tidak	Ya (5.0.1+)	Tidak	Tidak
$ tanggal FromString	Ya	Ya	Ya	Ya	Ya
$ DateSubtract	Tidak	Tidak	Ya	Ya	Ya
$ tanggal ToParts	Tidak	Tidak	Ya (5.0.1+)	Tidak	Tidak

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$ tanggal ToString	Ya	Ya	Ya	Ya	Ya
$ DateTrunc	-	-	Tidak	Ya	Tidak
$ hari OfMonth	Ya	Ya	Ya	Ya	Ya
$ hari OfWeek	Ya	Ya	Ya	Ya	Ya
$ hari OfYear	Ya	Ya	Ya	Ya	Ya
$hour	Ya	Ya	Ya	Ya	Ya
$ iso DayOfWeek	Ya	Ya	Ya	Ya	Ya
$isoWeek	Ya	Ya	Ya	Ya	Ya
$ iso WeekYear	Ya	Ya	Ya	Ya	Ya
$millisecond	Ya	Ya	Ya	Ya	Ya
$minute	Ya	Ya	Ya	Ya	Ya
$month	Ya	Ya	Ya	Ya	Ya
$second	Ya	Ya	Ya	Ya	Ya
$week	Ya	Ya	Ya	Ya	Ya
$year	Ya	Ya	Ya	Ya	Ya

Operator literal

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$literal	Ya	Ya	Ya	Ya	Ya

Gabungkan operator

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$mergeObjects	Ya	Ya	Ya	Ya	Ya

Operator alami

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$natural	Ya	Ya	Ya	Ya	Ya

Tetapkan operator

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$ semua ElementsTrue	Tidak	Ya	Ya	Ya	Ya
$ apa saja ElementTrue	Tidak	Ya	Ya	Ya	Ya

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$setDifference	Tidak	Ya	Ya	Ya	Ya
$setEquals	Ya	Ya	Ya	Ya	Ya
$setIntersection	Ya	Ya	Ya	Ya	Ya
$ setIsSubset	Ya	Ya	Ya	Ya	Ya
$setUnion	Ya	Ya	Ya	Ya	Ya
$ setWindowFields	Tidak	Tidak	Tidak	Tidak	Tidak

Operator panggung

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$addFields	Ya	Ya	Ya	Ya	Ya
$bucket	Tidak	Tidak	Tidak	Ya	Tidak
$bucketAuto	Tidak	Tidak	Tidak	Tidak	
$ ChangeStream	Ya	Ya	Ya	Ya	Tidak
$collStats	Tidak	Ya	Ya	Ya	Tidak
$count	Ya	Ya	Ya	Ya	Ya
$currentOp	Ya	Ya	Ya	Ya	Ya

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$facet	Tidak	Tidak	Tidak	Tidak	Tidak
$geoNear	Ya	Ya	Ya	Ya	Ya
$graphLookup	Tidak	Tidak	Tidak	Tidak	Tidak
$group	Ya	Ya	Ya	Ya	Ya
$indexStats	Ya	Ya	Ya	Ya	Ya
$limit	Ya	Ya	Ya	Ya	Ya
$ daftar LocalSessions	Tidak	Tidak	Tidak	Tidak	Tidak
$listSessions	Tidak	Tidak	Tidak	Tidak	Tidak
$lookup	Ya	Ya	Ya	Ya	Ya
$match	Ya	Ya	Ya	Ya	Ya
$ bergabung	-	-	Tidak	Ya	Tidak
$out	Ya	Ya	Ya	Ya	Tidak
$ rencana CacheStats	-	-	Tidak	Tidak	Tidak
$project	Ya	Ya	Ya	Ya	Ya
$redact	Ya	Ya	Ya	Ya	Ya
$replaceRoot	Ya	Ya	Ya	Ya	Ya
$sample	Ya	Ya	Ya	Ya	Ya

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$set	-	-	Tidak	Ya	Tidak
$ set WindowFields	-	-	Tidak	Tidak	Tidak
$skip	Ya	Ya	Ya	Ya	Ya
$sort	Ya	Ya	Ya	Ya	Ya
$ sortir ByCount	Tidak	Tidak	Tidak	Tidak	Tidak
$unionDengan	-	-	Tidak	Tidak	Tidak
$unset	-	-	Tidak	Ya	Tidak
$unwind	Ya	Ya	Ya	Ya	Ya
$ GantiDengan	Tidak	Tidak	Tidak	Ya	Tidak
$ VectorSearch	Tidak	Tidak	Tidak	Ya	Tidak

Operator String

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$concat	Ya	Ya	Ya	Ya	Ya
$ indeksOfBytes	Ya	Ya	Ya	Ya	Ya

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$indexOfCP	Ya	Ya	Ya	Ya	Ya
$ltrim	Tidak	Ya	Ya	Ya	Tidak
$RegexFind	-	-	Ya	Ya	Tidak
$ regex FindAll	-	-	Ya	Ya	Tidak
$RegexMatch	-	-	Ya	Ya	Tidak
$ GantiSemua	-	-	Ya	Ya	Tidak
$ GantiSatu	-	-	Ya	Ya	Tidak
$rtrim	Tidak	Ya	Ya	Ya	Tidak
$split	Ya	Ya	Ya	Ya	Ya
$strcasecmp	Ya	Ya	Ya	Ya	Ya
$ str LenBytes	Ya	Ya	Ya	Ya	Ya
$strLenCP	Ya	Ya	Ya	Ya	Ya
$substr	Ya	Ya	Ya	Ya	Ya
$substrBytes	Ya	Ya	Ya	Ya	Ya
$substrCP	Ya	Ya	Ya	Ya	Ya
$toLower	Ya	Ya	Ya	Ya	Ya
$toUpper	Ya	Ya	Ya	Ya	Ya
$trim	Tidak	Ya	Ya	Ya	Tidak

Variabel sistem

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$$CURRENT	Tidak	Tidak	Tidak	Tidak	Tidak
$$DESCEND	Ya	Ya	Ya	Ya	Ya
$$KEEP	Ya	Ya	Ya	Ya	Ya
$$PRUNE	Ya	Ya	Ya	Ya	Ya
$$REMOVE	Tidak	Tidak	Tidak	Tidak	Tidak
$ AKAR	Ya	Ya	Ya	Ya	Ya

Operator pencarian teks

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$meta	Tidak	Tidak	Ya	Ya	Tidak
$ pencarian	Tidak	Tidak	Ya	Ya	Tidak

Jenis operator konversi

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$convert	Tidak	Ya	Ya	Ya	Ya
$ isNumber	-	-	Tidak	Tidak	Tidak
$toBool	Tidak	Ya	Ya	Ya	Ya

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$toDate	Tidak	Ya	Ya	Ya	Ya
$toDecimal	Tidak	Ya	Ya	Ya	Ya
$toDouble	Tidak	Ya	Ya	Ya	Ya
$toInt	Tidak	Ya	Ya	Ya	Ya
$toLong	Tidak	Ya	Ya	Ya	Ya
$ untuk ObjectId	Tidak	Ya	Ya	Ya	Ya
$toString	Tidak	Ya	Ya	Ya	Ya

Operator variabel

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$let	Ya	Ya	Ya	Ya	Ya
$map	Ya	Ya	Ya	Ya	Ya

Operator lain-lain

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
$ GetField	-	-	Tidak	Tidak	Tidak
$ rand	-	-	Ya (5.0.1+)	Ya	Tidak
$ Samplerate	-	-	Tidak	Tidak	Tidak

Jenis Data

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
32-bit Integer (int)	Ya	Ya	Ya	Ya	Ya
64-bit Integer (long	Ya	Ya	Ya	Ya	Ya
Array	Ya	Ya	Ya	Ya	Ya
Data Biner	Ya	Ya	Ya	Ya	Ya
Boolean	Ya	Ya	Ya	Ya	Ya
Date	Ya	Ya	Ya	Ya	Ya
DBPointer	Tidak	Tidak	Tidak	Tidak	Tidak
dbREFS	Tidak	Tidak	Tidak	Tidak	Tidak
Decimal128	Ya	Ya	Ya	Ya	Ya
Ganda	Ya	Ya	Ya	Ya	Ya
JavaScript	Tidak	Tidak	Tidak	Tidak	Tidak
JavaScript(dengan ruang lingkup)	Tidak	Tidak	Tidak	Tidak	Tidak
MaxKey	Ya	Ya	Ya	Ya	Ya
MinKey	Ya	Ya	Ya	Ya	Ya
Null	Ya	Ya	Ya	Ya	Ya

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
Objek	Ya	Ya	Ya	Ya	Ya
ObjectId	Ya	Ya	Ya	Ya	Ya
Ekspresi reguler	Ya	Ya	Ya	Ya	Ya
Tali	Ya	Ya	Ya	Ya	Ya
Simbol	Tidak	Tidak	Tidak	Tidak	Tidak
Stempel waktu	Ya	Ya	Ya	Ya	Ya
Tidak terdefinisi	Tidak	Tidak	Tidak	Tidak	Tidak

Indeks dan properti indeks

Topik

- Indeks
- Properti indeks

Indeks

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
2dsphere	Ya	Ya	Ya	Ya	Ya
Indeks 2d	Tidak	Tidak	Tidak	Tidak	Tidak
Indeks Gabungan	Ya	Ya	Ya	Ya	Ya

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
Indeks hashed	Tidak	Tidak	Tidak	Tidak	Tidak
Indeks Multikey	Ya	Ya	Ya	Ya	Ya
Indeks Bidang Tunggal	Ya	Ya	Ya	Ya	Ya
Indeks Teks	Tidak	Tidak	Ya	Ya	Tidak
Wildcard	Tidak	Tidak	Tidak	Tidak	Tidak

Properti indeks

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
Latar Belakang	Ya	Ya	Ya	Ya	Ya
Tidak peka huruf besar/ kecil	Tidak	Tidak	Tidak	Ya	Tidak
Tersembunyi	Tidak	Tidak	Tidak	Tidak	Tidak
Sebagian	Tidak	Tidak	Ya	Ya	Tidak
Jarang	Ya	Ya	Ya	Ya	Ya
Teks	Tidak	Tidak	Ya	Ya	Tidak
TTL	Ya	Ya	Ya	Ya	Ya

Perintah	3.6	4.0	5.0	8.0	Cluster elastis
Unik	Ya	Ya	Ya	Ya	Ya
vektor	Tidak	Tidak	Ya	Ya	Tidak

Untuk informasi rinci tentang operator MongoDB tertentu, lihat topik berikut:

- Operator pipa agregasi
- Geospasial
- Operator proyeksi
- Perbarui operator

Operator pipa agregasi

Bagian ini memberikan informasi terperinci tentang operator pipeline agregasi yang didukung oleh Amazon DocumentDB.

Topik

- $$DESCEND
- $$KEEP
- $$PRUNE
- $ AKAR
- $abs
- $add
- $ tambahkan ToSet
- $addFields
- $ semua ElementsTrue
- $and
- $ apa saja ElementTrue
- $ array ElemAt
- $ array ToObject

- $avg
- $bucket
- $ceil
- $ ChangeStream
- $cmp
- $collStats
- $concat
- $concatArrays
- $cond
- $convert
- $count
- $currentOp
- $ DateAdd
- $ DateDiff
- $ tanggal FromString
- $ DateSubtract
- $ tanggal ToString
- $ DateTrunc
- $ hari OfMonth
- $ hari OfWeek
- $ hari OfYear
- $divide
- $eq
- $exp
- $filter
- $first
- $floor
- $geoNear
- $group

- $gt
- $gte
- $hour
- $ifNull
- $in
- $ indeks OfArray
- $ indeks OfBytes
- $indexOfCP
- $indexStats
- $isArray
- $ iso DayOfWeek
- $isoWeek
- $ iso WeekYear
- $last
- $let
- $limit
- $literal
- $ln
- $log
- $log10
- $lt
- $lte
- $lookup
- $ltrim
- $map
- $match
- $max
- $meta
- $ bergabung
- $mergeObjects

- $millisecond
- $min
- $minute
- $mod
- $month
- $multiply
- $natural
- $ne
- $not
- $ objek ToArray
- $or
- $out
- $pow
- $push
- $project
- $ rand
- $range
- $redact
- $reduce
- $RegexFind
- $ regex FindAll
- $RegexMatch
- $ GantiSemua
- $ GantiSatu
- $replaceRoot
- $ GantiDengan
- $reverseArray
- $rtrim
- $sample
- $ pencarian

- $second
- $set
- $setDifference
- $setEquals
- $setIntersection
- $ set IsSubset
- $setUnion
- $skip
- $slice
- $size
- $sort
- $split
- $sqrt
- $ str LenBytes
- $strLenCP
- $strcasecmp
- $substr
- $substrBytes
- $substrCP
- $subtract
- $sum
- $switch
- $toBool
- $toDate
- $toDecimal
- $toDouble
- $toInt
- $toLong
- $toLower
- $ untuk ObjectId

- $toString
- $toUpper
- $trim
- $type
- $unset
- $unwind
- $ VectorSearch
- $week
- $year
- $zip

$$DESCEND

`$$DESCEND`Operator di Amazon DocumentDB adalah operator array posisi khusus yang digunakan dalam tahap pipeline. `$redact` Ini menginstruksikan pipa agregasi untuk turun ke dokumen saat ini dan memproses semua bidang, terlepas dari tingkat sarangnya.

Ketika `$redact` tahap bertemu dengan `$$DESCEND` operator, itu akan membuat semua bidang dalam dokumen saat ini terlihat dan memprosesnya lebih jauh ke bawah pipa. Ini berguna ketika Anda ingin menyunting atau memangkas bidang tertentu secara selektif berdasarkan suatu kondisi, sambil mempertahankan struktur dokumen.

Parameter

Tidak ada.

Contoh (MongoDB Shell)

Dalam contoh ini, kita akan menggunakan `$redact` panggung dengan `$$DESCEND` operator untuk secara selektif menampilkan dokumen di mana `code` bidang sama dengan “Reg”.

Buat dokumen sampel

```
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" }
]);
```

Contoh kueri

```
db.patient.aggregate([
  { $redact: {
    $cond: {
      if: { $eq: ["Reg", "$code"] },
      then: "$$DESCEND",
      else: "$$PRUNE"
    }
  }}
]);
```

Keluaran

```
{
  "_id": 2,
  "code": "Reg",
  "patient": "Jane Doe",
  "DOB": "3/27/1989",
  "Hospital": "Cherry Hill"
}
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan $$DESCEND perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

Variabel `$$KEEP` sistem digunakan dengan `$redact` tahapan dalam pipa agregasi untuk menjaga dokumen atau bidang saat ini tidak berubah dan memasukkannya ke dalam output.

Parameter

Tidak ada

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$$KEEP` dalam pipeline agregasi Amazon DocumentDB. Dokumen hanya disimpan jika akses sama dengan “publik”, jika tidak maka akan dihapus.

Buat dokumen sampel

```
db.articles.insertMany([
  { title: "Article A", access: "public", content: "Visible content" },
  { title: "Article B", access: "private", content: "Hidden content" }
]);
```

Contoh kueri

```
db.articles.aggregate([
  {
    $redact: {
      $cond: [
        { $eq: ["$access", "public"] },
        "$$KEEP",
        "$$PRUNE"
      ]
    }
  }
]);
```

Keluaran

```
[
  {
    "_id" : ObjectId("..."),
    "title" : "Article A",
    "access" : "public",
    "content" : "Visible content"
```

```
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan $$KEEP perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

Variabel $$PRUNE sistem digunakan dengan `$redact` tahapan dalam pipa agregasi untuk mengecualikan dokumen atau level dokumen tertanam dari hasil. Ketika suatu kondisi dievaluasi$ $PRUNE, dokumen atau subdokumen saat ini dihapus dari output. Ini biasanya digunakan dengan $ $DESCEND (untuk menyimpan dan melintasi dokumen) atau $$KEEP (untuk menyimpan dokumen di semua tingkatan).

Parameter

Tidak ada. Variabel $$PRUNE sistem digunakan tanpa parameter apa pun dan harus digunakan dengan`$redact`.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan cara menggunakan $$PRUNE dengan `$redact` untuk mengecualikan pengguna berusia di atas 30 tahun dari hasil.

Buat dokumen sampel

```
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" } }
])
```

Contoh kueri

```
db.users.aggregate([
  {
    $redact: {
      $cond: {
        if: { $gt: ["$age", 30] },
        then: "$$PRUNE",
        else: "$$DESCEND"
      }
    }
  }
])
```

Keluaran

```
[
  {
    "_id": 2,
    "name": "Saanvi Sarkar",
    "age": 28,
    "address": {
      "street": "456 Oak Rd",
      "city": "Someplace",
      "state": "NY"
    }
```

```
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan $$PRUNE perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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()
```

$ AKAR

$R00TOperator di Amazon DocumentDB digunakan untuk mereferensikan seluruh dokumen input dalam pipeline agregasi. Ini memungkinkan Anda untuk mengakses dan memanipulasi dokumen lengkap, termasuk semua bidang dan struktur bersarangnya.

Parameter

Tidak ada

Contoh (MongoDB Shell)

Contoh ini menunjukkan penggunaan $R00T untuk membuat log audit yang menangkap dokumen asli lengkap bersama dengan metadata tentang kapan diproses.

Buat dokumen sampel

```
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
  }
]);
```

Contoh kueri

```
db.orders.aggregate([
  {
    $project: {
      processedAt: new Date(),
      originalDocument: "$$ROOT",
      summary: {
        $concat: [
          "Order ",
          "$orderId",
          " for ",
          "$customer",
          " - Total: $",
          { $toString: "$totalAmount" }
        ]
      }
    }
  }
]);
```

Keluaran

```
[
  {
    _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'
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan $R00T perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$abs`Operator di Amazon DocumentDB mengembalikan nilai absolut dari suatu angka. Ini dapat digunakan dalam pipa agregasi untuk melakukan operasi matematika pada bidang numerik.

Parameter

- `number`: Ekspresi numerik yang nilai absolutnya akan dikembalikan.

Contoh (MongoDB Shell)

Contoh ini menunjukkan penggunaan `$abs` operator untuk menemukan nilai absolut dari bidang numerik.

Buat dokumen sampel

```
db.numbers.insertMany([
  { "_id": 1, "value": -5 },
  { "_id": 2, "value": 10 },
  { "_id": 3, "value": -3.14 },
  { "_id": 4, "value": 0 }
]);
```

Contoh kueri

```
db.numbers.aggregate([
  { $project: {
    "_id": 1,
    "absolute_value": { $abs: "$value" }
  }}
]);
```

Keluaran

```
[
  { "_id": 1, "absolute_value": 5 },
  { "_id": 2, "absolute_value": 10 },
  { "_id": 3, "absolute_value": 3.14 },
  { "_id": 4, "absolute_value": 0 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$abs` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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()
```

$add

$addOperator di Amazon DocumentDB digunakan untuk menambahkan angka atau tanggal bersama-sama. Ini dapat digunakan untuk melakukan operasi aritmatika pada bidang numerik atau untuk melakukan aritmatika tanggal dengan menambahkan sejumlah unit waktu ke bidang tanggal.

Parameter

- `expression1`: Nomor atau tanggal pertama untuk ditambahkan.
- `expression2`: Angka atau tanggal kedua untuk ditambahkan.
- `expression3`: (opsional) Nomor atau tanggal tambahan untuk ditambahkan.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$add` operator untuk menambahkan dua angka bersama-sama.

Buat dokumen sampel

```
db.numbers.insertMany([
  { _id: 1, a: 5, b: 3 },
  { _id: 2, a: 10, b: 7 },
  { _id: 3, a: 2, b: 8 }
]);
```

Contoh kueri

```
db.numbers.aggregate([
  { $project: {
    _id: 1,
    sum: { $add: ["$a", "$b"] }
  }}
])
```

Keluaran

```
[
  { "_id" : 1, "sum" : 8 },
  { "_id" : 2, "sum" : 17 },
  { "_id" : 3, "sum" : 10 }
]
```

Dalam contoh ini, `$add` operator digunakan untuk menambahkan nilai `a` dan `b` bidang untuk setiap dokumen dan menyimpan hasilnya di `sum` bidang.

Contoh kode

Untuk melihat contoh kode untuk menggunakan $add perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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()
```

$ tambahkan ToSet

Operator `$addToSet` agregasi mengembalikan array nilai unik dari ekspresi tertentu untuk setiap grup. Ini digunakan dalam `$group` tahap untuk mengakumulasi nilai yang berbeda, secara otomatis menghilangkan duplikat.

Parameter

- `expression`: Ekspresi untuk mengevaluasi setiap dokumen dalam grup.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$addToSet` operator untuk mengumpulkan kota-kota unik di mana pesanan ditempatkan untuk setiap pelanggan.

Buat dokumen sampel

```
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 }
]);
```

Contoh kueri

```
db.orders.aggregate([
  {
    $group: {
      _id: "$customer",
      cities: { $addToSet: "$city" }
    }
```

```
  }
]);
```

Keluaran

```
[
  { _id: 'Bob', cities: [ 'Seattle', 'Boston' ] },
  { _id: 'Alice', cities: [ 'Seattle', 'Portland' ] }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan operator `$addToSet` agregasi, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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([
    {
      $group: {
        _id: "$customer",
        cities: { $addToSet: "$city" }
      }
    }
  ]).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([
        {
            '$group': {
                '_id': '$customer',
                'cities': { '$addToSet': '$city' }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$addFields

`$addFields`Tahap dalam pipeline agregasi Amazon DocumentDB memungkinkan Anda menambahkan bidang terkomputasi baru ke dokumen. Ini dapat berguna untuk menambahkan data yang diturunkan atau diubah ke dokumen.

Parameter

- `newField`: Nama bidang baru untuk ditambahkan.
- `expression`: Ekspresi yang menyelesaikan nilai bidang baru.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$addFields` untuk menambahkan bidang baru `TotalInventory` yang menghitung total inventaris berdasarkan `Inventory.OnHand` dan `Inventory.OrderQnty` bidang.

Buat dokumen sampel

```
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
  }
]);
```

Contoh kueri

```
db.example.aggregate([
  {
    $addFields: {
      TotalInventory: { $add: ["$Inventory.OnHand", "$Inventory.OrderQnty"] }
    }
  }
])
```

Keluaran

```
[
  {
    "_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
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$addFields` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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()
```

$ semua ElementsTrue

Baru dari versi 4.0

`$allElementsTrue`Operator digunakan untuk memeriksa apakah semua elemen dalam array mengevaluasi ke nilai sebenarnya.

Parameter

- `expression`: Ekspresi yang mengevaluasi ke array.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$allElementsTrue` untuk memeriksa apakah semua elemen dalam array benar.

Buat dokumen sampel

```
db.collection.insert([
  { "name": "John", "scores": [100, 90, 80] },
  { "name": "Jane", "scores": [80, 85, 0] },
  { "name": "Bob", "scores": [90, 95, null] }
])
```

Contoh kueri

```
db.collection.find({
  "scores": { "$allElementsTrue": [{ "$gt": 0 }] }
})
```

Keluaran

```
[
  { "_id" : ObjectId("..."), "name" : "John", "scores" : [ 100, 90, 80 ] },
  { "_id" : ObjectId("..."), "name" : "Bob", "scores" : [ 90, 95, null ] }
]
```

Dalam contoh ini, query memeriksa apakah semua elemen dalam `scores` array lebih besar dari 0. Dokumen dengan `"name": "Jane"` dikecualikan karena `scores` array berisi 0, yang merupakan nilai palsu.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$allElementsTrue` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
```

```
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Operator `$and` agregasi mengevaluasi beberapa ekspresi dan mengembalikan `true` hanya jika semua ekspresi mengevaluasi. `true` Jika ada ekspresi`false`, ia kembali`false`.

Parameter

- `expressions`: Sebuah array ekspresi untuk mengevaluasi.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$and` operator untuk memeriksa apakah produk memenuhi beberapa kriteria.

Buat dokumen sampel

```
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 }
]);
```

Contoh kueri

```
db.products.aggregate([
  {
    $project: {
      name: 1,
      price: 1,
      inStock: 1,
      affordable: {
        $and: [
          { $lt: ["$price", 100] },
          { $eq: ["$inStock", true] }
        ]
      }
    }
  }
]);
```

Keluaran

```
[
  { _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 }
```

```
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan operator $and agregasi, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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()
```

$ apa saja ElementTrue

Baru dari versi 4.0

`$anyElementTrue`Operator digunakan untuk menentukan apakah ada elemen dalam array yang benar.

Parameter

- `field`: Bidang array untuk mengevaluasi.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$anyElementTrue` untuk memeriksa apakah ada elemen dalam array benar.

Buat dokumen sampel

```
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] }
])
```

Contoh kueri

```
db.grades.aggregate([
  {
    $project: {
      student: 1,
      isAnyTrue: { $anyElementTrue: ["$scores"] },
      _id: 0
    }
  }
])
```

Keluaran

```
[
  { student: 'Tim', isAnyTrue: false },
  { student: 'Bob', isAnyTrue: false },
  { student: 'Ivy', isAnyTrue: true }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$anyElementTrue` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');
```

```
async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('grades');

  const result = await collection.aggregate([
    {
      $project: {
        student: 1,
        isAnyTrue: { $anyElementTrue: ["$scores"] },
        _id: 0
      }
    }
  ]).toArray();

  console.log(result);

  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['grades']

    result = list(collection.aggregate([
      {
        "$project": {
          "student": 1,
          "isAnyTrue": { "$anyElementTrue": ["$scores"] },
          "_id": 0
        }
```

```
      }
    ]))

    print(result)

    client.close()

example()
```

$ array ElemAt

`$arrayElemAt`Operator di Amazon DocumentDB memungkinkan Anda untuk mengambil elemen dari array dengan posisi indeksnya. Ini sangat berguna ketika Anda perlu mengakses elemen tertentu dalam bidang array dalam dokumen Anda.

Parameter

- `array`: Array input dari mana elemen akan diambil.
- `index`: Posisi indeks berbasis nol dari elemen untuk mengambil. Nilai ini harus berupa bilangan bulat non-negatif.

Contoh (MongoDB Shell)

Dalam contoh ini, kita akan menunjukkan bagaimana menggunakan `$arrayElemAt` operator untuk mengambil elemen tertentu dari `flight_miles` array dalam `miles` koleksi.

Buat dokumen sampel

```
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 ]}
]);
```

Contoh kueri

```
db.miles.aggregate([
```

```
  { $project: {
    "_id": 1,
    "first_mile": { $arrayElemAt: [ "$flight_miles", 0 ] },
    "last_mile": { $arrayElemAt: [ "$flight_miles", -1 ] }
  }}
]);
```

Keluaran

```
{ "_id" : 1, "first_mile" : 1205, "last_mile" : 880 }
{ "_id" : 2, "first_mile" : 1205, "last_mile" : 2780 }
{ "_id" : 3, "first_mile" : 1205, "last_mile" : 880 }
```

Dalam contoh ini, kita menggunakan `$arrayElemAt` operator untuk mengambil elemen pertama dan terakhir dari `flight_miles` array untuk setiap dokumen.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$arrayElemAt` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('miles');

  const result = await collection.aggregate([
    { $project: {
      "_id": 1,
      "first_mile": { $arrayElemAt: [ "$flight_miles", 0 ] },
      "last_mile": { $arrayElemAt: [ "$flight_miles", -1 ] }
    }}
  ]).toArray();

  console.log(result);
  client.close();
```

```
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.mydatabase
    collection = db.miles

    result = list(collection.aggregate([
        { "$project": {
            "_id": 1,
            "first_mile": { "$arrayElemAt": [ "$flight_miles", 0 ] },
            "last_mile": { "$arrayElemAt": [ "$flight_miles", -1 ] }
        }}
    ]))

    print(result)
    client.close()

example()
```

$ array ToObject

`$arrayToObject`Operator di Amazon DocumentDB adalah kebalikan dari operator. `$objectToArray` Dibutuhkan array dokumen pasangan kunci-nilai dan mengubahnya menjadi satu dokumen. Ini sangat berguna ketika Anda perlu mengonversi array pasangan kunci-nilai kembali ke objek atau struktur dokumen.

Parameter

- `array expression`: Ekspresi yang menyelesaikan ke array. Elemen array harus dokumen dengan dua bidang: k (kunci) dan v (nilai).

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$arrayToObject` untuk mengkonversi array pasangan kunci-nilai kembali ke dokumen.

Buat dokumen sampel

```
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 } }
]);
```

Contoh kueri

```
db.videos.aggregate([
  { $project: { name: 1, videos: { $objectToArray: "$inventory" } } },
  { $project: { name: 1, inventory: { $arrayToObject: "$videos" } } }
]);
```

Keluaran

```
{ "_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 } }
```

Dalam contoh ini, `$objectToArray` operator digunakan untuk mengubah `inventory` objek menjadi array pasangan kunci-nilai. `$arrayToObject`Operator kemudian digunakan untuk mengubah array kembali menjadi dokumen, memulihkan struktur objek asli.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$arrayToObject` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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()
```

$avg

Operator `$avg` agregasi di Amazon DocumentDB menghitung nilai rata-rata ekspresi yang ditentukan di seluruh dokumen yang dimasukkan ke panggung. Operator ini berguna untuk menghitung rata-rata bidang numerik atau ekspresi di satu set dokumen.

Parameter

- `expression`: Ekspresi yang digunakan untuk menghitung rata-rata. Ini bisa berupa jalur bidang (misalnya`"$field"`) atau ekspresi (misalnya`{ $multiply: ["$field1", "$field2"] }`).

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$avg` operator untuk menghitung skor rata-rata di satu set dokumen siswa.

Buat dokumen sampel

```
db.students.insertMany([
  { name: "John", score: 85 },
  { name: "Jane", score: 92 },
  { name: "Bob", score: 78 },
  { name: "Alice", score: 90 }
]);
```

Contoh kueri

```
db.students.aggregate([
  { $group: {
    _id: null,
    avgScore: { $avg: "$score" }
  }}
]);
```

Keluaran

```
[
  {
    "_id": null,
    "avgScore": 86.25
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$avg` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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');
  const result = await db.collection('students').aggregate([
    { $group: {
      _id: null,
      avgScore: { $avg: '$score' }
    }}
  ]).toArray();
  console.log(result);
  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
```

```
    result = list(db.students.aggregate([
        { '$group': {
            '_id': None,
            'avgScore': { '$avg': '$score' }
        }}
    ]))
    print(result)
    client.close()

calculate_avg_score()
```

$bucket

Baru dari versi 8.0

Tidak didukung oleh cluster elastis.

Tahap `$bucket` agregasi di Amazon DocumentDB memungkinkan Anda mengelompokkan dokumen masukan ke dalam bucket berdasarkan ekspresi dan batas bucket yang ditentukan. Ini dapat berguna untuk menganalisis data yang termasuk dalam rentang nilai atau kategori tertentu.

Parameter

- `groupBy`(required): Ekspresi yang menentukan nilai untuk dikelompokkan menurut.
- `boundaries`(required): Array nilai ganda yang menentukan batas bucket. Dokumen ditugaskan ke ember berdasarkan nilai `groupBy` ekspresi yang termasuk dalam batas-batas yang ditentukan.
- `default`(opsional): Nilai literal yang merupakan output untuk dokumen yang nilai `groupBy` ekspresinya tidak termasuk dalam batas yang ditentukan.
- `output`(opsional): Objek yang menentukan informasi yang akan dikeluarkan untuk setiap bucket. Anda dapat menggunakan operator akumulator seperti`$sum`,, `$avg``$min`, dan `$max` untuk menghitung agregasi untuk setiap bucket.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$bucket` panggung untuk mengelompokkan data penjualan berdasarkan kisaran harga.

Buat dokumen sampel

```
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") }
]);
```

Contoh kueri

```
db.sales.aggregate([
  {
    $bucket: {
      groupBy: "$price",
      boundaries: [0, 10, 20, 30],
      default: "Other",
      output: {
        "count": { $sum: 1 },
        "totalQuantity": { $sum: "$quantity" }
      }
    }
  },
  {
    $sort: { _id: 1 }
  }
])
```

Keluaran

```
[
  { _id: 0, count: 1, totalQuantity: 3 },
  { _id: 10, count: 2, totalQuantity: 4 },
  { _id: 20, count: 2, totalQuantity: 2 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$bucket` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('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()
```

$ceil

Baru dari versi 4.0

`$ceil`Operator di Amazon DocumentDB, seperti di MongoDB, membulatkan angka ke bilangan bulat terdekat. Ini berguna ketika Anda perlu melakukan operasi matematika pada bidang numerik dan memastikan hasilnya adalah bilangan bulat.

Parameter

- `expression`: Ekspresi numerik untuk dibulatkan.

Contoh (MongoDB Shell)

Contoh ini menunjukkan bagaimana menggunakan `$ceil` operator untuk mengumpulkan bidang numerik.

Buat dokumen sampel

```
db.numbers.insertMany([
  { "_id": 1, "value": 3.14 },
  { "_id": 2, "value": -2.7 },
  { "_id": 3, "value": 0 }
])
```

Contoh kueri

```
db.numbers.aggregate([
  { $project: {
    "roundedUp": { $ceil: "$value" }
  }}
])
```

Keluaran

```
{ "_id": 1, "roundedUp": 4 }
{ "_id": 2, "roundedUp": -2 }
{ "_id": 3, "roundedUp": 0 }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$ceil` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Tidak didukung oleh cluster elastis.

Tahap `$changeStream` agregasi membuka kursor aliran perubahan untuk memantau perubahan real-time ke koleksi. Ia mengembalikan perubahan dokumen peristiwa ketika menyisipkan, memperbarui, mengganti, atau menghapus operasi terjadi.

Parameter

- `fullDocument`: Menentukan apakah akan mengembalikan dokumen lengkap untuk operasi update. Pilihannya adalah `default` atau`updateLookup`.

- `resumeAfter`: Opsional. Lanjutkan token untuk melanjutkan dari titik tertentu dalam aliran perubahan.
- `startAtOperationTime`: Opsional. Stempel waktu untuk memulai aliran perubahan dari.
- `allChangesForCluster`: Opsional. Nilai Boolean. Kapan`true`, perhatikan semua perubahan di seluruh cluster (untuk database admin). Ketika `false` (default), hanya menonton koleksi yang ditentukan.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan menggunakan `$changeStream` panggung untuk memantau perubahan koleksi.

Contoh kueri

```
// 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()));
}
```

Keluaran

```
{
  _id: { _data: '...' },
  operationType: 'insert',
  clusterTime: Timestamp(1, 1234567890),
  fullDocument: { _id: 1, item: 'Widget', qty: 10 },
  ns: { db: 'test', coll: 'inventory' },
  documentKey: { _id: 1 }
}
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan tahap `$changeStream` agregasi, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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()
```

$cmp

`$cmp`Operator di Amazon DocumentDB digunakan untuk membandingkan dua nilai dan mengembalikan nilai integer yang menunjukkan urutan relatifnya. Ini adalah operator perbandingan yang membandingkan dua ekspresi dan mengembalikan nilai integer -1, 0, atau 1, tergantung pada apakah nilai pertama kurang dari, sama dengan, atau lebih besar dari nilai kedua, masing-masing.

Parameter

- `expression1`: Ekspresi pertama untuk membandingkan.
- `expression2`: Ekspresi kedua untuk membandingkan.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$cmp` operator untuk membandingkan dua nilai numerik.

Buat dokumen sampel

```
db.collection.insertMany([
  { _id: 1, value1: 10, value2: 20 },
  { _id: 2, value1: 15, value2: 15 },
  { _id: 3, value1: 20, value2: 10 }
]);
```

Contoh kueri

```
db.collection.find({
  $expr: {
    $cmp: ["$value1", "$value2"]
  }
})
```

Keluaran

```
[
  { "_id" : 1, "value1" : 10, "value2" : 20 },
  { "_id" : 3, "value1" : 20, "value2" : 10 }
]
```

Dalam contoh ini, `$cmp` operator membandingkan `value2` bidang `value1` dan untuk setiap dokumen. Hasilnya adalah:

```
- `$cmp: ["$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).
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$cmp` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

Berikut adalah contoh penggunaan `$cmp` operator dalam Node.js aplikasi dengan `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

Berikut adalah contoh penggunaan `$cmp` operator dalam aplikasi Python dengan driver: `pymongo`

```
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()
```

Output dari kedua contoh Node.js dan Python akan sama dengan contoh MongoDB Shell:

```
[
  { "_id" : 1, "value1" : 10, "value2" : 20 },
  { "_id" : 2, "value1" : 15, "value2" : 15 },
  { "_id" : 3, "value1" : 20, "value2" : 10 }
]
```

$collStats

Baru dari versi 4.0

Tahap `$collStats` agregasi di Amazon DocumentDB menyediakan statistik tentang koleksi yang ditentukan, mirip dengan perintah di shell `db.collection.stats()` MongoDB. Tahap ini dapat digunakan untuk mengambil informasi tentang koleksi, seperti jumlah dokumen, ukuran total koleksi, dan berbagai metrik kinerja.

Parameter

- `latencyStats`: (opsional) Dokumen yang menentukan opsi untuk mengumpulkan statistik latensi. Parameter ini tidak didukung di Amazon DocumentDB.
- `recordStats`: (opsional) Dokumen yang menentukan opsi untuk mengumpulkan statistik catatan. Parameter ini tidak didukung di Amazon DocumentDB.

- `queryExecStats`: (opsional) Sebuah dokumen yang menentukan pilihan untuk mengumpulkan statistik eksekusi query. Parameter ini tidak didukung di Amazon DocumentDB.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan tahap `$collStats` agregasi untuk mengambil statistik tentang koleksi bernama `test` dalam database. db

Buat dokumen sampel

```
db.test.insertMany([
  { "name": "John", "age": 30 },
  { "name": "Jane", "age": 25 },
  { "name": "Bob", "age": 35 }
]);
```

Contoh kueri

```
db.test.aggregate([
  { $collStats: {} }
]);
```

Keluaran

```
{
  "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)
}
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$collStats` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

Berikut adalah contoh cara menggunakan tahap `$collStats` agregasi dalam Node.js aplikasi menggunakan driver MongoDB Node.js resmi:

```
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

Berikut adalah contoh cara menggunakan tahap `$collStats` agregasi dalam aplikasi Python menggunakan 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

Operator `$concat` agregasi di Amazon DocumentDB menggabungkan (atau menggabungkan) beberapa string dalam dokumen untuk menghasilkan string tunggal yang dapat dikembalikan ke aplikasi. Ini mengurangi pekerjaan yang dilakukan dalam aplikasi, karena manipulasi string dilakukan di tingkat database.

Parameter

- `expression1`: String pertama yang digabungkan.
- `expression2`: String kedua untuk digabungkan.
- `...`: String tambahan untuk digabungkan (opsional).

Contoh (MongoDB Shell)

Dalam contoh ini, kami menggabungkan nama depan dan belakang pengguna untuk menghasilkan nama lengkap setiap orang.

Buat dokumen sampel

```
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"}
])
```

Contoh kueri

```
db.people.aggregate([
  { $project: { full_name: { $concat: [ "$first_name", " ", "$last_name"] } } }
])
```

Keluaran

```
{ "_id" : 1, "full_name" : "Jane Doe" }
{ "_id" : 2, "full_name" : "John Doe" }
{ "_id" : 3, "full_name" : "Steve Smith" }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$concat` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

Operator `$concatArrays` agregasi di Amazon DocumentDB digunakan untuk menggabungkan dua atau lebih array menjadi satu array. Ini dapat berguna ketika Anda perlu menggabungkan beberapa array data ke dalam satu array untuk pemrosesan atau analisis lebih lanjut.

Parameter

- `array1`: Array pertama yang akan digabungkan.

- `array2`: Array kedua yang akan digabungkan.
- `[array3, ...]`: (opsional) Array tambahan yang akan digabungkan.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$concatArrays` operator untuk menggabungkan dua array ke dalam array tunggal.

Buat dokumen sampel

```
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"]
  }
]);
```

Contoh kueri

```
db.collection.aggregate([
  {
    $project: {
      _id: 0,
      name: 1,
      all_activities: { $concatArrays: ["$hobbies", "$skills"] }
    }
  }
]);
```

Keluaran

```
[
  {
```

```
    "name": "John Doe",
    "all_activities": [
      "reading",
      "swimming",
      "programming",
      "design"
    ]
  },
  {
    "name": "Jane Smith",
    "all_activities": [
      "hiking",
      "cooking",
      "marketing",
      "analysis"
    ]
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$concatArrays` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await 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

`$cond`Operator di Amazon DocumentDB digunakan untuk mengevaluasi ekspresi bersyarat dan mengembalikan salah satu dari dua ekspresi hasil yang mungkin.

Parameter

- `if`: Ekspresi boolean untuk mengevaluasi.

- `then`: Ekspresi untuk kembali jika `if` ekspresi benar.
- `else`: Ekspresi untuk kembali jika `if` ekspresi salah.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$cond` operator untuk mengembalikan nilai berdasarkan usia seseorang.

Buat dokumen sampel

```
db.people.insertMany([
  { _id: 1, name: "John Doe", age: 35 },
  { _id: 2, name: "Jane Doe", age: 25 },
  { _id: 3, name: "Bob Smith", age: 65 }
]);
```

Contoh kueri

```
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"
            }
          }
        }
      }
    }
  }
])
```

Keluaran

```
[
  { "_id" : 1, "name" : "John Doe", "ageGroup" : "middle-aged" },
  { "_id" : 2, "name" : "Jane Doe", "ageGroup" : "young" },
  { "_id" : 3, "name" : "Bob Smith", "ageGroup" : "elderly" }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan $cond perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Baru dari versi 4.0

`$convert`Operator di Amazon DocumentDB digunakan untuk mengonversi nilai dari satu tipe data ke tipe data lainnya. Operator ini berguna ketika Anda perlu melakukan operasi pada data dari berbagai jenis, seperti mengubah string ke angka atau tanggal ke stempel waktu.

Parameter

- `to`: Tipe data target untuk mengonversi nilainya menjadi. Nilai yang didukung adalah `"string"`, `"double"`, `"long"`, `"int"`, `"date"`, dan `"boolean"`.
- `from`: Tipe data saat ini dari nilai. Jika tidak ditentukan, Amazon DocumentDB akan mencoba mendeteksi tipe data secara otomatis.
- `onError`: (opsional) Nilai yang akan dikembalikan jika konversi gagal. Dapat berupa nilai tertentu atau salah satu dari nilai khusus berikut:`"null"`,`"zerofill"`, atau`"error"`.
- `onNull`: (opsional) Nilai untuk kembali jika nilai masukan`null`. Dapat berupa nilai tertentu atau salah satu dari nilai khusus berikut:`"null"`,`"zerofill"`, atau`"error"`.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan konversi nilai string ke tanggal menggunakan operator. `$convert`

Buat dokumen sampel

```
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" }
]);
```

Contoh kueri

```
db.users.aggregate([
  {
    $project: {
      _id: 1,
      name: 1,
      joinedOn: {
```

```
        $convert: {
          input: "$joinedOn",
          to: "date",
          onError: "null",
          onNull: "null"
        }
      }
    }
  }
])
```

Keluaran

```
[
  { "_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 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$convert` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require("mongodb");

async function example() {
  const client = await MongoClient.connect("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()
```

$count

Tahap `$count` agregasi di Amazon DocumentDB digunakan untuk menghitung jumlah dokumen yang masuk ke panggung. Ini sering digunakan sebagai tahap akhir dalam pipa agregasi untuk mengembalikan jumlah total dokumen yang cocok dengan tahap sebelumnya.

Parameter

- `field`: Bidang untuk dihitung. Parameter ini opsional, dan jika tidak disediakan, tahap akan menghitung jumlah total dokumen input.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$count` panggung untuk mendapatkan jumlah total dokumen dalam koleksi.

Buat dokumen sampel

```
db.users.insertMany([
  { name: "John", age: 30 },
  { name: "Jane", age: 25 },
  { name: "Bob", age: 35 },
  { name: "Alice", age: 28 }
]);
```

Contoh kueri

```
db.users.aggregate([
  { $count: "total" }
]);
```

Keluaran

```
{ "total" : 4 }
```

Contoh agregat `users` koleksi dan menggunakan `$count` panggung untuk menghitung jumlah total dokumen.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$count` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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()
```

$currentOp

Tahap `$currentOp` agregasi mengembalikan informasi tentang operasi yang sedang berjalan di database. Tahap ini berguna untuk memantau kueri dan operasi aktif dalam jalur agregasi.

Parameter

- `allUsers`(opsional): Ketika diatur ke`true`, mengembalikan operasi untuk semua pengguna. Default-nya adalah `false`.
- `idleConnections`(opsional): Ketika diatur ke`true`, termasuk koneksi idle. Default-nya adalah `false`.
- `idleCursors`(opsional): Ketika diatur ke`true`, termasuk informasi tentang kursor siaga. Default-nya adalah `false`.
- `idleSessions`(opsional): Ketika diatur ke`true`, termasuk informasi tentang sesi idle. Default-nya adalah `true`.
- `localOps`(opsional): Ketika diatur ke`true`, termasuk operasi lokal. Default-nya adalah `false`.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan menggunakan tahap `$currentOp` agregasi untuk mengambil informasi tentang operasi baca aktif.

Contoh kueri

```
db.aggregate([
  { $currentOp: { allUsers: true, idleConnections: false } },
  { $match: { op: "query" } }
```

```
])
```

Keluaran

```
[
  {
    "opid": "12345",
    "active": true,
    "op": "query",
    "ns": "test.users",
    "secs_running": 2
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$currentOp` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await 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

Baru dari versi 5.0

Operator `$dateAdd` agregasi di Amazon DocumentDB memungkinkan Anda menambahkan durasi ke nilai tanggal dan waktu.

Parameter

- `date`: Nilai tanggal dan waktu untuk menambahkan durasi.
- `duration`: Durasi untuk menambah `date` nilai. Ini dapat ditentukan sebagai objek dengan kunci untuk`years`,`months`,`weeks`, `days`,`hours`,`minutes`, dan`seconds`.
- `timezone`: (opsional) Zona waktu yang akan digunakan saat melakukan penambahan tanggal. Jika tidak ditentukan, zona waktu default cluster Amazon DocumentDB digunakan.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$dateAdd` operator untuk menambahkan 2 hari dan 12 jam ke tanggal.

Buat dokumen sampel

```
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") }
]);
```

Contoh kueri

```
db.events.aggregate([
  {
    $project: {
      _id: 1,
      eventDate: 1,
      eventDatePlustwodaysandtwelvehours: {
        $dateAdd: {
          startDate: {
            $dateAdd: {
              startDate: "$eventDate",
              unit: "day",
              amount: 2
            }
          },
          unit: "hour",
          amount: 12
        }
      }
    }
  }
])
```

Keluaran

```
[
  {
    "_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)"
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan $dateAdd perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Baru dari versi 5.0

Tidak didukung oleh cluster elastis.

Operator `$dateDiff` agregasi menghitung perbedaan antara dua tanggal dalam satuan tertentu. Ini mengembalikan jumlah batas unit yang dilintasi antara tanggal mulai dan akhir.

Parameter

- `startDate`: Ekspresi tanggal awal.
- `endDate`: Ekspresi tanggal akhir.
- `unit`: Satuan waktu untuk perbedaan. Unit yang didukung adalah `year`,`quarter`,`month`,`week`,`day`,`hour`,`minute`,`second`, dan`millisecond`.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$dateDiff` operator untuk menghitung jumlah hari antara penempatan pesanan dan pengiriman.

Buat dokumen sampel

```
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")
  }
]);
```

Contoh kueri

```
db.shipments.aggregate([
  {
    $project: {
      orderId: 1,
      orderDate: 1,
      deliveryDate: 1,
      daysToDeliver: {
        $dateDiff: {
          startDate: "$orderDate",
          endDate: "$deliveryDate",
          unit: "day"
        }
      }
    }
  }
]);
```

Keluaran

```
[
  {
    _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
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$dateDiff` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('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()
```

$ tanggal FromString

Operator `$dateFromString` agregasi di Amazon DocumentDB memungkinkan Anda mengonversi string tanggal-waktu menjadi objek tanggal. Ini berguna ketika data Anda disimpan sebagai string tetapi Anda perlu melakukan operasi berbasis tanggal pada data.

Parameter

- `dateString`: String yang mewakili tanggal dan waktu.
- `format`: (opsional) Sebuah string yang menentukan format. `dateString` Jika tidak disediakan, Amazon DocumentDB akan mencoba mengurai string dalam format. ISO-8601

- `timezone`: (opsional) Sebuah string yang menentukan zona waktu. Jika tidak disediakan, Amazon DocumentDB akan menggunakan zona waktu server.
- `onError`: (opsional) Menentukan tindakan yang akan diambil jika konversi gagal. Nilai yang mungkin adalah `'error'` (default, yang melempar kesalahan), `'null'` (mengembalikan`null`), atau `'replace'` (menggantikan nilai dengan string pengganti yang ditentukan dalam `onErrorMessage` opsi).
- `onErrorMessage`: (opsional) Jika `onError` diatur ke`'replace'`, opsi ini menentukan string pengganti.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan untuk mengkonversi string tanggal `$dateFromString` ke objek tanggal di Amazon DocumentDB.

Buat dokumen sampel

```
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"}
]);
```

Contoh kueri

```
db.missionLog.aggregate([
  {
    $project: {
      event: '$event',
      logDate: {
        $dateFromString: {
          dateString: '$logDate'
        }
      }
    }
  }
]);
```

Keluaran

```
[
  {
    "_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")
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$dateFromString` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Baru dari versi 5.0

Operator `$dateSubtract` agregasi di Amazon DocumentDB memungkinkan Anda mengurangi durasi tertentu dari nilai tanggal.

Parameter

- `date`: Ekspresi tanggal yang menyelesaikan tanggal atau stempel waktu.
- `subtrahend`: Ekspresi durasi yang menentukan jumlah waktu untuk mengurangi dari ekspresi. `date`
- `unit`: Sebuah string yang menentukan satuan waktu untuk `subtrahend` ekspresi. Unit yang didukung adalah “tahun”, “kuartal”, “bulan”, “minggu”, “hari”, “jam”, “menit”, “kedua”, dan “milidetik”.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan cara menggunakan `$dateSubtract` operator untuk menghitung tanggal satu tahun yang lalu dari tanggal saat ini.

Buat dokumen sampel

```
db.events.insertOne({
  eventName: "Player joined",
  eventTime: ISODate("2023-04-01T12:00:00Z")
});
```

Contoh kueri

```
db.events.aggregate([
  {
    $project: {
      eventName: 1,
      oneYearAgo: {
```

```
        $dateSubtract: {
          startDate: "$eventTime",
          amount: 1,
          unit: "year"
        }
      }
    }
  }
])
```

Keluaran

```
{
  "_id" : ObjectId("64567890abcdef012345678"),
  "eventName" : "Player joined",
  "oneYearAgo" : ISODate("2022-04-01T12:00:00Z")
}
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$dateSubtract` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('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()
```

$ tanggal ToString

Operator `$dateToString` agregasi di Amazon DocumentDB digunakan untuk mengonversi nilai tanggal atau stempel waktu menjadi representasi string. Ini berguna ketika Anda perlu memformat tanggal dan waktu dengan cara tertentu untuk tampilan atau pemrosesan lebih lanjut.

Parameter

- `date`: Nilai tanggal atau stempel waktu yang akan dikonversi ke string.
- `format`: Sebuah string yang menentukan format di mana tanggal harus diwakili. String format dapat mencakup berbagai penentu format, seperti `%Y` untuk tahun empat digit, `%m` untuk bulan dua digit, `%d` untuk hari dua digit dalam sebulan, dll.
- `timezone`: (opsional) Zona waktu yang akan digunakan untuk konversi. Jika tidak ditentukan, zona waktu server yang menghosting cluster Amazon DocumentDB digunakan.
- `onNull`: (opsional) Nilai yang akan dikembalikan jika `date` parameternya`null`.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$dateToString` operator untuk memformat `logDate` bidang `missionLog` koleksi.

Buat dokumen sampel

```
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") }
]);
```

Contoh kueri

```
db.missionLog.aggregate([
  {
    $project: {
      event: "$event",
      logDateFormatted: {
```

```
        $dateToString: {
          format: "%Y-%m-%d %H:%M:%S",
          date: "$logDate"
        }
      }
    }
  }
])
```

Keluaran

```
[
  {
    "_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"
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$dateToString` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

Berikut adalah contoh penggunaan `$dateToString` operator dalam Node.js aplikasi:

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Berikut adalah contoh penggunaan `$dateToString` operator dalam aplikasi Python:

```
from pymongo import MongoClient

def example():
    client = 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

Baru dari versi 8.0

Tidak didukung oleh cluster elastis.

Operator `$dateTrunc` agregasi di Amazon DocumentDB memotong tanggal ke unit tertentu.

Parameter

- `date`: Ekspresi tanggal yang menyelesaikan tanggal atau stempel waktu.
- `unit`: Sebuah string yang menentukan satuan waktu untuk ekspresi subtrahend. Unit yang didukung adalah `year`,`quarter`,`month`,,`week`,`day`,`hour`,`minute`,`second`, dan`millisecond`.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan cara menggunakan `$dateTrunc` operator untuk memotong tanggal ke jam.

Buat dokumen sampel

```
db.events.insertMany([
  {
    eventName: "Event 1",
    eventTime: ISODate("2025-04-01T12:15:00Z")
  },
  {
    eventName: "Event 2",
    eventTime: ISODate("2025-08-15T14:33:22Z")
  },
]);
```

Contoh kueri

```
db.events.aggregate([
  {
    $project: {
      eventName: 1,
      eventTime: 1,
      truncatedToHour: {
        $dateTrunc: {
          date: "$eventTime",
          unit: "hour"
        }
      }
    }
  }
]);
```

Keluaran

```
[
  {
    _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')
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$dateTrunc` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('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()
```

$ hari OfMonth

Operator `$dayOfMonth` agregasi di Amazon DocumentDB mengambil hari dalam sebulan (dari 1 hingga 31) untuk tanggal tertentu. Operator ini berguna untuk mengelompokkan, memfilter, atau mengekstrak hari dalam sebulan dari bidang tanggal dalam dokumen Anda.

Parameter

- `date expression`: Ekspresi tanggal dapat berupa bidang tanggal dari dokumen, objek tanggal, atau string tanggal.

Contoh (MongoDB Shell)

Contoh ini menunjukkan cara menggunakan `$dayOfMonth` operator untuk mengekstrak hari dalam sebulan dari bidang tanggal dalam dokumen.

Buat dokumen sampel

```
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") }
]);
```

Contoh kueri

```
db.events.aggregate([
  { $project: {
    eventDay: { $dayOfMonth: "$eventDate" }
  }}
])
```

Keluaran

```
[
  { "_id" : 1, "eventDay" : 15 },
  { "_id" : 2, "eventDay" : 28 },
  { "_id" : 3, "eventDay" : 1 },
  { "_id" : 4, "eventDay" : 30 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$dayOfMonth` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');
```

```
async function example() {
  const client = await MongoClient.connect('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()
```

$ hari OfWeek

`$dayOfWeek`Operator di Amazon DocumentDB mengekstrak hari dalam seminggu dari bidang tanggal tertentu. Ini mengembalikan hari dalam seminggu sebagai angka antara 1 (Minggu) dan 7 (Sabtu), yang merupakan perilaku yang sama seperti di MongoDB.

Parameter

- `date field`: Bidang tanggal untuk mengekstrak hari dalam seminggu dari.

Contoh (MongoDB Shell)

Contoh ini menunjukkan cara menggunakan `$dayOfWeek` operator untuk mengekstrak hari dalam seminggu dari `date` bidang dalam `weather` koleksi.

Buat dokumen sampel

```
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")
  }
]);
```

Contoh kueri

```
db.weather.aggregate([
  {
    $project: {
      dayOfWeek: { $dayOfWeek: "$date" }
```

```
    }
  }
]).pretty();
```

Keluaran

```
{ "_id" : ObjectId("64272c6663f4f8ce422c2d91"), "dayOfWeek" : 7 }
{ "_id" : ObjectId("64272c6663f4f8ce422c2d92"), "dayOfWeek" : 1 }
{ "_id" : ObjectId("64272c6663f4f8ce422c2d93"), "dayOfWeek" : 2 }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$dayOfWeek` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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()
```

$ hari OfYear

`$dayOfYear`Operator di Amazon DocumentDB mengembalikan hari dalam setahun untuk tanggal sebagai angka antara 1 dan 366 (365 dalam tahun non-kabisat).

Parameter

- `expression`: Bidang tanggal atau ekspresi dari mana untuk mengekstrak hari dalam setahun.

Contoh (MongoDB Shell)

Contoh ini menunjukkan cara menggunakan `$dayOfYear` operator untuk mengekstrak hari dalam setahun dari bidang tanggal dalam koleksi Amazon DocumentDB.

Buat dokumen sampel

```
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")
  }
])
```

Contoh kueri

```
db.weather.aggregate([
  {
    $project: {
      dayOfYear: { $dayOfYear: "$date" }
    }
  }
]).pretty()
```

Keluaran

```
{ "_id" : ObjectId("642b86fc7d8e07af279bbe63"), "dayOfYear" : 74 }
{ "_id" : ObjectId("642b86fc7d8e07af279bbe64"), "dayOfYear" : 365 }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$dayOfYear` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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()
```

$divide

`$divide`Operator di jalur agregasi Amazon DocumentDB digunakan untuk membagi satu nomor dengan yang lain. Ini adalah operator yang berguna untuk melakukan operasi matematika pada bidang numerik dalam dokumen Anda.

Parameter

- `numerator`: Dividen atau nomor yang akan dibagi.
- `denominator`: Pembagi atau nomor untuk dibagi dengan.

Contoh (MongoDB Shell)

Contoh ini menunjukkan cara menggunakan `$divide` operator untuk menghitung tarif per jam untuk karyawan berdasarkan gaji tahunan mereka dan jumlah jam kerja per tahun.

Buat dokumen sampel

```
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 }
]);
```

Contoh kueri

```
db.employees.aggregate([
  {
    $project: {
      name: 1,
      hourlyRate: { $divide: ["$salary", "$hoursPerYear"] }
    }
  }
])
```

Keluaran

```
[
  { "_id" : 1, "name" : "John Doe", "hourlyRate" : 38.46153846153846 },
  { "_id" : 2, "name" : "Jane Smith", "hourlyRate" : 43.26923076923077 },
```

```
  { "_id" : 3, "name" : "Bob Johnson", "hourlyRate" : 36.05769230769231 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$divide` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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()
```

$eq

Operator `$eq` agregasi membandingkan dua nilai dan kembali `true` jika mereka sama, jika tidak kembali. `false`

Parameter

- `expression1`: Nilai pertama untuk membandingkan.
- `expression2`: Nilai kedua untuk membandingkan.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan menggunakan `$eq` operator untuk memeriksa apakah jumlah produk cocok dengan nilai target.

Buat dokumen sampel

```
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 }
]);
```

Contoh kueri

```
db.inventory.aggregate([
  {
    $project: {
      item: 1,
      qty: 1,
      target: 1,
      meetsTarget: { $eq: ["$qty", "$target"] }
    }
  }
]);
```

Keluaran

```
[
  { _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 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan operator $eq agregasi, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Baru dari versi 4.0

`$exp`Operator di Amazon DocumentDB memungkinkan Anda menaikkan konstanta e ke nomor tertentu.

Parameter

- `expression`: Ekspresi untuk mengevaluasi. Ini dapat berupa ekspresi agregasi yang valid, termasuk referensi bidang, operasi aritmatika, dan tahap agregasi lainnya.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$exp` operator untuk menemukan semua dokumen di mana `quantity` bidang lebih besar dari `price` bidang.

Buat dokumen sampel

```
db.items.insertMany([
  { item: "canvas", quantity: 4 },
  { item: "journal", quantity: 2 }
]);
```

Contoh kueri

```
db.items.aggregate([
  { $project: {
    "quantityRaised": {$exp: "$quantity"}}
  }
]);
```

Keluaran

```
[
  {
    _id: ObjectId('6920b785311cf98b79d2950d'),
    quantityRaised: 54.598150033144236
  },
  {
    _id: ObjectId('6920b785311cf98b79d2950e'),
    quantityRaised: 7.38905609893065
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$exp` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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()
```

$filter

`$filter`Operator di Amazon DocumentDB digunakan untuk menerapkan ekspresi filter ke setiap elemen array dan mengembalikan array yang hanya berisi elemen yang cocok dengan kondisi yang ditentukan. Operator ini berguna ketika Anda perlu melakukan operasi penyaringan kompleks pada bidang array dalam dokumen Anda.

Parameter

- `input`: Bidang array untuk menyaring.
- `as`: Nama variabel yang digunakan untuk setiap elemen `input` array dalam `cond` ekspresi.
- `cond`: Ekspresi boolean yang menentukan apakah elemen tertentu harus disertakan dalam array output.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan cara menggunakan `$filter` operator untuk memproyeksikan pelanggan setiap pesanan dan membuat bidang array baru PaidItem berisi hanya item dari array item di mana harga lebih besar dari 15. Pada dasarnya, ini menyaring item setiap pesanan untuk memasukkan hanya produk yang harganya lebih dari 15.

Buat dokumen sampel

```
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 }
  ]}
]);
```

Contoh kueri

```
db.orders.aggregate([
  {
    $project: {
      customer: 1,
      paidItems: {
        $filter: {
          input: "$items",
          as: "item",
          cond: { $gt: ["$$item.price", 15] }
        }
      }
    }
  }
]).pretty();
```

Keluaran

```
[
  {
    _id: 1,
    customer: 'abc123',
    paidItems: [ { name: 'Product B', price: 20, qty: 1 } ]
  },
  { _id: 2, customer: 'def456', paidItems: [] },
  { _id: 3, customer: 'ghi789', paidItems: [] }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$filter` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

Baru dari versi 5.0.

Tidak didukung oleh cluster elastis.

`$first`Operator di Amazon DocumentDB mengembalikan dokumen pertama dari kumpulan dokumen yang dikelompokkan. Hal ini biasanya digunakan dalam pipeline agregasi untuk mengambil dokumen pertama yang cocok dengan kondisi tertentu.

Parameter

- `expression`: Ekspresi untuk kembali sebagai nilai pertama di setiap grup.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$first` operator untuk mengambil nilai item pertama yang ditemui untuk setiap kategori selama agregasi.

Catatan: `$first` mengembalikan dokumen pertama berdasarkan urutan dokumen saat ini dalam pipeline. Untuk memastikan pesanan tertentu (misalnya, berdasarkan tanggal, harga, dll.), `$sort` Tahap harus digunakan sebelum `$group` tahap.

Buat dokumen sampel

```
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" }
]);
```

Contoh kueri

```
db.products.aggregate([
  { $group: { _id: "$category", firstItem: { $first: "$item" } } }
]);
```

Keluaran

```
[
  { "_id" : "food", "firstItem" : "abc" },
  { "_id" : "toy", "firstItem" : "xyz" }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$first` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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('products');

    const result = await collection.aggregate([
      { $group: { _id: "$category", firstItem: { $first: "$item" } } }
    ]).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['products']

        result = list(collection.aggregate([
            { '$group': { '_id': '$category', 'firstItem': { '$first': '$item' } } }
        ]))

        pprint(result)

    except Exception as e:
        print(f"An error occurred: {e}")

    finally:
        if client:
            client.close()

example()
```

$floor

Baru dari versi 4.0.

`$floor`Operator di Amazon DocumentDB mengembalikan bilangan bulat terbesar yang kurang dari atau sama dengan nomor yang ditentukan. Operator ini berguna untuk membulatkan nilai numerik.

Parameter

- `expression`: Ekspresi numerik untuk membulatkan ke bawah.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$floor` operator untuk membulatkan nilai desimal ke bilangan bulat terdekat.

Buat dokumen sampel

```
db.numbers.insertOne({ value: 3.14 });
```

Contoh kueri

```
db.numbers.aggregate([
  { $project: { _id: 0, floored: { $floor: "$value" } } }
]);
```

Keluaran

```
{ "floored" : 3 }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$floor` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

Tahap `$geoNear` agregasi mengembalikan dokumen dalam urutan kedekatan dengan titik tertentu. Ini menghitung jarak dari titik dan termasuk jarak dalam dokumen output.

Parameter

- `near`: Titik dari mana untuk menghitung jarak, ditentukan sebagai GeoJSON atau koordinat warisan.
- `distanceField`: Nama bidang untuk menyimpan jarak yang dihitung.
- `spherical`: Boolean menunjukkan apakah akan menggunakan geometri bola (diperlukan untuk titik GeoJSON).
- `maxDistance`: Opsional. Jarak maksimum dari titik pusat.
- `minDistance`: Opsional. Jarak minimum dari titik pusat.
- `query`: Opsional. Kriteria filter tambahan untuk diterapkan.
- `limit`: Opsional. Jumlah maksimum dokumen untuk dikembalikan.
- `key`: Opsional. Bidang yang akan digunakan untuk kueri geospasial ketika beberapa indeks geospasial ada.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan menggunakan `$geoNear` panggung untuk menemukan toko terdekat dengan lokasi tertentu.

Buat dokumen sampel

```
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] } }
]);
```

Contoh kueri

```
db.stores.aggregate([
  {
    $geoNear: {
      near: { type: "Point", coordinates: [-122.4, 37.8] },
      distanceField: "distance",
      spherical: true
    }
```

```
  }
]);
```

Keluaran

```
[
  { _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 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan tahap `$geoNear` agregasi, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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()
```

$group

Tahap `$group` agregasi di Amazon DocumentDB memungkinkan Anda mengelompokkan dokumen dengan ekspresi tertentu dan melakukan berbagai operasi akumulatif pada data yang dikelompokkan. Ini dapat berguna untuk tugas-tugas seperti menghitung total, rata-rata, atau statistik lain berdasarkan data yang dikelompokkan.

Parameter

- `_id`: Menentukan ekspresi dimana dokumen masukan harus dikelompokkan. Ini bisa berupa nama bidang, ekspresi yang dihitung, atau kombinasi keduanya.

- `accumulator expressions`: (opsional) Satu atau lebih ekspresi akumulator yang harus diterapkan pada data yang dikelompokkan. Ekspresi ini menggunakan operator akumulator yang disebutkan di atas.

Contoh (MongoDB Shell)

Contoh berikut mengelompokkan pelanggan berdasarkan kota mereka dan menghitung jumlah total pesanan untuk setiap kota.

Buat dokumen sampel

```
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 }] }
]);
```

Contoh kueri

```
db.customers.aggregate([
  {
    $group: {
      _id: "$city",
      totalOrders: { $sum: { $sum: "$orders.amount" } }
    }
  }
]);
```

Keluaran

```
[
  { _id: 'Chicago', totalOrders: 150 },
  { _id: 'Los Angeles', totalOrders: 450 },
  { _id: 'New York', totalOrders: 500 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$group` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

Operator `$gt` agregasi membandingkan dua nilai dan kembali `true` jika nilai pertama lebih besar dari yang kedua, jika tidak kembali. `false`

Parameter

- `expression1`: Nilai pertama untuk membandingkan.
- `expression2`: Nilai kedua untuk membandingkan.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$gt` operator untuk mengidentifikasi produk yang melebihi ambang harga.

Buat dokumen sampel

```
db.products.insertMany([
```

```
  { _id: 1, name: "Laptop", price: 1200 },
  { _id: 2, name: "Mouse", price: 25 },
  { _id: 3, name: "Monitor", price: 400 }
]);
```

Contoh kueri

```
db.products.aggregate([
  {
    $project: {
      name: 1,
      price: 1,
      expensive: { $gt: ["$price", 100] }
    }
  }
]);
```

Keluaran

```
[
  { _id: 1, name: 'Laptop', price: 1200, expensive: true },
  { _id: 2, name: 'Mouse', price: 25, expensive: false },
  { _id: 3, name: 'Monitor', price: 400, expensive: true }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan operator `$gt` agregasi, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Operator `$gte` agregasi membandingkan dua nilai dan kembali `true` jika nilai pertama lebih besar dari atau sama dengan yang kedua, jika tidak kembali. `false`

Parameter

- `expression1`: Nilai pertama untuk membandingkan.
- `expression2`: Nilai kedua untuk membandingkan.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan menggunakan `$gte` operator untuk memeriksa apakah siswa lulus ujian.

Buat dokumen sampel

```
db.students.insertMany([
  { _id: 1, name: "Alice", score: 85 },
  { _id: 2, name: "Bob", score: 60 },
  { _id: 3, name: "Charlie", score: 72 }
]);
```

Contoh kueri

```
db.students.aggregate([
  {
    $project: {
      name: 1,
      score: 1,
      passed: { $gte: ["$score", 70] }
    }
  }
]);
```

Keluaran

```
[
  { _id: 1, name: 'Alice', score: 85, passed: true },
  { _id: 2, name: 'Bob', score: 60, passed: false },
```

```
  { _id: 3, name: 'Charlie', score: 72, passed: true }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan operator `$gte` agregasi, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

$hourOperator mengekstrak komponen jam dari bidang tanggal atau stempel waktu.

Parameter

- dateExpression: Tanggal penerapan operator. Ini harus diselesaikan ke tanggal BSON yang valid (misalnya, bidang seperti $createDat atau tanggal literal).

Parameter juga dapat ditentukan sebagai dokumen dalam format berikut:

{tanggal:<dateExpression>, zona waktu:<timezoneExpression>}

Ini memungkinkan untuk menerapkan operasi tanggal sadar zona waktu.

```
- `<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.
```

Contoh (MongoDB Shell)

Contoh berikut menunjukkan cara menggunakan $hour operator untuk mengekstrak komponen jam dari bidang tanggal dan mengelompokkan data yang sesuai.

Buat dokumen sampel

```
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") }
]);
```

Contoh kueri

```
db.events.aggregate([
  {
    $project: {
      hour: { $hour: "$timestamp" }
    }
  },
  {
    $group: {
      _id: "$hour",
      count: { $sum: 1 }
    }
  },
  {
    $sort: { _id: 1 }
  }
]);
```

Keluaran

```
[
  { "_id": 8, "count": 1 },
  { "_id": 10, "count": 1 },
  { "_id": 12, "count": 1 },
  { "_id": 16, "count": 1 },
  { "_id": 23, "count": 1 }
]
```

Kueri ini mengelompokkan peristiwa berdasarkan komponen jam `timestamp` bidang dan menghitung jumlah peristiwa untuk setiap jam.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$hour` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$ifNull`Operator digunakan untuk mengembalikan nilai tertentu jika ekspresi input mengevaluasi ke null atau undefined. Operator ini dapat berguna dalam skenario di mana Anda ingin memberikan nilai default atau menangani null/undefined kasus.

Parameter

- `expression`: Ekspresi untuk mengevaluasi.
- `replacement`: Nilai untuk kembali jika <`expression`> mengevaluasi ke null atau undefined.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$ifNull` operator untuk memberikan nilai default ketika `name` bidang adalah null atau undefined.

Buat dokumen sampel

```
db.users.insertMany([
  { _id: 1, name: "John" },
  { _id: 2, name: null },
  { _id: 3 }
]);
```

Contoh kueri

```
db.users.aggregate([
  {
    $project: {
      _id: 1,
      name: { $ifNull: ["$name", "No Name"] }
    }
  }
]);
```

Keluaran

```
[
  { "_id": 1, "name": "John" },
```

```
  { "_id": 2, "name": "No Name" },
  { "_id": 3, "name": "No Name" }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$ifNull` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Operator `$in` agregasi memeriksa apakah nilai tertentu ada dalam array. Ia kembali `true` jika nilai ditemukan dalam array, dan `false` sebaliknya.

Parameter

- `value`: Nilai untuk mencari.
- `array`: Array untuk mencari di dalam.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$in` operator untuk memeriksa apakah keterampilan tertentu ada dalam keahlian masing-masing karyawan.

Buat dokumen sampel

```
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"] }
]);
```

Contoh kueri

```
db.employees.aggregate([
  {
    $project: {
      name: 1,
      hasPython: { $in: ["Python", "$skills"] }
    }
  }
]);
```

Keluaran

```
[
  { _id: 1, name: 'Sarah', hasPython: true },
  { _id: 2, name: 'Mike', hasPython: false },
  { _id: 3, name: 'Emma', hasPython: true }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan operator `$in` agregasi, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');
```

```
async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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()
```

$ indeks OfArray

`$indexOfArray`Operator di Amazon DocumentDB digunakan untuk menemukan indeks kemunculan pertama elemen tertentu dalam array. Operator ini mengembalikan posisi indeks berbasis nol dari elemen pertama dalam array yang cocok dengan nilai yang ditentukan. Jika nilai tidak ditemukan, ia mengembalikan -1.

Parameter

- `array`: Array untuk mencari.
- `value`: Nilai untuk mencari dalam array.
- `start`: (opsional) Posisi dalam array untuk memulai pencarian dari. Nilai default-nya adalah 0.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan OfArray operator $index untuk menemukan indeks kemunculan pertama dari elemen “mangga” dalam array “buah-buahan” untuk setiap dokumen.

Buat dokumen sampel

```
db.collection.insertMany([
  { _id: 1, fruits: ["apple", "banana", "cherry", "durian"] },
  { _id: 2, fruits: ["mango", "orange", "pineapple"] },
  { _id: 3, fruits: ["kiwi", "lemon", "mango"] }
]);
```

Contoh kueri

```
db.collection.aggregate([
  {
    $project: {
      _id: 1,
      fruitIndex: { $indexOfArray: ["$fruits", "mango"] }
```

```
    }
  }
]);
```

Keluaran

```
{ "_id" : 1, "fruitIndex" : 1 }
{ "_id" : 2, "fruitIndex" : 0 }
{ "_id" : 3, "fruitIndex" : 2 }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$indexOfArray` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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()
```

$ indeks OfBytes

OfBytes Operator $index di Amazon DocumentDB digunakan untuk menemukan indeks awal substring dalam string, berdasarkan posisi byte karakter. Ini dapat berguna ketika bekerja dengan data teks yang mungkin berisi karakter multi-byte, seperti yang ditemukan dalam skrip non-Latin.

Parameter

- `string`: String masukan untuk mencari.
- `substring`: Substring untuk mencari dalam string input.
- `[<start>]`: (opsional) Posisi awal (berbasis nol) pencarian. Jika tidak ditentukan, pencarian dimulai pada awal string.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$indexOfBytes` untuk menemukan indeks karakter tanda hubung pertama dalam satu set string yang mewakili lokasi meja.

Buat dokumen sampel

```
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" }
]);
```

Contoh kueri

```
db.people.aggregate([
  { $project: { stateLocation: { $indexOfBytes: ["$Desk", "-"] } } }
]);
```

Keluaran

```
{ "_id" : 1, "stateLocation" : 11 }
{ "_id" : 2, "stateLocation" : 6 }
{ "_id" : 3, "stateLocation" : 7 }
{ "_id" : 4, "stateLocation" : 8 }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$indexOfBytes` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('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

`$indexOfCP`Operator di Amazon DocumentDB digunakan untuk menemukan indeks, dalam titik kode (CP), dari kemunculan pertama substring tertentu dalam ekspresi string. Ini dapat berguna saat mengurai dan mengekstrak konten dari bidang string.

Parameter

- `string expression`: String untuk mencari.
- `substring`: Substring untuk mencari.
- `[<start>]`: (opsional) Posisi untuk memulai pencarian (indeks berbasis nol). Default-nya adalah 0.

Contoh (MongoDB Shell)

Dalam contoh ini, kami menggunakan `$indexOfCP` operator untuk menemukan indeks kemunculan pertama karakter tanda hubung - di bidang Desk setiap dokumen.

Buat dokumen sampel

```
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"}
])
```

Contoh kueri

```
db.people.aggregate([
  { $project: { stateLocation: { $indexOfCP: [ "$Desk", "-"] } } }
])
```

Keluaran

```
{ "_id" : 1, "stateLocation" : 10 }
{ "_id" : 2, "stateLocation" : 6 }
{ "_id" : 3, "stateLocation" : 7 }
{ "_id" : 4, "stateLocation" : 8 }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$indexOfCP` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

Tahap `$indexStats` agregasi di Amazon DocumentDB memberikan wawasan tentang penggunaan indeks dalam koleksi. Operator ini memungkinkan Anda memantau pola akses indeks Anda, yang dapat membantu Anda membuat keputusan berdasarkan informasi tentang manajemen dan pengoptimalan indeks.

Parameter

Tidak ada

Contoh (MongoDB Shell)

Contoh berikut menunjukkan cara menggunakan `$indexStats` operator untuk menganalisis penggunaan indeks dalam koleksi Amazon DocumentDB.

Buat dokumen sampel

```
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 }
]);
```

Contoh kueri

```
db.grocery.aggregate([
  { $indexStats: {} }
]);
```

Keluaran

```
[
  {
    "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")
    }
  }
]
```

Dalam contoh ini, `$indexStats` operator menunjukkan bahwa `_id_` indeks telah diakses 5 kali, dan `product_1` indeks telah diakses 10 kali sejak reset terakhir atau reboot server.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$indexStats` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$isArray`Operator di Amazon DocumentDB digunakan untuk memeriksa apakah bidang dalam dokumen adalah array. Operator ini dapat berguna dalam pipeline agregasi dan ekspresi bersyarat untuk menangani bidang tipe array.

Parameter

- `field`: Jalur bidang untuk memeriksa apakah itu adalah array.

Contoh (MongoDB Shell)

Contoh ini menunjukkan cara menggunakan `$isArray` operator untuk mengidentifikasi dokumen di mana bidang “inventaris” adalah array.

Buat dokumen sampel

```
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}]}
]);
```

Contoh kueri

```
db.videos.aggregate([
  {
    $match: {
```

```
      $isArray: "$inventory"
    }
  },
  {
    $project: {
      _id: 1,
      name: 1,
      "inventory.location": 1,
      "inventory.count": 1
    }
  }
]).pretty();
```

Keluaran

```
{
  "_id": 4,
  "name": "Bravemind",
  "inventory": [
    {
      "location": "Mason City",
      "count": 250
    },
    {
      "location": "Des Moines",
      "count": 1000
    },
    {
      "location": "Ames",
      "count": 500
    }
  ]
}
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$isArray` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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()
```

$ iso DayOfWeek

`$isoDayOfWeek`Operator di Amazon DocumentDB mengembalikan hari ISO dalam seminggu untuk tanggal sebagai nilai integer. Sistem tanggal minggu ISO mendefinisikan setiap minggu dimulai pada hari Senin dan berakhir pada hari Minggu, dengan minggu 1 menjadi minggu yang berisi Kamis pertama tahun itu.

Parameter

- `expression`: Ekspresi tanggal untuk mengembalikan hari ISO dalam seminggu.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$isoDayOfWeek` operator untuk mengambil hari ISO dalam seminggu untuk satu set dokumen acara.

Buat dokumen sampel

```
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") }
]);
```

Contoh kueri

```
db.events.aggregate([
  { $project: {
    _id: 1,
    eventDate: 1,
    isoDayOfWeek: { $isoDayOfWeek: "$eventDate" }
  }}
]);
```

Keluaran

```
[
  { "_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 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$isoDayOfWeek` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('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

`$isoWeek`Operator di Amazon DocumentDB mengembalikan nomor minggu ISO untuk tanggal. Sistem tanggal minggu ISO adalah cara penomoran minggu dalam setahun, di mana minggu pertama tahun baru adalah minggu yang berisi Kamis pertama tahun itu. Ini berbeda dengan kalender Gregorian, di mana minggu pertama tahun baru adalah minggu yang berisi 1 Januari.

Parameter

Tidak ada

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$isoWeek` operator untuk mengambil nomor minggu ISO untuk tanggal tertentu.

Buat dokumen sampel

```
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") }
])
```

Contoh kueri

```
db.dates.aggregate([
  {
    $project: {
      _id: 1,
      isoWeek: { $isoWeek: "$date" }
    }
  }
])
```

Keluaran

```
[
  { "_id": 1, "isoWeek": 52 },
  { "_id": 2, "isoWeek": 52 },
```

```
  { "_id": 3, "isoWeek": 1 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$isoWeek` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

`$isoWeekYear`Operator di Amazon DocumentDB mengembalikan nomor tahun ISO 8601 minggu untuk tanggal tertentu. Nomor tahun minggu ISO berbeda dari tahun kalender Gregorian karena tahun minggu dapat berbeda dari tahun kalender, terutama pada awal dan akhir tahun.

Parameter

- `expression`: Ekspresi tanggal untuk mengembalikan nomor tahun ISO 8601 minggu.

Contoh (MongoDB Shell)

Contoh ini menunjukkan cara menggunakan `$isoWeekYear` operator untuk mengambil tahun ISO 8601 minggu untuk bidang tanggal setiap dokumen dalam koleksi peristiwa.

Buat dokumen sampel

```
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") }
]);
```

Contoh kueri

```
db.events.aggregate([
  { $project: {
    name: 1,
    isoWeekYear: { $isoWeekYear: "$date" }
  }}
]);
```

Keluaran

```
[
  { "_id" : 1, "name" : "Event 1", "isoWeekYear" : 2023 },
  { "_id" : 2, "name" : "Event 2", "isoWeekYear" : 2023 },
  { "_id" : 3, "name" : "Event 3", "isoWeekYear" : 2023 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$isoWeekYear` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('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

`$last`Operator di Amazon DocumentDB digunakan untuk mengembalikan elemen terakhir dalam array yang cocok dengan kriteria kueri. Hal ini sangat berguna untuk mengambil elemen terbaru atau terakhir dalam array yang memenuhi kondisi tertentu.

Parameter

- `expression`: Ekspresi untuk mencocokkan elemen array.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$last` operator dalam kombinasi dengan `$filter` untuk mengambil elemen terakhir dari array yang memenuhi kondisi tertentu (misalnya, subjek adalah 'sains').

Buat dokumen sampel

```
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 }
    ]
  }
]);
```

Contoh kueri

```
db.collection.aggregate([
    { $match: { name: "John" } },
    {
      $project: {
        name: 1,
        lastScienceScore: {
          $last: {
            $filter: {
              input: "$scores",
```

```
            as: "score",
            cond: { $eq: ["$$score.subject", "science"] }
          }
        }
      }
    }
  }
]);
```

Keluaran

```
[
  {
    _id: 1,
    name: 'John',
    lastScienceScore: { subject: 'science', score: 90 }
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$last` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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: { name: "John" } },
    {
      $project: {
        name: 1,
        lastScienceScore: {
```

```
          $last: {
            $filter: {
              input: "$scores",
              as: "score",
              cond: { $eq: ["$$score.subject", "science"] }
            }
          }
        }
      }
    }
  ]).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.collection

    pipeline = [
        { "$match": { "name": "John" } },
        {
            "$project": {
                "name": 1,
                "lastScienceScore": {
                    "$last": {
                        "$filter": {
                            "input": "$scores",
                            "as": "score",
                            "cond": { "$eq": ["$$score.subject", "science"] }
                        }
```

```
                    }
                }
            }
        }
    ]

    result = list(collection.aggregate(pipeline))

    print(result)

    client.close()

example()
```

$let

`$let`Operator di Amazon DocumentDB digunakan untuk mengikat variabel ke nilai dan menggunakan variabel tersebut dalam ekspresi. Ini memungkinkan Anda untuk menentukan variabel lokal yang dapat digunakan dalam ekspresi berikutnya dalam tahap yang sama dari pipa agregasi.

Parameter

- `vars`: Objek yang mendefinisikan variabel yang akan digunakan dalam ekspresi.
- `in`: Ekspresi di mana variabel yang didefinisikan dalam parameter vars digunakan.

Contoh (MongoDB Shell)

Contoh ini menunjukkan penggunaan `$let` operator untuk menghitung luas persegi panjang.

Buat dokumen sampel

```
db.shapes.insertMany([
  { name: "Rectangle 1", length: 5, width: 3 },
  { name: "Rectangle 2", length: 7, width: 4 },
  { name: "Rectangle 3", length: 6, width: 2 }
]);
```

Contoh kueri

```
db.shapes.aggregate([
```

```
  {
    $project: {
      name: 1,
      area: {
        $let: {
          vars: {
            length: "$length",
            width: "$width"
          },
          in: {
            $multiply: ["$$length", "$$width"]
          }
        }
      }
    }
  }
])
```

Keluaran

```
[
  {
    "_id": ObjectId("6161e5b1a3eba3c7f2960d03"),
    "name": "Rectangle 1",
    "area": 15
  },
  {
    "_id": ObjectId("6161e5b1a3eba3c7f2960d04"),
    "name": "Rectangle 2",
    "area": 28
  },
  {
    "_id": ObjectId("6161e5b1a3eba3c7f2960d05"),
    "name": "Rectangle 3",
    "area": 12
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$let` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$limit`Operator di Amazon DocumentDB digunakan untuk membatasi jumlah dokumen yang dikembalikan oleh kueri. Ini mirip dengan operator `$limit` MongoDB, tetapi ada beberapa pertimbangan khusus saat menggunakannya dengan Amazon DocumentDB.

Di Amazon DocumentDB, `$limit` operator berguna untuk pagination, di mana Anda ingin mengambil subset dari total dokumen yang cocok. Ini memungkinkan Anda untuk mengontrol jumlah dokumen yang dikembalikan dalam setiap respons, meningkatkan kinerja dan mengurangi jumlah data yang ditransfer melalui jaringan.

Parameter

- `limit`: Jumlah maksimum dokumen untuk dikembalikan. Ini harus berupa nilai integer non-negatif.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan cara menggunakan `$limit` operator untuk mengembalikan maksimal satu dokumen yang cocok dengan filter yang diberikan.

Buat dokumen sampel

```
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" }
]);
```

Contoh kueri

```
db.test.createIndex({ comments: "text" });

db.test.find({
  $and: [
    { star_rating: 5 },
    { $text: { $search: "couch" } }
  ]
}).limit(1);
```

Keluaran

```
[ { _id: 2, star_rating: 5, comments: 'comfortable couch' } ]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$limit` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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()
```

$literal

`$literal`Operator di Amazon DocumentDB digunakan untuk mewakili nilai literal dalam tahap pipeline agregasi. Ini memungkinkan Anda untuk memasukkan nilai tertentu, seperti angka, string, atau boolean, tanpa menafsirkannya sebagai referensi bidang atau ekspresi.

Operator ini sangat berguna ketika Anda perlu menyertakan nilai literal sebagai bagian dari pipeline agregasi yang lebih kompleks, seperti saat membuat filter kueri dinamis atau melakukan perhitungan.

Parameter

Tidak ada

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$literal` operator untuk menyertakan nilai literal dalam pipeline agregasi. `$literal`Operator digunakan untuk memasukkan nilai 18 sebagai nilai literal dalam ekspresi $gt. Hal ini memungkinkan jalur agregasi untuk membandingkan bidang usia dengan nilai literal 18 untuk menentukan apakah orang tersebut dewasa.

Buat dokumen sampel

```
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" }
]);
```

Contoh kueri

```
db.collection.aggregate([
  {
    $project: {
      name: 1,
      age: 1,
      city: 1,
      isAdult: { $gt: ["$age", { $literal: 18 }] }
    }
  }
]);
```

Keluaran

```
[
  {
    "_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
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$literal` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Baru dari versi 4.0.

`$ln`Operator di Amazon DocumentDB menghitung logaritma natural (basis e) dari nomor tertentu. Ia mengembalikan logaritma angka ke basis e.

Parameter

- `expression`: Angka yang logaritma natural akan dihitung.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$log` operator untuk menghitung logaritma natural dari suatu angka.

Buat dokumen sampel

```
db.numbers.insertMany([
  { _id: 1, value: 10 },
  { _id: 2, value: 100 },
  { _id: 3, value: 1000 }
]);
```

Contoh kueri

```
db.numbers.aggregate([
  { $project: {
    _id: 1,
    naturalLog: { $ln: "$value" }
  }}
]);
```

Keluaran

```
[
  { "_id" : 1, "naturalLog" : 2.302585092994046 },
  { "_id" : 2, "naturalLog" : 4.605170185988092 },
  { "_id" : 3, "naturalLog" : 6.907755278982137 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$ln` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
```

```
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Baru dari versi 4.0.

`$log`Operator di Amazon DocumentDB menghitung logaritma natural dari suatu angka. Ia mengembalikan logaritma basis-e dari nomor yang ditentukan.

Parameter

- `expression`: Angka yang logaritma natural akan dihitung.
- `base`: Nilai dasar untuk menghitung log.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$log` operator untuk menghitung logaritma natural dari suatu angka.

Buat dokumen sampel

```
db.numbers.insertMany([
  { _id: 1, value: 10 },
  { _id: 2, value: 100 },
  { _id: 3, value: 1000 }
]);
```

Contoh kueri

```
db.numbers.aggregate([
  { $project: {
    _id: 1,
    naturalLog: { $log: ["$value", 10] }
  }}
]);
```

Keluaran

```
[
  { "_id" : 1, "naturalLog" : 1 },
```

```
  { "_id" : 2, "naturalLog" : 2 },
  { "_id" : 3, "naturalLog" : 2.9999999999999996 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$log` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Baru dari versi 4.0.

`$log10`Operator di Amazon DocumentDB digunakan untuk menghitung logaritma basis-10 dari suatu angka. Hal ini berguna untuk melakukan perhitungan logaritmik pada bidang numerik dalam pipeline agregasi.

Parameter

- `expression`: Ekspresi numerik yang logaritma basis-10 akan dihitung.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$log10` operator untuk menghitung logaritma basis-10 dari bidang numerik.

Buat dokumen sampel

```
db.numbers.insertMany([
  { _id: 1, value: 1 },
  { _id: 2, value: 10 },
  { _id: 3, value: 100 },
  { _id: 4, value: 1000 }
]);
```

Contoh kueri

```
db.numbers.aggregate([
  {
    $project: {
      _id: 1,
      log10Value: { $log10: "$value" }
    }
  }
]);
```

Keluaran

```
[
  { "_id": 1, "log10Value": 0 },
  { "_id": 2, "log10Value": 1 },
  { "_id": 3, "log10Value": 2 },
  { "_id": 4, "log10Value": 3 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$log10` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

Operator `$lt` agregasi membandingkan dua nilai dan kembali `true` jika nilai pertama kurang dari yang kedua, jika tidak kembali. `false`

Parameter

- `expression1`: Nilai pertama untuk membandingkan.
- `expression2`: Nilai kedua untuk membandingkan.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan menggunakan `$lt` operator untuk mengidentifikasi item stok rendah.

Buat dokumen sampel

```
db.warehouse.insertMany([
  { _id: 1, item: "Bolts", stock: 5 },
  { _id: 2, item: "Nuts", stock: 25 },
  { _id: 3, item: "Screws", stock: 8 }
]);
```

Contoh kueri

```
db.warehouse.aggregate([
  {
    $project: {
      item: 1,
      stock: 1,
      lowStock: { $lt: ["$stock", 10] }
    }
```

```
  }
]);
```

Keluaran

```
[
  { _id: 1, item: 'Bolts', stock: 5, lowStock: true },
  { _id: 2, item: 'Nuts', stock: 25, lowStock: false },
  { _id: 3, item: 'Screws', stock: 8, lowStock: true }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan operator `$lt` agregasi, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Operator `$lte` agregasi membandingkan dua nilai dan kembali `true` jika nilai pertama kurang dari atau sama dengan yang kedua, jika tidak kembali. `false`

Parameter

- `expression1`: Nilai pertama untuk membandingkan.
- `expression2`: Nilai kedua untuk membandingkan.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$lte` operator untuk mengidentifikasi item yang ramah anggaran.

Buat dokumen sampel

```
db.menu.insertMany([
  { _id: 1, dish: "Salad", price: 8 },
  { _id: 2, dish: "Pasta", price: 12 },
  { _id: 3, dish: "Soup", price: 6 }
]);
```

Contoh kueri

```
db.menu.aggregate([
  {
    $project: {
      dish: 1,
      price: 1,
      affordable: { $lte: ["$price", 10] }
    }
  }
]);
```

Keluaran

```
[
  { _id: 1, dish: 'Salad', price: 8, affordable: true },
  { _id: 2, dish: 'Pasta', price: 12, affordable: false },
  { _id: 3, dish: 'Soup', price: 6, affordable: true }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan operator `$lte` agregasi, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Tahap `$lookup` agregasi di Amazon DocumentDB memungkinkan Anda melakukan gabungan luar kiri antara dua koleksi. Operasi ini memungkinkan Anda menggabungkan data dari beberapa koleksi berdasarkan nilai bidang yang cocok. Ini sangat berguna ketika Anda perlu memasukkan data dari koleksi terkait ke dalam hasil kueri Anda.

Parameter

- `from`: Nama koleksi untuk melakukan bergabung dengan.
- `localField`: Bidang dari dokumen masukan untuk mencocokkan dengan`foreignField`.
- `foreignField`: Bidang dari dokumen dalam `from` koleksi untuk dicocokkan dengan`localField`.
- `as`: Nama bidang baru untuk ditambahkan ke dokumen keluaran yang berisi dokumen yang cocok dari `from` koleksi.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan `$lookup` operasi sederhana yang menggabungkan data dari `orders` koleksi ke dalam koleksi. `customers`

Buat dokumen sampel

```
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 }
]);
```

Contoh kueri

```
db.customers.aggregate([
  {
```

```
    $lookup: {
      from: "orders",
      localField: "_id",
      foreignField: "customer_id",
      as: "orders"
    }
  }
]);
```

Keluaran

```
[
  {
    _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 } ]
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$lookup` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function 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()
```

$ltrim

Baru dari versi 4.0.

Tidak didukung oleh cluster elastis.

`$ltrim`Operator di Amazon DocumentDB digunakan untuk menghapus karakter utama dari string. Secara default, ini menghapus karakter spasi utama, tetapi Anda juga dapat menentukan satu set karakter untuk dihapus dengan meneruskan argumen karakter.

Parameter

- `input`: String masukan dari mana untuk menghapus karakter spasi utama.
- `chars`: (opsional) Untuk menghapus karakter tertentu.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$ltrim` untuk menghapus karakter tertentu (“*”) dari awal string.

Buat dokumen sampel

```
db.collection.insertMany([
  { name: " *John Doe", age: 30 },
  { name: "Jane Doe*", age: 25 },
  { name: "  Bob Smith  ", age: 35 }
]);
```

Contoh kueri

```
db.collection.aggregate([
  {
    $project: {
      _id: 0,
      name: {
        $ltrim: { input: "$name", chars: " *" }
      },
      age: 1
    }
  }
]);
```

Keluaran

```
[
  { "name": "John Doe", "age": 30 },
  { "name": "Jane Doe ", "age": 25 },
  { "name": "Bob Smith  ", "age": 35 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$ltrim` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function 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

$mapOperator di Amazon DocumentDB memungkinkan Anda menerapkan ekspresi tertentu ke setiap elemen dalam array dan mengembalikan array baru dengan elemen yang diubah. Operator ini sangat berguna untuk memanipulasi dan mengubah data dalam array, yang dapat membantu menyederhanakan kode aplikasi Anda dan meningkatkan kinerja kueri dengan mendorong pemrosesan array ke tingkat database.

Parameter

- `input`: Array yang akan diubah.
- `as`: (opsional) Nama variabel yang akan digunakan dalam ekspresi in untuk mewakili elemen saat ini sedang diproses.
- `in`: Ekspresi yang akan diterapkan untuk setiap elemen dalam array input.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan operator $map untuk mengubah array angka, menggandakan setiap nilai.

Buat dokumen sampel

```
db.collection.insertMany([
  { _id: 1, numbers: [1, 2, 3, 4, 5] },
  { _id: 2, numbers: [10, 20, 30, 40, 50] }
])
```

Contoh kueri

```
db.collection.aggregate([
  {
    $project: {
      doubledNumbers: { $map: { input: "$numbers", as: "num", in: { $multiply: ["$
$num", 2] } } }
    }
  }
])
```

Keluaran

```
[
  { _id: 1, doubledNumbers: [2, 4, 6, 8, 10] },
  { _id: 2, doubledNumbers: [20, 40, 60, 80, 100] }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$map` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
```

```
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Tahap `$match` pipeline di Amazon DocumentDB digunakan untuk memfilter dokumen input hanya ke dokumen yang cocok dengan kriteria kueri yang ditentukan. Ini adalah salah satu tahapan pipa yang paling umum digunakan dalam operasi agregasi. `$match`Tahap ini diterapkan sebelum tahap pipa lainnya, memungkinkan Anda untuk secara efisien mengurangi jumlah dokumen yang perlu diproses oleh tahap selanjutnya.

Parameter

- `query`: Dokumen yang menyatakan kriteria seleksi untuk operasi. Dokumen query menggunakan sintaks yang sama dengan `find()` metode.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$match` panggung untuk memfilter dokumen berdasarkan nilai bidang tertentu.

Buat dokumen sampel

```
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" }
]);
```

Contoh kueri

```
db.collection.aggregate([
  { $match: { age: { $gt: 30 } } },
  { $project: { _id: 1, name: 1, city: 1 } }
]);
```

Keluaran

```
[
  { "_id": 3, "name": "Bob", "city": "Chicago" },
  { "_id": 4, "name": "Alice", "city": "Miami" }
]
```

`$match`Tahap menyaring dokumen untuk memasukkan hanya yang `age` bidangnya lebih besar dari 30.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$match` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Tahap `$max` agregasi digunakan untuk mengembalikan nilai maksimum bidang tertentu di semua dokumen dalam tahap pipeline. Operator ini berguna untuk menemukan nilai tertinggi dalam satu set dokumen.

Parameter

- `expression`: Ekspresi yang digunakan untuk menghitung nilai maksimum.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$max` operator untuk menemukan skor maksimum dalam koleksi dokumen siswa. `$group`Tahap mengelompokkan semua dokumen bersama-sama, dan `$max` operator digunakan untuk menghitung nilai maksimum `score` bidang di semua dokumen.

Buat dokumen sampel

```
db.students.insertMany([
  { name: "John", score: 85 },
  { name: "Jane", score: 92 },
  { name: "Bob", score: 78 },
  { name: "Alice", score: 90 }
```

```
])
```

Contoh kueri

```
db.students.aggregate([
  { $group: { _id: null, maxScore: { $max: "$score" } } },
  { $project: { _id: 0, maxScore: 1 } }
])
```

Keluaran

```
[ { maxScore: 92 } ]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$max` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function findMaxScore() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const students = db.collection('students');

  const result = await students.aggregate([
    { $group: { _id: null, maxScore: { $max: "$score" } } }
  ]).toArray();

  console.log(result);
  await client.close();
}

findMaxScore();
```

Python

```
from pymongo import MongoClient
```

```
def find_max_score():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    students = db.students

    result = list(students.aggregate([
        { "$group": { "_id": None, "maxScore": { "$max": "$score" } } }
    ]))

    print(result)
    client.close()

find_max_score()
```

$meta

Operator `$meta` agregasi mengakses metadata yang terkait dengan dokumen dalam pipeline agregasi. Hal ini biasanya digunakan untuk mengambil skor pencarian teks dan mengurutkan hasil berdasarkan relevansi.

Parameter

- `textScore`: Mengambil skor pencarian teks yang menunjukkan relevansi dokumen dengan permintaan pencarian.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$meta` operator dalam pipeline agregasi untuk mengambil dan mengurutkan berdasarkan skor pencarian teks.

Buat dokumen sampel

```
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." }
]);
```

Contoh kueri

```
db.articles.aggregate([
  { $match: { $text: { $search: "Python" } } },
  { $addFields: { score: { $meta: "textScore" } } },
  { $sort: { score: -1 } }
]);
```

Keluaran

```
[
  {
    _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
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan operator `$meta` agregasi, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
```

```
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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()
```

$ bergabung

Diperkenalkan pada 8.0

Tahap `$merge` agregasi di Amazon DocumentDB digunakan untuk menggabungkan hasil tahap pipeline sebelumnya ke dalam pengumpulan target. Hal ini berguna untuk memperbarui atau menyisipkan dokumen dalam koleksi target berdasarkan data dari dokumen masukan.

`$merge`Tahap ini memungkinkan Anda untuk melakukan berbagai tindakan berdasarkan kondisi pencocokan antara dokumen input dan pengumpulan target, seperti:

```
- Insert new documents
- Update existing documents
- Delete documents
- Fail the operation if there are any conflicts
```

Parameter

- `into`: (wajib) Nama koleksi target untuk menggabungkan dokumen masukan ke dalam.
- on: (wajib) Bidang yang akan digunakan sebagai kondisi yang cocok antara dokumen masukan dan pengumpulan target.
- `whenMatched`: (opsional) Tindakan yang harus dilakukan ketika dokumen input cocok dengan dokumen yang ada dalam koleksi target. Nilai yang didukung adalah:`"merge"`,`"replace"`,`"keepExisting"`, dan`"fail"`.
- `whenNotMatched`: (opsional) Tindakan yang harus dilakukan ketika dokumen input tidak cocok dengan dokumen apa pun dalam koleksi target. Nilai yang didukung adalah: `"insert"` dan`"fail"`.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$merge` tahap untuk memperbarui `users` koleksi dengan data baru dari pipeline input.

Buat dokumen sampel

```
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" }
]);
```

Contoh kueri

```
db.inputData.aggregate([
  {
    $merge: {
      into: "users",
      on: "_id",
      whenMatched: "merge",
      whenNotMatched: "insert"
    }
  }
])
```

Keluaran

Setelah menjalankan `$merge` pipeline, `users` koleksi akan berisi dokumen-dokumen berikut:

```
[
  { _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" }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$merge` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

Berikut adalah contoh penggunaan operator $merge dalam aplikasi: Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
```

```
  const client = await MongoClient.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

Berikut adalah contoh menggunakan operator $merge dalam aplikasi Python:

```
from pymongo import MongoClient

def 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

`$mergeObjects`Operator di Amazon DocumentDB digunakan untuk menggabungkan beberapa dokumen atau objek ke dalam satu dokumen. Operator ini sangat berguna ketika Anda perlu menggabungkan konten dari dua atau lebih dokumen atau objek, berpotensi menimpa nilai dari satu objek dengan yang dari yang lain.

Parameter

- `expression1`: Objek pertama yang akan digabungkan.
- `expression2`: (opsional) Objek kedua yang akan digabungkan.
- `expression3`: (opsional) Objek tambahan yang akan digabungkan.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$mergeObjects` operator untuk menggabungkan dua objek.

Buat dokumen sampel

```
db.collection.insertMany([
  { "_id": 1, "name": "John", "address": { "city": "New York", "state": "NY" } },
  { "_id": 2, "name": "Jane", "address": { "city": "Los Angeles", "state": "CA" } }
]);
```

Contoh kueri

```
db.collection.aggregate([
```

```
  {
    $project: {
      "combinedAddress": {
        $mergeObjects: ["$address", { "country": "USA" }]
      }
    }
  }
])
```

Keluaran

```
[
  {
    "_id": 1,
    "combinedAddress": {
      "city": "New York",
      "state": "NY",
      "country": "USA"
    }
  },
  {
    "_id": 2,
    "combinedAddress": {
      "city": "Los Angeles",
      "state": "CA",
      "country": "USA"
    }
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$mergeObjects` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

`$millisecond`Operator di Amazon DocumentDB digunakan untuk mengekstrak bagian milidetik dari nilai tanggal.

Parameter

Tidak ada

Contoh (MongoDB Shell)

Contoh ini menunjukkan bagaimana menggunakan `$millisecond` operator untuk mengekstrak bagian milidetik dari nilai tanggal.

Buat dokumen sampel

```
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")
  }
])
```

Contoh kueri

```
db.events.aggregate([
  {
    $project: {
      name: 1,
      milliseconds: { $millisecond: "$timestamp" }
    }
```

```
  }
])
```

Keluaran

```
[
  {
    "_id": ObjectId("644332a42054ed1b0d15f0c1"),
    "name": "Event 1",
    "milliseconds": 123
  },
  {
    "_id": ObjectId("644332a42054ed1b0d15f0c2"),
    "name": "Event 2",
    "milliseconds": 456
  },
  {
    "_id": ObjectId("644332a42054ed1b0d15f0c3"),
    "name": "Event 3",
    "milliseconds": 789
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$millisecond` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$min`Operator mengembalikan nilai minimum dari array nilai. Ini dapat digunakan dalam tahap agregasi untuk menemukan nilai minimum untuk bidang tertentu di beberapa dokumen.

Parameter

- `expression`: Ekspresi untuk mengevaluasi. Ini bisa berupa jalur bidang, variabel, atau ekspresi apa pun yang menyelesaikan nilai.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$min` operator untuk menemukan nilai minimum `age` bidang di beberapa dokumen.

Buat dokumen sampel

```
db.users.insertMany([
  { name: "John", age: 35 },
  { name: "Jane", age: 28 },
  { name: "Bob", age: 42 },
  { name: "Alice", age: 31 }
]);
```

Contoh kueri

```
db.users.aggregate([
  { $group: { _id: null, minAge: { $min: "$age" } } },
  { $project: { _id: 0, minAge: 1 } }
])
```

Keluaran

```
[ { minAge: 28 } ]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$min` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function findMinAge() {
  const client = await MongoClient.connect('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([
    { $group: {
        _id: null,
        minAge: { $min: "$age" }
      }}
  ]).toArray();

  console.log(result);
  client.close();
}

findMinAge();
```

Python

```
from pymongo import MongoClient

def find_min_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.aggregate([
        { "$group": {
            "_id": None,
            "minAge": { "$min": "$age" }
        }}
    ]))

    print(result)
    client.close()

find_min_age()
```

$minute

Tahap pipeline `$minute` agregasi di Amazon DocumentDB mengekstrak nilai menit dari bidang tanggal atau stempel waktu.

Operator ini berguna saat Anda perlu melakukan perhitungan atau pengelompokan berbasis tanggal dan waktu dalam jalur agregasi Anda.

Parameter

- `expression`: Bidang tanggal atau stempel waktu untuk mengekstrak nilai menit.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$minute` operator untuk mengelompokkan dokumen dengan nilai menit yang diekstrak dari bidang stempel waktu dan menghitung jumlah dokumen di setiap grup.

Buat dokumen sampel

```
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") }
]);
```

Contoh kueri

```
db.events.aggregate([
  {
    $group: {
      _id: {
        minute: { $minute: "$timestamp" }
      },
      count: { $count: {} }
    }
  },
  { $sort: { "_id.minute": 1 } }
]);
```

Keluaran

```
[
  { "_id": { "minute": 30 }, "count": 2 },
  { "_id": { "minute": 31 }, "count": 2 },
  { "_id": { "minute": 32 }, "count": 1 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$minute` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

$modOperator adalah operator aritmatika yang melakukan operasi modulo pada angka. Ia mengembalikan sisa membagi satu nomor dengan yang lain. Operator ini biasanya digunakan untuk menentukan apakah suatu bilangan ganjil atau genap, atau untuk mendistribusikan item ke dalam jumlah grup yang terbatas.

Parameter

- `expression1`: Ekspresi dividen.
- `expression2`: Ekspresi pembagi.

Contoh (MongoDB Shell)

Contoh ini menunjukkan bagaimana menggunakan $mod operator untuk menentukan jumlah widget sisa saat pengiriman dalam paket 100.

Buat dokumen sampel

```
db.widgets.insertMany([
  { "_id" : 1, "widget" : "A", "count" : 80372 },
  { "_id" : 2, "widget" : "B", "count" : 409282 },
  { "_id" : 3, "widget" : "C", "count" : 60739 }
])
```

Contoh kueri

```
db.widgets.aggregate([
  { $addFields: { leftOver: { $mod: [ "$count", 100 ] } } }
])
```

Keluaran

```
[
  { "_id" : 1, "widget" : "A", "count" : 80372, "leftOver" : 72 },
  { "_id" : 2, "widget" : "B", "count" : 409282, "leftOver" : 82 },
  { "_id" : 3, "widget" : "C", "count" : 60739, "leftOver" : 39 }
]
```

Output menunjukkan sisa `count` dibagi dengan 100 untuk setiap dokumen, yang mewakili jumlah widget sisa saat pengiriman dalam paket 100.

Contoh kode

Untuk melihat contoh kode untuk menggunakan $mod perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

$monthOperator di Amazon DocumentDB mengembalikan bulan tanggal sebagai angka antara 1 dan 12. Operator ini berguna untuk mengekstrak komponen bulan dari bidang tanggal dan melakukan agregasi dan analisis berbasis tanggal.

Parameter

- `date_expression`: Ini adalah ekspresi atau bidang yang berisi tanggal atau stempel waktu dari mana Anda ingin mengekstrak bulan.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan cara menggunakan $month operator untuk mengekstrak bulan dari bidang tanggal dan mengelompokkan data berdasarkan bulan.

Buat dokumen sampel

```
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") }
]);
```

Contoh kueri

```
db.sales.aggregate([
  { $group: {
      _id: { month: { $month: "$date" } },
      totalSales: { $sum: "$price" }
    }},
  { $sort: { "_id.month": 1 } }
]);
```

Keluaran

```
[
  { _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 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$month` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$multiply`Operator di Amazon DocumentDB digunakan untuk melipatgandakan nilai dua atau lebih bidang atau ekspresi. Operator ini sangat berguna untuk melakukan operasi aritmatika pada bidang numerik dalam dokumen. Ini dapat digunakan dalam berbagai tahap pipa agregasi, seperti `$project` dan`$addFields`, untuk membuat bidang baru atau memodifikasi yang sudah ada.

Parameter

- `expression1`: Ekspresi numerik pertama yang akan dikalikan.
- `expression2`: Ekspresi numerik kedua yang akan dikalikan.
- `[expression3, ...]`: (opsional) Ekspresi numerik tambahan yang akan dikalikan.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$multiply` untuk menghitung `bonus_miles` dengan mengalikan `base_miles` dan `bonus_rate` untuk pelanggan yang menggunakan kartu kredit untuk perjalanan.

Contoh dokumen

```
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 }
]);
```

Contoh kueri

```
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"]
      }
    }
  }
]);
```

Keluaran

```
[
  { _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 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$multiply` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$natural`Operator di Amazon DocumentDB digunakan untuk mengurutkan dokumen dalam urutan alami mereka, yang merupakan urutan di mana mereka dimasukkan ke dalam koleksi. Ini berbeda dengan perilaku penyortiran default, yaitu mengurutkan dokumen berdasarkan nilai bidang yang ditentukan.

Parameter

Tidak ada

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$natural` operator untuk mengurutkan dokumen dalam koleksi dalam urutan alami mereka.

Buat dokumen sampel

```
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 }
]);
```

Contoh kueri

```
db.people.find({}, { "_id": 1, "name": 1 }).sort({ "$natural": 1 });
```

Keluaran

```
[
  { "_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" }
]
```

Kueri mengurutkan dokumen dalam koleksi dalam urutan alami mereka, yang merupakan urutan di mana mereka dimasukkan.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$natural` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');
```

```
async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Operator $ne agregasi membandingkan dua nilai dan kembali `true` jika mereka tidak sama, jika tidak kembali. `false`

Parameter

- `expression1`: Nilai pertama untuk membandingkan.
- `expression2`: Nilai kedua untuk membandingkan.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan menggunakan $ne operator untuk mengidentifikasi pesanan dengan perubahan status.

Buat dokumen sampel

```
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" }
]);
```

Contoh kueri

```
db.orders.aggregate([
  {
    $project: {
      orderId: 1,
      status: 1,
      expectedStatus: 1,
      needsAttention: { $ne: ["$status", "$expectedStatus"] }
    }
  }
]);
```

Keluaran

```
[
  { _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 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan operator $ne agregasi, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Operator $not agregasi melakukan operasi NOT logis pada ekspresi. Ia kembali `true` jika ekspresi mengevaluasi`false`, dan `false` jika ekspresi mengevaluasi untuk. `true`

Parameter

- `expression`: Ekspresi untuk meniadakan.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan menggunakan $not operator untuk membalikkan nilai boolean.

Buat dokumen sampel

```
db.users.insertMany([
  { _id: 1, name: "Alice", active: true },
  { _id: 2, name: "Bob", active: false },
  { _id: 3, name: "Charlie", active: true }
]);
```

Contoh kueri

```
db.users.aggregate([
```

```
  {
    $project: {
      name: 1,
      active: 1,
      inactive: { $not: ["$active"] }
    }
  }
]);
```

Keluaran

```
[
  { _id: 1, name: 'Alice', active: true, inactive: false },
  { _id: 2, name: 'Bob', active: false, inactive: true },
  { _id: 3, name: 'Charlie', active: true, inactive: false }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan operator $not agregasi, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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()
```

$ objek ToArray

Operator `$objectToArray` agregasi di Amazon DocumentDB mengubah objek (atau dokumen) menjadi array. Masukan ke operator adalah dokumen, dan output terdiri dari elemen array untuk setiap pasangan nilai bidang dalam dokumen input. Operator ini berguna ketika Anda perlu bekerja dengan masing-masing bidang dokumen sebagai array, seperti ketika Anda ingin menemukan dokumen dengan nilai maksimum atau minimum untuk bidang tertentu.

Parameter

- `expression`: Ekspresi dokumen untuk dikonversi menjadi array.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan cara menggunakan `$objectToArray` operator untuk menemukan dokumen dengan inventaris maksimum untuk rantai toko penyewaan video.

Buat dokumen sampel

```
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
    }
  }
```

```
]);
```

Contoh kueri

```
db.videos.aggregate([
  {
    $project: {
      name: 1,
      videos: {
        $objectToArray: "$inventory"
      }
    }
  },
  {
    $unwind: "$videos"
  },
  {
    $group: {
      _id: "$name",
      maxInventory: {
        $max: "$videos.v"
      }
    }
  }
]);
```

Keluaran

```
[
  {
    "_id": "Bravemind",
    "maxInventory": 1000
  },
  {
    "_id": "Live Soft",
    "maxInventory": 1000
  },
  {
    "_id": "Romancing the Rock",
    "maxInventory": 500
  },
  {
    "_id": "Top Pilot",
```

```
    "maxInventory": 1000
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$objectToArray` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

Operator `$or` agregasi mengevaluasi beberapa ekspresi dan mengembalikan `true` jika setidaknya satu ekspresi mengevaluasi. `true` Ia kembali `false` hanya jika semua ekspresi`false`.

Parameter

- `expressions`: Sebuah array ekspresi untuk mengevaluasi.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$or` operator untuk memeriksa apakah produk memenuhi salah satu dari beberapa kriteria.

Buat dokumen sampel

```
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 }
]);
```

Contoh kueri

```
db.items.aggregate([
  {
    $project: {
      name: 1,
      price: 1,
      onSale: 1,
      goodDeal: {
        $or: [
          { $lt: ["$price", 50] },
          { $eq: ["$onSale", true] }
        ]
      }
    }
  }
]);
```

Keluaran

```
[
  { _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 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan operator `$or` agregasi, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

`$out`Operator di Amazon DocumentDB digunakan untuk menulis hasil pipeline agregasi ke koleksi tertentu.

`$out`harus menjadi tahap terakhir dalam pipa.

Parameter

- `output_collection`: Nama koleksi keluaran untuk menulis hasil agregasi ke.

Catatan: Jika koleksi sudah ada, maka akan diganti dengan hasil tahap agregasi.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan cara menggunakan `$out` operator di Amazon DocumentDB untuk menulis hasil pipeline agregasi ke koleksi baru.

Buat dokumen sampel

```
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 }
]);
```

Contoh kueri

```
db.products.aggregate([
  { $group: { _id: "$category", totalPrice: { $sum: "$price" } } },
  { $out: "product_categories" }
])
```

Keluaran

Tidak ada (hasilnya ditulis ke koleksi output).

Pipa agregasi mengelompokkan produk berdasarkan kategori dan menghitung harga total item untuk setiap kategori. `$out`Operator menulis hasilnya ke koleksi baru bernama “product_categories”.

Untuk melihat hasil dalam koleksi output:

```
db.product_categories.find()
[
{ "_id" : "Books", "totalPrice" : 125 },
{ "_id" : "Electronics", "totalPrice" : 300 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$out` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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()
```

$pow

Diperkenalkan pada 8.0

$powOperator di Amazon DocumentDB memungkinkan Anda menaikkan angka ke daya. Ini dapat berguna untuk melakukan perhitungan eksponensial dalam pipeline agregasi Anda.

Parameter

- <number>(wajib): Angka yang akan dinaikkan menjadi kekuatan.
- <exponent>(wajib): Kekuatan yang harus dinaikkan nomor tersebut.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan $pow operator untuk menghitung kuadrat angka.

Buat dokumen sampel

```
db.numbers.insertMany([
  { "_id": 1, "value": 2 },
  { "_id": 2, "value": 3 },
  { "_id": 3, "value": 4 }
```

```
]);
```

Contoh kueri

```
db.numbers.aggregate([
  { $addFields: { "square": { $pow: ["$value", 2] } } }
])
```

Keluaran

```
[
  { "_id": 1, "value": 2, "square": 4 },
  { "_id": 2, "value": 3, "square": 9 },
  { "_id": 3, "value": 4, "square": 16 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan $pow perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

Berikut adalah contoh menggunakan operator $pow dalam aplikasi: Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Berikut adalah contoh menggunakan operator $pow dalam aplikasi Python:

```
from pymongo import MongoClient

def example():
    client = 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

Operator `$push` agregasi mengembalikan array dari semua nilai dari ekspresi tertentu untuk setiap kelompok. Hal ini biasanya digunakan dalam `$group` tahap untuk mengakumulasi nilai-nilai ke dalam array.

Parameter

- `expression`: Ekspresi untuk mengevaluasi setiap dokumen dalam grup.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan menggunakan `$push` operator untuk mengumpulkan semua nama produk untuk setiap kategori.

Buat dokumen sampel

```
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 }
]);
```

Contoh kueri

```
db.sales.aggregate([
  {
    $group: {
      _id: "$category",
      products: { $push: "$product" }
    }
  }
]);
```

Keluaran

```
[
  { _id: 'Furniture', products: [ 'Desk', 'Chair' ] },
  { _id: 'Electronics', products: [ 'Laptop', 'Mouse', 'Keyboard' ] }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan operator $push agregasi, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('sales');
```

```
  const result = await collection.aggregate([
    {
      $group: {
        _id: "$category",
        products: { $push: "$product" }
      }
    }
  ]).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['sales']

    result = list(collection.aggregate([
        {
            '$group': {
                '_id': '$category',
                'products': { '$push': '$product' }
            }
        }
    ]))

    print(result)
    client.close()

example()
```

$project

`$project`Operator di Amazon DocumentDB memungkinkan Anda untuk secara selektif menyertakan atau mengecualikan bidang dari dokumen keluaran, meneruskan nilai ke tahap pipeline berikutnya, dan menghitung bidang baru dari nilai dokumen masukan.

Parameter

- `field`: Bidang untuk menyertakan atau mengecualikan dari dokumen keluaran, dapat berupa jalur bidang (misalnya, “a.b.c”).
- `1`atau`true`: Termasuk bidang dalam output.
- `0`atau`false`: Mengecualikan bidang dari output.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$project` operator pada koleksi siswa

Buat dokumen sampel

```
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" }
  ]);
```

Kueri ini hanya mencakup `name` dan `math` bidang dalam output. `_id`Bidang disertakan secara default kecuali secara eksplisit dikecualikan.

```
db.students.aggregate([
  { $project: { "name": 1, "math": 1 } }
])
```

Keluaran

```
{ _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 }
```

Kueri ini mengecualikan `grade` dan `_id` bidang dari output, menampilkan semua bidang lainnya (`name`,`math`,`science`).

```
db.students.aggregate([
  { $project: { "grade": 0, "_id": 0 } }
])
```

Keluaran

```
{ 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 }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$project` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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()
```

$ rand

Baru dari versi 8.0

`$rand`Operator di Amazon DocumentDB digunakan untuk menghasilkan angka acak antara 0 dan 1.

Parameter

Tidak ada

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$rand` operator untuk secara acak memilih dua dokumen dari koleksi. `temp`

Buat dokumen sampel

```
db.items.insertMany([
  { "name": "pencil", "quantity": 110 },
  { "name": "pen", "quantity": 159 }
])
```

Contoh kueri

```
db.items.aggregate([
  {
    $project: {
      randomValue: { $rand: {} }
    }
  }
])
```

Keluaran

```
[
  {
    _id: ObjectId('6924a5edd66dcae121d29517'),
    randomValue: 0.8615243955294392
  },
  {
    _id: ObjectId('6924a5edd66dcae121d29518'),
    randomValue: 0.22815483022099903
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$rand` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Operator `$range` agregasi di Amazon DocumentDB digunakan untuk membuat array angka berurutan dalam rentang tertentu. Operator ini sangat berguna untuk menghasilkan urutan angka,

seperti penanda mil untuk stasiun bantuan dalam perlombaan, seperti yang ditunjukkan dalam contoh berikut.

Parameter

- `start`: Nilai awal untuk rentang.
- `end`: Nilai akhir untuk rentang.
- `step`: (opsional) Nilai langkah yang akan digunakan saat menghasilkan rentang. Jika tidak disediakan, nilai langkah default adalah 1.

Contoh (MongoDB Shell)

Dalam contoh ini, kita akan menggunakan `$range` operator untuk menghasilkan penanda mil untuk stasiun air dalam perlombaan sepeda.

Buat dokumen sampel

```
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 }
]);
```

Contoh kueri

```
db.races.aggregate([
  {
    $project: {
      race: 1,
      "waterStations": { $range: [20, "$distance", 20] }
    }
  }
]);
```

Keluaran

```
[
  {
    _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 ] }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$range` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function 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()
```

$redact

Tahap `$redact` agregasi di Amazon DocumentDB digunakan untuk secara selektif menyertakan atau mengecualikan konten dari dokumen keluaran berdasarkan nilai bidang tertentu. Ini sangat berguna dalam skenario di mana Anda perlu mengontrol visibilitas data sensitif berdasarkan tingkat akses atau izin pengguna.

Parameter

- `$cond`: Ekspresi yang mengevaluasi salah satu`$$KEEP`,`$$PRUNE`, atau `$$DESCEND` untuk setiap bidang dalam dokumen.
- `$$KEEP`: Mempertahankan bidang saat ini dalam dokumen output.
- `$$PRUNE`: Menghapus bidang saat ini dari dokumen output.
- `$$DESCEND`: Secara rekursif menerapkan `$redact` tahap ke bidang saat ini, yang merupakan objek atau array.

Contoh (MongoDB Shell)

Dalam contoh ini, kita akan menggunakan `$redact` tahap untuk memfilter pesanan berdasarkan statusnya, hanya menampilkan pesanan dengan nilai status tertentu.

Buat dokumen sampel

```
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" }
])
```

Contoh kueri

```
db.orders.aggregate([
  {
    $redact: {
      $cond: {
        if: { $in: ["$status", ["shipped", "processing"]] },
        then: "$$KEEP",
        else: "$$PRUNE"
      }
    }
  }
])
```

Keluaran

```
[
  { _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' }
]
```

Dalam contoh ini, `$redact` tahap memeriksa nilai `status` bidang di setiap dokumen. Jika "`status`dikirim" atau "diproses", dokumen disimpan ($$KEEP). Jika tidak, dokumen dipangkas ($$PRUNE).

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$redact` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Operator `$reduce` agregasi di Amazon DocumentDB digunakan untuk menerapkan fungsi dua argumen secara kumulatif ke elemen array untuk mengurangi array menjadi satu nilai. Operator ini sangat berguna untuk melakukan perhitungan atau transformasi kompleks pada data array dalam pipeline agregasi.

Parameter

- `input`: Array yang akan dikurangi.
- `initialValue`: Nilai awal yang akan digunakan dalam operasi reduksi.
- `in`: Ekspresi yang akan dievaluasi pada setiap elemen `input` array. Ekspresi ini harus mengembalikan nilai yang akan digunakan dalam iterasi reduksi berikutnya.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$reduce` operator untuk menghitung jumlah semua elemen dalam array.

Buat dokumen sampel

```
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] }
])
```

Contoh kueri

```
db.orders.aggregate([
  {
    $project: {
      total: {
        $reduce: {
```

```
          input: "$items",
          initialValue: 0,
          in: { $add: ["$$value", "$$this"] }
        }
      }
    }
  }
])
```

Keluaran

```
[
  { "_id": 1, "total": 15 },
  { "_id": 2, "total": 60 },
  { "_id": 3, "total": 80 },
  { "_id": 4, "total": 300 }
]
```

`$reduce`Operator iterasi atas `items` array, menambahkan setiap elemen ke `initialValue` 0. Hasilnya adalah jumlah semua elemen dalam array.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$reduce` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

Berikut adalah contoh penggunaan `$reduce` operator dalam Node.js aplikasi:

```
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

Berikut adalah contoh penggunaan `$reduce` operator dalam aplikasi 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

Baru dari versi 5.0.

Tidak didukung oleh cluster elastis.

`$regexFind`Operator di Amazon DocumentDB digunakan untuk melakukan pencocokan ekspresi reguler pada bidang string dalam dokumen. Ini memungkinkan Anda untuk mencari dan mengekstrak substring tertentu yang cocok dengan pola ekspresi reguler yang diberikan.

Parameter

- `input`: Bidang string atau ekspresi untuk mencari.
- `regex`: Pola ekspresi reguler untuk mencocokkan.
- `options`: (opsional) Objek yang menentukan parameter opsional untuk ekspresi reguler, seperti sensitivitas huruf kecil dan pencocokan multi-baris. Opsi yang didukung adalah `i` (case-insensitive) dan `m` (multi-line).

Contoh (MongoDB Shell)

Contoh berikut menunjukkan cara menggunakan `$regexFind` operator untuk mencari dokumen di mana `name` bidang cocok dengan pola ekspresi reguler tertentu.

Buat dokumen sampel

```
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" }
]);
```

Contoh kueri

```
db.users.aggregate([
```

```
  {
    $project: {
      names: {
        $regexFind: { input: '$name', regex: 'j', options: 'i' }
      }
    }
  },
  { $match: {names: {$ne: null}}}
])
```

Kueri ini akan mengembalikan semua dokumen di mana `name` bidang berisi huruf “j” (case-insensitive).

Keluaran

```
[
  { _id: 1, names: { match: 'J', idx: 0, captures: [] } }
]
```

Catatan: Jika kueri Anda menggunakan Amazon DocumentDB planner versi 1, Anda harus menggunakan petunjuk untuk menggunakan indeks. Tanpa petunjuk, kueri dapat melakukan pemindaian koleksi. Untuk memeriksa versi perencana dan mempelajari lebih lanjut tentang menggunakan petunjuk, lihat [Dokumentasi Perencana Kueri Amazon DocumentDB] (). https://docs.aws.amazon.com/documentdb/latest/devguide/query-planner.html

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$regexFind` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

Diperkenalkan pada 5.0

`$regexFindAll`Operator di Amazon DocumentDB digunakan untuk melakukan pencocokan ekspresi reguler pada bidang string dalam dokumen. Ini memungkinkan Anda untuk mencari dan mengekstrak substring tertentu yang cocok dengan pola ekspresi reguler yang diberikan, mengembalikan semua kecocokan ekspresi reguler.

Parameter

- `input`: Bidang string atau ekspresi untuk mencari.
- `regex`: Pola ekspresi reguler untuk mencocokkan.
- `options`: (opsional) Objek yang menentukan parameter opsional untuk ekspresi reguler, seperti sensitivitas huruf kecil dan pencocokan multi-baris. Opsi yang didukung adalah `i` (case-insensitive) dan `m` (multi-line).

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$regexFindAll` operator untuk mengekstrak semua urutan huruf dari lapangan. `email`

Buat dokumen sampel

```
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" }

]);
```

Contoh kueri

```
db.users.aggregate([
  {
    $project: {
      name: 1,
      emailMatches: {
        $regexFindAll: { input: '$email', regex: '[a-z]+', options: 'i' }
      }
    }
  }
])
```

Keluaran

```
[
```

```
  {
    _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: [] }
    ]
  }
]
```

Catatan: Jika kueri Anda menggunakan Amazon DocumentDB planner versi 1, Anda harus menggunakan petunjuk untuk menggunakan indeks. Tanpa petunjuk, kueri dapat melakukan pemindaian koleksi. Untuk memeriksa versi perencana dan mempelajari lebih lanjut tentang menggunakan petunjuk, lihat [Dokumentasi Perencana Kueri Amazon DocumentDB] (). https://docs.aws.amazon.com/documentdb/latest/devguide/query-planner.html

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$regexFindAll` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

Berikut adalah contoh penggunaan `$regexFind` operator dalam Node.js aplikasi:

```
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

Berikut adalah contoh penggunaan `$regexFind` operator dalam aplikasi 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

Baru dari versi 5.0. Tidak didukung oleh cluster elastis.

`$regexMatch`Operator di Amazon DocumentDB digunakan untuk melakukan pencocokan ekspresi reguler pada bidang string. Ia mengembalikan nilai boolean (`true`atau`false`) menunjukkan apakah string input cocok dengan pola yang ditentukan.

Parameter

- `input`: String untuk menguji terhadap ekspresi reguler.
- `regex`: Pola ekspresi reguler untuk mencocokkan.
- `options`: (Opsional) Bendera untuk mengubah perilaku ekspresi reguler, seperti pencocokan case-insensitive (`i`) atau pencocokan multiline (). `m`

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$regexMatch` operator untuk memeriksa apakah nama dimulai dengan huruf 'M'. Operator mengembalikan `true` atau `false` untuk setiap dokumen.

Buat dokumen sampel

```
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" },

]);
```

Contoh kueri

```
db.users.aggregate([
  {
    $project: {
      name: 1,
      startsWithM: {
        $regexMatch: {
          input: "$name",
          regex: "^M",
          options: "i"
        }
      }
    }
  }
]);
```

Keluaran

```
{ _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 }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$regexMatch` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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()
```

$ GantiSemua

Diperkenalkan pada 5.0

`$replaceAll`Operator di Amazon DocumentDB digunakan untuk mengganti semua kemunculan pola string tertentu dalam bidang dengan string baru. Operator ini dapat berguna untuk tugas-tugas seperti normalisasi data, pembersihan teks, dan manipulasi string.

Parameter

- `input`: Bidang atau ekspresi yang berisi string yang akan diganti.
- `find`: Pola string untuk mencari dan mengganti.
- `replacement`: String untuk mengganti kejadian yang cocok dengan.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan cara menggunakan `$replaceAll` operator dalam pipeline agregasi untuk mengganti semua kemunculan string “Chocolatier” dengan “Chocolate Co.” di bidang “BrandName” dari koleksi “produk”.

Buat dokumen sampel

```
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
  }
]);
```

Contoh kueri

```
db.products.aggregate([
  {
    $addFields: {
      "brandName": {
```

```
        $replaceAll: {
          input: "$brandName",
          find: "Chocolatier",
          replacement: "Chocolate Co."
        }
      }
    }
  }
])
```

Keluaran

```
[
  {
    _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
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$replaceAll` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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()
```

$ GantiSatu

Diperkenalkan pada 5.0

`$replaceOne`Operator di Amazon DocumentDB adalah operator ekspresi string yang digunakan dalam pipeline agregasi untuk menggantikan kemunculan pertama substring tertentu dalam string dengan string pengganti. Operator ini peka huruf besar/kecil dan hanya menggantikan kecocokan pertama yang ditemukan.

Parameter

- `input`: String (bidang) untuk melakukan pencarian.
- `find`: String untuk mencari di dalam input.
- `replacement`: String untuk menggantikan kemunculan pertama dari temuan di input (field).

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$replaceOne` operator dalam pipeline agregasi untuk mengganti substring dalam nama produk.

Buat dokumen sampel

```
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 }
]);
```

Contoh agregasi

```
db.products.aggregate([
  {
    $addFields: {
      standardizedName: {
        $replaceOne: {
          input: "$name",
          find: "Pack",
          replacement: "Package"
        }
      }
    }
  }
]);
```

Keluaran

Output menunjukkan bahwa hanya kemunculan pertama “Pack” di setiap nama produk diganti dengan “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"
  }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$replaceOne` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$replaceRoot`Operator digunakan untuk mengganti dokumen root dengan dokumen tertanam yang ditentukan. Ini berguna ketika Anda ingin mempromosikan dokumen bersarang ke tingkat atas atau merestrukturisasi output data Anda.

Parameter

- `newRoot`: Dokumen root baru yang akan menggantikan dokumen root yang ada.

Contoh (MongoDB Shell)

Contoh ini menunjukkan cara mengekstrak informasi alamat pengiriman dari pesanan pelanggan, yang berguna untuk menghasilkan label pengiriman atau daftar alamat.

Buat dokumen sampel

```
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 } ])
```

Contoh kueri

```
db.orders.aggregate([
  {
    $replaceRoot: {
      newRoot: "$shippingAddress"
    }
  }
])
```

Keluaran

```
{
```

```
    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'
  }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$replaceRoot` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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()
```

$ GantiDengan

Baru dari versi 8.0

Tidak didukung oleh cluster elastis.

Tahap `$replaceWith` agregasi di Amazon DocumentDB digunakan untuk mengganti dokumen input dengan dokumen baru. Semua bidang yang ada pada dokumen masukan, termasuk bidang _id, diganti dengan dokumen baru. `$replaceWith`biasanya digunakan untuk meratakan dokumen, atau mempromosikan dokumen tertanam ke tingkat atas.

Parameter

- `<replacement>`(wajib): Dokumen baru yang akan menggantikan dokumen yang ada.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan cara menggunakan `$replaceWith` operator untuk mengganti dokumen yang ada dalam koleksi Amazon DocumentDB.

Buat dokumen sampel

```
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 ]
  }
]);
```

Contoh kueri

```
db.restaurants.aggregate([
  { $replaceWith: {
      name: "$name",
      cuisine: "$cuisine",
      rating: { $avg: "$ratings" }
    }
  }
]);
```

Keluaran

```
[
  {
    name: 'Biryani Adda',
    cuisine: 'Indian',
    rating: 3.4
```

```
  },
  {
    name: 'The Burger Spot',
    cuisine: 'American',
    rating: 2.7777777777777777
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$replaceWith` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$reverseArray`Operator di Amazon DocumentDB digunakan untuk membalikkan elemen array dalam urutan yang ditentukan. Operator ini berguna ketika Anda perlu menyusun ulang elemen array dalam arah sebaliknya.

Parameter

- `expression`: Ekspresi array untuk membalikkan.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$reverseArray` operator untuk membalikkan urutan elemen dalam array.

Buat dokumen sampel

```
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]}
]);
```

Contoh kueri

```
db.miles.aggregate([
  {
    $project: {
      _id: 1,
      member_since: 1,
      credit_card: 1,
      reversed_flight_miles: { $reverseArray: "$flight_miles" }
    }
  }
]);
```

Keluaran

```
{ "_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 ] }
```

Dalam contoh ini, `$reverseArray` operator digunakan untuk membalikkan urutan `flight_miles` array. `reversed_flight_miles`Bidang yang dihasilkan dalam output menunjukkan elemen array dalam urutan terbalik.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$reverseArray` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

Berikut adalah contoh penggunaan `$reverseArray` operator dalam Node.js aplikasi:

```
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

Berikut adalah contoh penggunaan `$reverseArray` operator dalam aplikasi 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()
```

$rtrim

Baru dari versi 4.0.

Tidak didukung oleh cluster elastis.

`$rtrim`Operator di Amazon DocumentDB digunakan untuk menghapus karakter trailing dari string. Secara default, ini menghapus karakter spasi tambahan, tetapi Anda juga dapat menentukan satu set karakter yang akan dihapus dengan meneruskan argumen karakter.

Parameter

- `input`: String masukan dari mana untuk menghapus karakter spasi belakang.
- `chars`: (opsional) Untuk menghapus karakter tertentu.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$rtrim` operator untuk menghapus trailing karakter tertentu (“*”) dari string.

Buat dokumen sampel

```
db.collection.insert([
  { "name": "John Doe*  ", "age": 30 },
  { "name": "Jane Smith  ", "age": 25 },
  { "name": "Bob Johnson", "age": 35 }
]);
```

Contoh kueri

```
db.collection.aggregate([
  {
    $project: {
      _id: 0,
      name: { $rtrim: { input: "$name", chars: " *" } },
      age: 1
    }
  }
]);
```

Keluaran

```
[
  { age: 30, name: 'John Doe' },
  { age: 25, name: 'Jane Smith' },
  { age: 35, name: 'Bob Johnson' }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$rtrim` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

Tahap `$sample` agregasi di Amazon DocumentDB digunakan untuk memilih secara acak sejumlah dokumen tertentu dari koleksi. Ini berguna untuk tugas-tugas seperti analisis data, pengujian, dan menghasilkan sampel untuk diproses lebih lanjut.

Parameter

- `size`: Jumlah dokumen untuk dipilih secara acak.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$sample` panggung untuk secara acak memilih dua dokumen dari koleksi. `temp`

Buat dokumen sampel

```
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") }
]);
```

Contoh kueri

```
db.temp.aggregate([
   { $sample: { size: 2 } }
])
```

Keluaran

```
{ "_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") }
```

Seperti yang ditunjukkan hasil, 2 dari 10 dokumen diambil sampelnya secara acak. Anda sekarang dapat menggunakan dokumen-dokumen ini untuk menentukan rata-rata atau untuk melakukan min/max perhitungan.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$sample` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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()
```

$ pencarian

Baru dari versi 5.0.

`$search`Operator di Amazon DocumentDB digunakan untuk menyediakan kemampuan pencarian teks.

Parameter

Tidak ada

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$search` operator untuk melakukan query pencarian teks.

Buat dokumen sampel

```
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." }
]);
```

Contoh kueri

```
db.textcollection.find(
  { $text: { $search: "John" } }
);
```

Keluaran

```
[
  {
    _id: 1,
    name: 'John Doe',
    description: 'This is a sample document about John Doe.'
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$search` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$second`Operator di Amazon DocumentDB mengekstrak komponen detik dari tanggal atau stempel waktu. Ini digunakan untuk mengambil nilai detik dari bidang tanggal atau timestamp.

Parameter

- `expression`: Bidang tanggal atau stempel waktu untuk mengekstrak nilai detik dari. Ekspresi ini dapat berupa jalur bidang atau ekspresi valid apa pun yang menyelesaikan tanggal atau stempel waktu.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$second` operator untuk mengekstrak komponen detik dari bidang tanggal.

Buat dokumen sampel

```
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") }
]);
```

Contoh kueri

```
db.users.aggregate([{ $project: { name: 1, dobSeconds: { $second: "$dob" } } }])
```

Keluaran

```
[
  { "_id" : ObjectId("6089a9c306a829d1f8b456a1"), "name" : "John", "dobSeconds" : 45 },
  { "_id" : ObjectId("6089a9c306a829d1f8b456a2"), "name" : "Jane", "dobSeconds" : 59 },
  { "_id" : ObjectId("6089a9c306a829d1f8b456a3"), "name" : "Bob", "dobSeconds" : 0 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$second` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('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

Baru dari versi 8.0

Tidak didukung oleh cluster elastis.

Tahap `$set` agregasi di Amazon DocumentDB memungkinkan Anda menambahkan bidang baru atau memperbarui nilai bidang yang ada dalam dokumentasi selama pipeline agregasi.

Parameter

- `expression`: Ekspresi untuk mengevaluasi. Ini bisa berupa ekspresi agregasi yang valid, termasuk referensi bidang dan operasi aritmatika.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan tahap `$set` agregasi untuk menghitung total dengan mengalikan bidang dengan bidang. `quantity` `price`

Buat dokumen sampel

```
db.inventory.insertMany([
  { item: "pencil", quantity: 100, price: 0.24},
  { item: "pen", quantity: 204, price: 1.78 }
]);
```

Contoh agregasi

```
db.inventory.aggregate([
  {
    $set: {
      total: { $multiply: ["$quantity", "$price"] }
    }
  }
])
```

Keluaran

```
[
  {
    _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
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$set` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('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

Baru dari versi 4.0.

`$setDifference`Operator di Amazon DocumentDB digunakan untuk membandingkan dua set dan mengembalikan elemen yang ada di set pertama tetapi tidak di set kedua. Operator ini berguna untuk menemukan elemen unik antara dua set.

Parameter

- `firstSet`: Set pertama untuk membandingkan.
- `secondSet`: Set kedua untuk membandingkan.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$setDifference` operator untuk menemukan elemen unik antara dua set.

Buat dokumen sampel

```
db.collection.insertMany([
  { _id: 1, fruits: ["apple", "banana", "cherry", "date"] },
  { _id: 2, fruits: ["banana", "cherry", "date", "elderberry"] }
]);
```

Contoh kueri

```
db.collection.aggregate([
  {
    $project: {
      uniqueFruits: { $setDifference: ["$fruits", ["banana", "cherry", "date"]] }
    }
  }
]);
```

Keluaran

```
[
  { "_id": 1, "uniqueFruits": ["apple"] },
  { "_id": 2, "uniqueFruits": ["elderberry"] }
]
```

Kueri melakukan langkah-langkah berikut:

1. Ini menggunakan `$project` panggung untuk membuat bidang baru `uniqueFruits` untuk setiap dokumen.

2. `$setDifference`Operator membandingkan `fruits` array dengan array `["banana", "cherry", "date"]` dan mengembalikan elemen unik dalam `fruits` array.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$setDifference` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

Berikut adalah contoh cara menggunakan `$setDifference` operator dalam Node.js aplikasi:

```
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

Berikut adalah contoh cara menggunakan `$setDifference` operator dalam aplikasi 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

`$setEquals`Operator di Amazon DocumentDB digunakan untuk menentukan apakah dua set sama. Ini membandingkan dua array dan kembali `true` jika mengandung elemen berbeda yang sama, terlepas dari urutannya.

Parameter

- `expression1`: Array pertama untuk membandingkan.
- `expression2`: Array kedua untuk membandingkan.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$setEquals` operator untuk membandingkan dua set nilai.

Buat dokumen sampel

```
db.collection.insertMany([
  { _id: 1, fruits: ["apple", "banana", "cherry"] },
  { _id: 2, fruits: ["banana", "apple", "cherry"] },
```

```
  { _id: 3, fruits: ["apple", "banana", "orange"] }
])
```

Contoh kueri

```
db.collection.find({
  $expr: {
    $setEquals: ["$fruits", ["apple", "banana", "cherry"]]
  }
})
```

Keluaran

```
{ "_id" : 1, "fruits" : [ "apple", "banana", "cherry" ] }
{ "_id" : 2, "fruits" : [ "banana", "apple", "cherry" ] }
```

Query menggunakan `$setEquals` operator untuk membandingkan `fruits` bidang setiap dokumen dengan array`["apple", "banana", "cherry"]`. Dokumen di mana `fruits` bidang sama dengan array perbandingan dikembalikan.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$setEquals` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

`$setIntersection`Operator di Amazon DocumentDB digunakan untuk mengembalikan elemen umum antara dua atau lebih array. Operator ini sangat berguna saat bekerja dengan set data, memungkinkan Anda menemukan persimpangan beberapa set.

Parameter

- `array1`: Array pertama untuk berpotongan.
- `array2`: Array kedua untuk berpotongan.
- `arrayN`: (opsional) Array tambahan untuk berpotongan.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$setIntersection` operator untuk menemukan elemen umum antara dua array.

Buat dokumen sampel

```
db.collection.insertMany([
  { _id: 1, colors: ["red", "blue", "green"] },
  { _id: 2, colors: ["blue", "yellow", "orange"] },
  { _id: 3, colors: ["red", "green", "purple"] }
])
```

Contoh kueri

```
db.collection.aggregate([
  { $project: {
      _id: 1,
      commonColors: { $setIntersection: ["$colors", ["red", "blue", "green"]] }
    }
  }
])
```

Keluaran

```
[
  { "_id": 1, "commonColors": ["red", "blue", "green"] },
```

```
  { "_id": 2, "commonColors": ["blue"] },
  { "_id": 3, "commonColors": ["red", "green"] }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$setIntersection` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

`$setIsSubset`Operator di Amazon DocumentDB digunakan untuk menentukan apakah satu set nilai adalah subset dari himpunan lain. Hal ini berguna untuk melakukan perbandingan berbasis set dan operasi pada bidang array.

Parameter

- `field`: Bidang untuk menerapkan `$setIsSubset` operator ke.
- `set`: Set untuk membandingkan bidang terhadap.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$setIsSubset` operator untuk memeriksa apakah `tags` bidang adalah subset dari set yang ditentukan.

Buat dokumen sampel

```
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"] }
```

```
]);
```

Contoh kueri

```
db.products.find({
  $expr: { $setIsSubset: [["tag1", "tag2"], "$tags"] }
})
```

*Catatan: * `$setIsSubset` adalah operator agregasi dan tidak dapat digunakan secara langsung dalam kueri find (). Dalam contoh ini, `$expr` digunakan dengan `find()` untuk menjembatani kesenjangan antara operator query dan ekspresi agregasi.

Keluaran

```
[
  { "_id" : 1, "name" : "Product A", "tags" : [ "tag1", "tag2", "tag3" ] },
  { "_id" : 2, "name" : "Product B", "tags" : [ "tag1", "tag2" ] }
]
```

Query mengembalikan dokumen di mana `tags` bidang adalah bagian dari himpunan`["tag1", "tag2"]`.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$setIsSubset` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Operator `$setUnion` agregasi di Amazon DocumentDB digunakan untuk menggabungkan dua atau lebih set nilai dan mengembalikan satu set yang berisi semua elemen unik dari set input. Operator ini berguna ketika Anda perlu melakukan operasi berbasis set pada bidang array dalam dokumen Anda.

Parameter

- `expression1`: Ekspresi yang menyelesaikan ke array.
- `expression2`: Ekspresi yang menyelesaikan ke array.
- `expressionN`: Ekspresi tambahan yang menyelesaikan array (opsional).

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$setUnion` operator untuk menggabungkan elemen unik dari dua bidang array dalam koleksi.

Buat dokumen sampel

```
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"] }
]);
```

Contoh kueri

```
db.users.aggregate([
  {
    $project: {
      name: 1,
      allInterests: { $setUnion: ["$hobbies", "$skills"] }
    }
  }
]);
```

Keluaran

```
[
  { "_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" ] }
]
```

Dalam contoh ini, `$setUnion` operator digunakan untuk menggabungkan elemen unik dari bidang `hobbies` dan `skills` array untuk setiap dokumen pengguna. `allInterests`Bidang yang dihasilkan berisi penyatuan semua hobi dan keterampilan unik untuk setiap pengguna.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$setUnion` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('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()
```

$skip

Di Amazon DocumentDB, `$skip` operator digunakan untuk mengimbangi titik awal hasil kueri, memungkinkan Anda untuk mengambil subset tertentu dari dokumen yang cocok. Ini sangat berguna dalam skenario pagination, di mana Anda ingin mengambil halaman hasil berikutnya.

Parameter

- `skip`: Jumlah dokumen yang harus dilewati sebelum mengembalikan dokumen yang tersisa.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$skip` operator untuk mengambil halaman kedua hasil (dokumen 11-20) dari koleksi.

Buat dokumen sampel

```
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" }
]);
```

Contoh kueri

```
db.collection.find({}, { "name": 1 })
             .skip(10)
             .limit(10);
```

Keluaran

```
[
  { "_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" }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$skip` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

Operator `$slice` agregasi memungkinkan Anda mengembalikan subset array dengan melintasi array dari awal atau akhir array. Ini digunakan untuk menampilkan sejumlah item terbatas dari bidang array, seperti item N atas atau bawah.

Parameter

- `array`: Bidang array yang akan diiris.
- n: Sebuah integer yang menentukan jumlah elemen untuk kembali. Nilai positif dimulai dari awal array, sedangkan nilai negatif dimulai dari akhir array.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan cara menggunakan `$slice` untuk mengembalikan dua permen favorit pertama untuk setiap koki.

Buat dokumen sampel

```
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" ] }
]);
```

Contoh kueri

```
db.sweets.aggregate([
  { $project: { _id: 0, name: 1, topTwoFavorites: { $slice: [ "$favorites", 2 ] } } }
]);
```

Keluaran

```
[
```

```
  { name: 'Alvin', topTwoFavorites: [ 'chocolate', 'cake' ] },
  { name: 'Tom', topTwoFavorites: [ 'donuts', 'pudding' ] },
  { name: 'Jessica', topTwoFavorites: [ 'fudge', 'smores' ] },
  { name: 'Rachel', topTwoFavorites: [ 'ice cream' ] }
]
```

Dalam contoh ini, `$slice` operator digunakan untuk mengekstrak dua elemen pertama dari `favorites` array untuk setiap dokumen.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$slice` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

`$size`Operator digunakan untuk mengembalikan jumlah item dalam bidang array. Ini dapat digunakan untuk menentukan jumlah elemen dalam array yang disimpan dalam dokumen.

Parameter

- `field`: Jalur bidang yang ukuran arry-nya ingin Anda kembalikan.

Contoh (MongoDB Shell)

Contoh ini menunjukkan cara menggunakan `$size` operator untuk mengembalikan jumlah tim yang diikuti setiap pengguna.

Buat dokumen sampel

```
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"] }
]);
```

Contoh kueri

```
db.profiles.aggregate([
  {
    $project: {
```

```
      _id: 0,
      name: 1,
      "numberOfTeams": { $size: "$teams" }
    }
  }
])
```

Keluaran

```
[
  { name: 'John Doe', numberOfTeams: 3 },
  { name: 'Jane Smith', numberOfTeams: 2 },
  { name: 'Bob Johnson', numberOfTeams: 3 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$size` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

Tahap `$sort` agregasi memesan dokumen dalam pipeline berdasarkan nilai bidang yang ditentukan. Dokumen disusun dalam urutan naik atau turun sesuai dengan kriteria pengurutan yang disediakan.

Parameter

- `field`: Nama bidang untuk mengurutkan berdasarkan.
- `order`: Gunakan 1 untuk urutan menaik atau `-1` untuk urutan menurun.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$sort` panggung untuk memesan produk berdasarkan harga dalam urutan menurun.

Buat dokumen sampel

```
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 }
]);
```

Contoh kueri

```
db.products.aggregate([
  { $sort: { price: -1 } }
]);
```

Keluaran

```
[
  { _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 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan tahap `$sort` agregasi, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
```

```
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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()
```

$split

Operator `$split` agregasi di Amazon DocumentDB digunakan untuk membagi string menjadi array substring, berdasarkan pembatas tertentu. Ini dapat berguna untuk mengurai bidang string yang kompleks dan mengekstraksi komponen individual untuk diproses lebih lanjut.

Parameter

- `string`: String yang akan dibagi.
- `delimiter`: Karakter atau string yang digunakan untuk membagi string input.

Contoh (MongoDB Shell)

Dalam contoh ini, kita gunakan `$split` untuk memisahkan komponen bidang “Desk” ke dalam array, sehingga lebih mudah untuk memproses data.

Buat dokumen sampel

```
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" }
]);
```

Contoh kueri

```
db.people.aggregate([
  { $project: { parts: { $split: ["$Desk", "-"] } } }
]);
```

Keluaran

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

Output dari `$split` membuat array yang dapat digunakan dalam aplikasi untuk menampilkan informasi bagi karyawan.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$split` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('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

Baru dari versi 4.0.

`$sqrt`Operator di Amazon DocumentDB digunakan untuk menghitung akar kuadrat dari suatu angka.

Parameter

- `expression`: Argumen dapat berupa ekspresi yang valid selama diselesaikan menjadi angka non-negatif.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$sqrt` operator untuk menghitung akar kuadrat dari angka.

Buat dokumen sampel

```
db.numbers.insertMany([
  { "_id": 1, "number": 16 },
  { "_id": 2, "number": 36 },
  { "_id": 3, "number": 64 }
]);
```

Contoh kueri

```
db.numbers.aggregate([
  { $project: {
    "_id": 1,
    "square_root": { $sqrt: "$number" }
  }}
]);
```

Keluaran

```
[
  { _id: 1, square_root: 4 },
  { _id: 2, square_root: 6 },
  { _id: 3, square_root: 8 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$sqrt` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function 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()
```

$ str LenBytes

`$strLenBytes`Operator di Amazon DocumentDB digunakan untuk menentukan panjang string dalam byte. Ini berguna ketika Anda perlu memahami ukuran penyimpanan bidang string, terutama ketika berhadapan dengan karakter Unicode yang mungkin menggunakan lebih dari satu byte per karakter.

Parameter

- `expression`: Ekspresi string untuk menghitung panjang.

Contoh (MongoDB Shell)

Contoh ini menunjukkan bagaimana menggunakan `$strLenBytes` operator untuk menghitung panjang bidang string dalam byte.

Buat dokumen sampel

```
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" }
]);
```

Contoh kueri

```
db.people.aggregate([
  {
    $project: {
      "Desk": 1,
      "length": { $strLenBytes: "$Desk" }
    }
  }
])
```

Keluaran

```
{ "_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 }
```

Perhatikan bahwa panjang string “Düsseldorf-BVV-021” adalah 19 byte, yang berbeda dari jumlah titik kode (18) karena karakter Unicode “Ü” menempati 2 byte.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$strLenBytes` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

`$strLenCP`Operator di Amazon DocumentDB digunakan untuk menentukan panjang ekspresi string di titik kode (karakter Unicode). Ini berguna ketika Anda perlu mengetahui jumlah karakter dalam string, bukan jumlah byte.

Parameter

- `expression`: Ekspresi string untuk mengembalikan panjang dalam poin kode.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$strLenCP` operator untuk menentukan panjang string dengan karakter Unicode.

Buat dokumen sampel

```
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" }
])
```

Contoh kueri

```
db.people.aggregate([
  {
    $project: {
      "Desk": 1,
      "length": { $strLenCP: "$Desk" }
    }
  }
```

```
])
```

Keluaran

```
{ "_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 }
```

Perhatikan perbedaan pengukuran panjang untuk string “Düsseldorf-BVV-021”, yang berisi karakter Unicode (Ü). `$strLenCP`Operator menghitung jumlah karakter Unicode dengan benar, sedangkan `$strLenBytes` operator menghitung jumlah byte.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$strLenCP` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

Berikut adalah contoh penggunaan `$strLenCP` operator dalam Node.js aplikasi dengan 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

Berikut adalah contoh penggunaan `$strLenCP` operator dalam aplikasi Python dengan driver: PyMongo

```
from pymongo import MongoClient

def example():
    client = 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

`$strcasecmp`Operator di Amazon DocumentDB melakukan perbandingan case-insensitive antara dua string. Ia mengembalikan nilai integer yang menunjukkan perbandingan leksikografi dari dua string input, mengabaikan perbedaan kasus.

Parameter

- `string1`: String pertama untuk membandingkan.

- `string2`: String kedua untuk membandingkan.

Contoh (MongoDB Shell)

Contoh ini menunjukkan cara menggunakan `$strcasecmp` operator untuk membandingkan string lokasi meja dalam `people` koleksi, mengabaikan perbedaan kasus.

Buat dokumen sampel

```
db.people.insertMany([
  { "_id": 1, "Desk": "mke233-wi" },
  { "_id": 2, "Desk": "MKE233-WI" },
  { "_id": 3, "Desk": "mke233-wi" }
]);
```

Contoh kueri

```
db.people.aggregate([
  {
    $project: {
      item: 1,
      compare: { $strcasecmp: ["$Desk", "mke233-wi"] }
    }
  }
]);
```

Keluaran

```
{ "_id" : 1, "compare" : 0 }
{ "_id" : 2, "compare" : 0 }
{ "_id" : 3, "compare" : 0 }
```

Output menunjukkan bahwa perbandingan antara `"Desk"` bidang dan string `"mke233-wi"` kembali `0` untuk ketiga dokumen, menunjukkan bahwa string sama ketika kasus diabaikan.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$strcasecmp` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

`$substr`Operator di Amazon DocumentDB digunakan untuk mengekstrak substring dari string yang diberikan. Ini sangat berguna ketika Anda perlu mendefinisikan substring berdasarkan berbagai karakter, bukan rentang byte. Ini sangat penting ketika berhadapan dengan string Unicode, di mana jumlah byte yang digunakan untuk mewakili karakter dapat bervariasi.

Parameter

- `string`: String input dari mana untuk mengekstrak substring.
- `start`: Posisi awal (berbasis nol) dari substring yang akan diekstraksi. Bisa berupa ekspresi integer non-negatif.
- `length`: Jumlah karakter dalam substring yang diekstraksi. Bisa berupa ekspresi integer non-negatif.

Contoh (MongoDB Shell)

Dalam contoh ini, kami akan mendemonstrasikan penggunaan `$substr` untuk mengekstrak singkatan negara dari lokasi meja karyawan.

Buat dokumen sampel

```
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" }
]);
```

Contoh kueri

```
db.people.aggregate([
  {
    $project: {
      "state": { $substr: ["$Desk", 12, 3] }
    }
  }
])
```

Keluaran

```
{ "_id": 1, "state": "NRW" },
{ "_id": 2, "state": "HBB" },
{ "_id": 3, "state": "SHH" },
{ "_id": 4, "state": "BBB" }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$substr` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require("mongodb");

async function example() {
  const client = await MongoClient.connect("mongodb://
<username>:<password>@<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

`$substrBytes`Operator di Amazon DocumentDB digunakan untuk mengekstrak substring dari string berdasarkan rentang byte tertentu. Operator ini berguna ketika Anda perlu mengekstrak substring dari string dan jumlah byte yang diperlukan untuk mewakili setiap karakter dalam string adalah penting.

Tidak seperti`$substrCP`, yang beroperasi pada jumlah titik kode Unicode, `$substrBytes` beroperasi pada jumlah byte yang diperlukan untuk mewakili karakter dalam string. Ini bisa sangat berguna ketika bekerja dengan string yang berisi karakter non-ASCII, karena karakter ini mungkin memerlukan lebih dari satu byte untuk mewakili.

*Catatan: * `$substr` telah usang sejak versi 3.4. `$substr`sekarang alias untuk`$substrBytes`.

Parameter

- `string`: String input dari mana substring akan diekstraksi.
- `startByte`: Posisi byte awal berbasis nol dari substring yang akan diekstraksi. Nilai negatif dapat digunakan untuk menentukan posisi dari akhir string.
- `length`: Jumlah byte dalam substring yang akan diekstraksi.

Contoh (MongoDB Shell)

Dalam contoh ini, kita akan menggunakan `$substrBytes` untuk mengekstrak substring dari string yang berisi karakter non-ASCII.

Buat dokumen sampel

```
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" }
]);
```

Contoh kueri

```
db.people.aggregate([
  {
    $project: {
      "state": { $substrBytes: [ "$Desk", 12, 3] }
    }
  }
])
```

Keluaran

```
{ "_id": 1, "state": "NRW" },
{ "_id": 2, "state": "HBB" },
{ "_id": 3, "state": "SHH" },
{ "_id": 4, "state": "BBB" }
```

Dalam contoh ini, kita gunakan `$substrBytes` untuk mengekstrak substring 3-byte mulai dari byte ke-12 bidang. `Desk` Ini memungkinkan kita untuk mengekstrak singkatan status 2 karakter, meskipun string mungkin berisi karakter non-ASCII.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$substrBytes` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('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

`$substrCP`Operator di Amazon DocumentDB digunakan untuk mengekstrak substring dari string, di mana substring ditentukan sebagai rentang UTF-8 titik kode (CP). Operator ini sangat berguna ketika bekerja dengan string Unicode, karena memungkinkan Anda untuk mengekstrak substring tanpa harus khawatir tentang representasi byte yang mendasarinya dari karakter.

Berbeda dengan `$substrBytes` operator, yang beroperasi pada posisi byte, `$substrCP` operator bekerja dengan posisi titik kode. Ini membuatnya lebih mudah untuk bekerja dengan string yang berisi karakter non-ASCII, karena jumlah titik kode mungkin tidak cocok dengan jumlah byte atau karakter.

Parameter

- `string`: String input dari mana untuk mengekstrak substring.
- `start`: Posisi titik kode awal (berbasis nol) dari mana untuk mengekstrak substring.
- `length`: Jumlah poin kode untuk diekstrak.

Contoh (MongoDB Shell)

Dalam contoh ini, kita akan menggunakan `$substrCP` operator untuk mengekstrak singkatan status dari string yang berisi lokasi meja karyawan.

Buat dokumen sampel

```
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" }
]);
```

Contoh kueri

```
db.people.aggregate([
  {
    $project: {
      "state": { $substrCP: ["$Desk", 25, 2] }
    }
  }
]);
```

Keluaran

```
{ "_id" : 1, "state" : "MN" }
{ "_id" : 2, "state" : "LA" }
{ "_id" : 3, "state" : "CA" }
```

Dalam contoh ini, kita tahu bahwa singkatan status dimulai pada titik kode ke-25 di `Desk` bidang dan panjangnya 2 poin kode. Dengan menggunakan `$substrCP` operator, kita dapat mengekstrak singkatan status tanpa harus khawatir tentang representasi byte yang mendasarinya dari string.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$substrCP` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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()
```

Dalam contoh Node.js dan Python, kami menggunakan `$substrCP` operator untuk mengekstrak singkatan status dari `Desk` bidang, mirip dengan contoh MongoDB Shell.

$subtract

`$subtract`Operator di Amazon DocumentDB digunakan untuk mengurangi nilai. Ini dapat digunakan untuk mengurangi tanggal, angka, atau kombinasi keduanya. Operator ini berguna untuk menghitung selisih antara dua tanggal atau mengurangi nilai dari angka.

Parameter

- `expression1`: Nilai pertama yang akan dikurangi.
- `expression2`: Nilai kedua yang akan dikurangi dari. `<expression1>`

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$subtract` operator untuk menghitung perbedaan antara dua tanggal.

Buat dokumen sampel

```
db.dates.insert([
  {
  "_id": 1,
  "startDate": ISODate("2023-01-01T00:00:00Z"),
  "endDate": ISODate("2023-01-05T12:00:00Z")
  }
]);
```

Contoh kueri

```
db.dates.aggregate([
  {
    $project: {
      _id: 1,
      durationDays: {
        $divide: [
          { $subtract: ["$endDate", "$startDate"] },
          1000 * 60 * 60 * 24  // milliseconds in a day
        ]
      }
    }
  }
```

```
]);
```

Keluaran

```
[ { _id: 1, durationDays: 4.5 } ]
```

Dalam contoh ini, `$subtract` operator digunakan untuk menghitung perbedaan antara `$endDate` dan `$startDate` dalam hari.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$subtract` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function 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

`$sum`Operator di Amazon DocumentDB mengembalikan jumlah ekspresi yang ditentukan untuk setiap dokumen dalam grup. Ini adalah operator akumulator grup yang biasanya digunakan dalam tahap $group dari pipa agregasi untuk melakukan perhitungan penjumlahan.

Parameter

- `expression`: Ekspresi numerik untuk jumlah. Ini bisa berupa jalur lapangan, ekspresi, atau konstanta.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$sum` operator untuk menghitung total penjualan untuk setiap produk.

Buat dokumen sampel

```
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 }
]);
```

Contoh kueri

```
db.sales.aggregate([
  { $group: {
```

```
      _id: "$product",
      totalSales: { $sum: { $multiply: [ "$price", "$quantity" ] } }
    }}
]);
```

Keluaran

```
[
  { "_id": "abc", "totalSales": 50 },
  { "_id": "xyz", "totalSales": 120 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$sum` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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('sales');

    const result = await collection.aggregate([
      { $group: {
          _id: "$product",
          totalSales: { $sum: { $multiply: [ "$price", "$quantity" ] } }
        }}
    ]).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.sales

        result = list(collection.aggregate([
            { '$group': {
                '_id': '$product',
                'totalSales': { '$sum': { '$multiply': [ '$price', '$quantity' ] } }
            }}
        ]))

        pprint(result)

    except Exception as e:
        print(f"An error occurred: {e}")

    finally:
        if client:
            client.close()

example()
```

$switch

Baru dari versi 4.0.

Tidak didukung oleh cluster elastis.

`$switch`Operator adalah operator ekspresi bersyarat di Amazon DocumentDB yang memungkinkan Anda mengevaluasi daftar ekspresi kasus dan mengembalikan nilai kasus pertama yang mengevaluasi ke true, atau nilai default jika tidak ada ekspresi kasus yang benar.

Parameter

- `branches`: Sebuah array dokumen, yang masing-masing memiliki bidang kasus yang berisi ekspresi boolean untuk mengevaluasi, dan kemudian bidang yang berisi nilai untuk kembali jika ekspresi kasus benar.
- `default`: (opsional) Nilai yang akan dikembalikan jika tidak ada ekspresi kasus yang benar.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$switch` operator untuk menentukan biaya pengiriman pesanan berdasarkan total pesanan.

Buat dokumen sampel

```
db.orders.insertMany([
  { _id: 1, total: 50 },
  { _id: 2, total: 150 },
  { _id: 3, total: 250 }
]);
```

Contoh kueri

```
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
        }
      }
    }
  }
])
```

Keluaran

```
[
  {
    "_id": 1,
    "total": 50,
    "shippingCost": 5
  },
  {
    "_id": 2,
    "total": 150,
    "shippingCost": 10
  },
  {
    "_id": 3,
    "total": 250,
    "shippingCost": 15
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$switch` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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()
```

$toBool

`$toBool`Operator di Amazon DocumentDB mengonversi ekspresi ke nilai boolean.

Parameter

- `expression`: Ekspresi yang akan dikonversi ke nilai boolean.

Catatan: Setiap string dikonversi ke`true`.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$toBool` operator untuk menormalkan nilai status perangkat dari tipe data yang berbeda.

Buat dokumen sampel

```
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 }
]);
```

Contoh kueri

```
db.deviceStates.aggregate([
  {
    $project: {
      _id: 1,
      deviceId: 1,
      isActive: { $toBool: "$status" }
    }
  }
]);
```

Keluaran

```
[
  { "_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 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$toBool` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

Baru dari versi 4.0

Operator `$toDate` agregasi di Amazon DocumentDB digunakan untuk mengonversi string tanggal atau tanggal dan waktu ke jenis Tanggal BSON. Ini adalah operasi terbalik dari `$dateToString` operator.

Parameter

- `dateString`: Sebuah representasi string dari tanggal atau tanggal dan waktu yang akan dikonversi ke tipe Tanggal BSON.
- `format`: (opsional) Sebuah string yang menentukan format. `dateString` Jika tidak disediakan, operator akan mencoba mengurai `dateString` dalam berbagai format tanggal dan waktu standar.
- `timezone`: (opsional) String yang mewakili zona waktu yang akan digunakan untuk konversi. Jika tidak disediakan, zona waktu lokal digunakan.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$toDate` operator untuk mengkonversi string tanggal ke tipe BSON Date.

Buat dokumen sampel

```
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" }
]);
```

Contoh kueri

```
db.events.aggregate([
  {
    $project: {
      eventName: 1,
      eventTimeDate: { $toDate: "$eventTime" }
    }
  }
]);
```

Keluaran

```
[
  {
    "_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")
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$toDate` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Baru dari versi 4.0

`$toDecimal`Operator di Amazon DocumentDB digunakan untuk mengonversi nilai ke tipe data Decimal128. Ini berguna ketika Anda perlu melakukan aritmatika desimal yang tepat atau menangani nilai desimal besar yang tidak dapat direpresentasikan secara akurat menggunakan tipe data ganda.

Parameter

- `expression`: Ekspresi untuk mengkonversi ke tipe data Decimal128.

Contoh (MongoDB Shell)

Contoh ini menunjukkan bagaimana menggunakan `$toDecimal` operator untuk mengkonversi nilai string ke tipe data Decimal128.

Buat dokumen sampel

```
db.numbers.insertOne({ _id: 1, value: "3.14" });
db.numbers.insertOne({ _id: 2, value: "2.71" });
```

Contoh kueri

```
db.numbers.aggregate([
  { $project: {
    _id: 1,
    decimalValue: { $toDecimal: "$value" }
  }}
])
```

Keluaran

```
[
  { "_id" : 1, "decimalValue" : Decimal128("3.14") },
  { "_id" : 2, "decimalValue" : Decimal128("2.71") }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$toDecimal` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Baru dari versi 4.0

`$toDouble`Operator di Amazon DocumentDB digunakan untuk mengonversi nilai menjadi nomor floating-point 64-bit presisi ganda. Ini dapat berguna ketika Anda perlu melakukan operasi aritmatika pada nilai-nilai yang awalnya tidak dalam format numerik.

Parameter

`<expression>`: Ekspresi untuk mengkonversi ke nilai ganda. Ini bisa berupa ekspresi valid yang menyelesaikan nilai numerik, string, atau boolean.

Contoh (MongoDB Shell)

Contoh ini menunjukkan bagaimana menggunakan `$toDouble` operator untuk mengkonversi nilai string ke nilai numerik untuk tujuan melakukan perhitungan matematis.

Buat dokumen sampel

```
db.numbers.insertMany([
  { _id: 1, value: "10.5" },
  { _id: 2, value: "20.25" },
  { _id: 3, value: "7" }
])
```

Contoh kueri

```
db.numbers.aggregate([
  {
    $project: {
      _id: 1,
      value: 1,
      double_value: { $toDouble: "$value" },
      double_plus_five: { $add: [{ $toDouble: "$value" }, 5] }
    }
  }
])
```

Keluaran

```
[
  { "_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 }
```

```
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$toDouble` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Baru dari versi 4.0

`$toInt`Operator di Amazon DocumentDB digunakan untuk mengonversi nilai input ke tipe data integer. Operator ini berguna ketika Anda perlu memastikan bahwa bidang atau ekspresi direpresentasikan sebagai bilangan bulat, yang dapat menjadi penting untuk operasi tertentu atau tugas pemrosesan data.

Parameter

- `expression`: Ekspresi yang akan dikonversi ke integer.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$toInt` operator untuk mengkonversi nilai string ke integer.

Buat dokumen sampel

```
db.numbers.insertMany([
```

```
  { "name": "one", "value": "1" },
  { "name": "hundred", "value": "100" }
]);
```

Contoh kueri

```
db.numbers.aggregate([
  { $project: {
    "_id": 0,
    "name": 1,
    "intValue": { $toInt: "$value" }
  }}
]);
```

Keluaran

```
{ "name": "one", "intValue": 1 }
{ "name": "hundred", "intValue": 100 }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$toInt` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Baru dari versi 4.0

`$toLong`Operator di Amazon DocumentDB digunakan untuk mengonversi nilai menjadi tipe data integer (panjang) 64-bit. Ini dapat berguna ketika Anda perlu melakukan operasi aritmatika atau perbandingan pada nilai numerik yang dapat disimpan sebagai string atau tipe data lainnya.

Parameter

- `expression`: Ekspresi untuk mengkonversi ke integer 64-bit.

Contoh (MongoDB Shell)

Contoh ini menunjukkan bagaimana menggunakan `$toLong` operator untuk mengkonversi nilai string ke integer 64-bit.

Buat dokumen sampel

```
db.numbers.insertMany([
  { _id: 1, value: "42" },
  { _id: 3, value: "9223372036854775807" }
]);
```

Contoh kueri

```
db.numbers.aggregate([
  {
    $project: {
      _id: 1,
      longValue: { $toLong: "$value" }
    }
  }
])
```

Keluaran

```
[
  { "_id" : 1, "longValue" : 42 },
  { "_id" : 3, "longValue" : 9223372036854775807 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$toLong` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
```

```
  const client = await MongoClient.connect('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

`$toLower`Operator di Amazon DocumentDB digunakan untuk mengonversi string ke huruf kecil.

Parameter

- `expression`: Ekspresi string untuk mengkonversi ke huruf kecil.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$toLower` operator untuk mengkonversi `Desk` bidang ke huruf kecil.

Buat dokumen sampel

```
db.locations.insertMany([
  { "_id": 1, "Desk": "Düsseldorf-BVV-021" },
  { "_id": 2, "Desk": "Munich-HGG-32a" }
]);
```

Contoh kueri

```
db.locations.aggregate([
  { $project: { item: { $toLower: "$Desk" } } }
]);
```

Keluaran

```
{ "_id" : 1, "item" : "düsseldorf-bvv-021" }
{ "_id" : 2, "item" : "munich-hgg-32a" }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$toLower` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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()
```

$ untuk ObjectId

Baru dari versi 4.0

`$toObjectId`Operator di Amazon DocumentDB digunakan untuk mengonversi representasi string dari tipe data aktual ke tipe ObjectId data ObjectId aktual. Hal ini dapat berguna ketika bekerja dengan data yang telah disimpan sebagai representasi string ObjectIds, karena memungkinkan Anda untuk melakukan operasi yang memerlukan tipe ObjectId data.

Parameter

- `expression`: Ekspresi string yang mewakili valid ObjectId.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$toObjectId` operator untuk mengkonversi representasi string dari tipe ObjectId data. ObjectId

Buat dokumen sampel

```
db.employees.insertMany([
  { _id: 1, empId:"64e5f8886218c620cf0e8f8a", name: "Carol Smith", employeeId:
 "c720a" },
  { _id: 2, empId:"64e5f94e6218c620cf0e8f8c", name: "Bill Taylor", employeeId:
 "c721a" }
]);
```

Contoh kueri

```
db.employees.aggregate([
  { $project: {
    "empIdAsObjectId": {$toObjectId: "$empId"}}
  }
]);
```

Keluaran

```
[
  { _id: 1, empIdAsObjectId: ObjectId('64e5f8886218c620cf0e8f8a') },
  { _id: 2, empIdAsObjectId: ObjectId('64e5f94e6218c620cf0e8f8c') }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$toObjectId` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Baru dari versi 4.0

`$toString`Operator di Amazon DocumentDB digunakan untuk mengonversi nilai jenis apa pun (kecuali null) menjadi representasi string. Ini dapat berguna ketika Anda perlu melakukan operasi string pada nilai-nilai yang awalnya tidak dalam format string.

Parameter

- `expression`: Ekspresi untuk mengkonversi ke string.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$toString` operator untuk mengkonversi nilai numerik untuk string.

Buat dokumen sampel

```
db.numbers.insertMany([
  { "_id": 1, "value": 42 },
  { "_id": 2, "value": 3.14 }
]);
```

Contoh kueri

```
db.numbers.aggregate([
  { $project: {
    _id: 1,
    valueAsString: { $toString: "$value" }
  }}
]);
```

Keluaran

```
{ "_id": 1, "valueAsString": "42" }
{ "_id": 2, "valueAsString": "3.14" }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$toString` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

`$toUpper`Operator di Amazon DocumentDB digunakan untuk mengonversi string ke huruf besar.

Parameter

- `expression`: Ekspresi string untuk mengkonversi ke huruf besar.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$toUpper` operator untuk mengkonversi `Desk` bidang ke huruf besar.

Buat dokumen sampel

```
db.locations.insertMany([
  { "_id": 1, "Desk": "düsseldorf-bvv-021" },
  { "_id": 2, "Desk": "munich-hgg-32a" }
]);
```

Contoh kueri

```
db.locations.aggregate([
  { $project: { item: { $toUpper: "$Desk" } } }
]);
```

Keluaran

```
{ "_id" : 1, "item" : "DüSSELDORF-BVV-021" }
{ "_id" : 2, "item" : "MUNICH-HGG-32A" }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$toUpper` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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()
```

$trim

Baru dari versi 4.0

`$trim`Operator di Amazon DocumentDB digunakan untuk menghapus karakter spasi putih and/or terdepan dari string.

Parameter

- `input`: Ekspresi string untuk memangkas.
- `chars`: (opsional) Menentukan karakter untuk memangkas dari awal dan akhir input, defaultnya adalah spasi putih.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan cara menggunakan `$trim` operator untuk menghapus spasi putih depan dan belakang dari string.

Buat dokumen sampel

```
db.people.insertMany([
  { "name": "   John Doe   " },
  { "name": "   Bob Johnson   " }
])
```

Contoh kueri

```
db.people.aggregate([
  { $project: {
```

```
    "name": { $trim: {input: "$name"}}
  }}
])
```

Keluaran

```
[
  { "name": "John Doe" },
  { "name": "Bob Johnson" }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$trim` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Operator `$type` agregasi mengembalikan tipe data BSON dari bidang tertentu. Ini berguna untuk mengidentifikasi tipe data nilai bidang selama operasi agregasi.

Parameter

- `expression`: Bidang atau ekspresi yang tipenya akan dikembalikan.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$type` operator untuk mengidentifikasi tipe data bidang harga untuk setiap produk.

Buat dokumen sampel

```
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 }
]);
```

Contoh kueri

```
db.inventory.aggregate([
  {
    $project: {
      item: 1,
      price: 1,
      priceType: { $type: "$price" }
    }
  }
]);
```

Keluaran

```
[
  { _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' }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan operator `$type` agregasi, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Baru dari versi 8.0

Tidak didukung oleh cluster elastis.

Tahap `$unset` agregasi di Amazon DocumentDB memungkinkan Anda menghapus bidang dari dokumen.

Parameter

- `expression`: Nama bidang atau daftar beberapa nama bidang.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan tahap `$unset` agregasi untuk menghapus bidang. `price`

Buat dokumen sampel

```
db.inventory.insertMany([
  { item: "pencil", quantity: 100, price: 0.24},
  { item: "pen", quantity: 204, price: 1.78 }
]);
```

Contoh agregasi

```
db.inventory.aggregate([
  {
    $unset: "price"
  }
])
```

Keluaran

```
[
  {
    _id: ObjectId('69248951d66dcae121d2950d'),
    item: 'pencil',
    quantity: 100
  },
  {
    _id: ObjectId('69248951d66dcae121d2950e'),
    item: 'pen',
    quantity: 204
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$unset` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('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

$unwindOperator digunakan untuk mendekonstruksi bidang array dari dokumen input untuk output dokumen untuk setiap elemen. Ini dapat berguna ketika Anda ingin melakukan operasi pada elemen individual dari array, seperti memfilter, menyortir, atau mengubah data.

Parameter

- `path`: Jalur ke bidang array yang akan dilepas.
- `includeArrayIndex`: (opsional) Menentukan nama bidang baru untuk menahan indeks elemen array.
- `preserveNullAndEmptyArrays`: (opsional) Menentukan apakah operasi menyimpan dokumen asli ketika bidang array adalah null atau array kosong.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan $unwind operator untuk mendekonstruksi bidang array dan melakukan operasi lebih lanjut pada elemen individu.

Buat dokumen sampel

```
db.people.insertMany([
  { _id: 1, name: "jon", hobbies: ["painting", "dancing", "singing"] },
  { _id: 2, name: "jane", hobbies: ["reading", "swimming"] },
  { _id: 3, name: "jack", hobbies: [] }
])
```

Contoh kueri

```
db.people.aggregate([
  { $unwind: "$hobbies" }
```

```
])
```

Keluaran

```
[
  { _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' }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$unwind` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Baru dari versi 8.0

Tidak didukung oleh cluster elastis.

`$vectorSearch`Operator di Amazon DocumentDB memungkinkan Anda melakukan pencarian vektor, metode yang digunakan dalam pembelajaran mesin untuk menemukan titik data serupa dengan membandingkan representasi vektornya menggunakan metrik jarak atau kesamaan. Kemampuan ini menggabungkan fleksibilitas dan kueri yang kaya dari database JSON-based dokumen dengan kekuatan pencarian vektor, memungkinkan Anda untuk membangun pembelajaran mesin dan kasus penggunaan AI generatif seperti pencarian semantik, rekomendasi produk, dan banyak lagi.

Parameter

- `<exact>`(opsional): Bendera yang menentukan apakah akan menjalankan pencarian Exact Nearest Neighbor (ENN) atau Perkiraan Tetangga Terdekat (ANN). Nilai dapat menjadi salah satu dari berikut ini:
- false - untuk menjalankan pencarian ANN
- true - untuk menjalankan pencarian ENN

Jika dihilangkan atau disetel ke false, `numCandidates` diperlukan.

```
- `<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.
```

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$vectorSearch` operator untuk menemukan deskripsi produk serupa berdasarkan representasi vektor mereka.

Buat dokumen sampel

```
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]
  }
]);
```

Buat indeks pencarian vektor

```
db.runCommand(
    {
        createIndexes: "products",
```

```
        indexes: [{
            key: {
                "description_vector": "vector"
            },
            vectorOptions: {
                type: "hnsw",
                dimensions: 3,
                similarity: "cosine",
                m: 16,
                efConstruction: 64
            },
            name: "description_index"
        }]
    }
);
```

Contoh kueri

```
db.products.aggregate([
  { $vectorSearch: {
      index: "description_index",
      limit: 2,
      numCandidates: 10,
      path: "description_vector",
      queryVector: [0.1, 0.2, 0.3]
    }
  }
]);
```

Keluaran

```
[
  {
    "_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 ]
```

```
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$vectorSearch` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$week`Operator di Amazon DocumentDB mengembalikan nomor minggu tanggal (0-53) berdasarkan standar ISO 8601. Jumlah minggu dihitung berdasarkan tahun dan hari dalam seminggu, dengan Senin sebagai hari pertama dalam seminggu.

Parameter

Tidak ada

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$week` operator untuk mengambil nomor minggu dari tanggal tertentu.

Buat dokumen sampel

```
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") }
]);
```

Contoh kueri

```
db.events.aggregate([
  { $project: {
    _id: 1,
    week: { $week: "$date" }
  }}
]);
```

Keluaran

```
[
  { "_id": 1, "week": 1 },
  { "_id": 2, "week": 2 },
  { "_id": 3, "week": 53 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan $week perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('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

`$year`Operator di Amazon DocumentDB mengekstrak komponen tahun dari tanggal atau stempel waktu.

Parameter

- `expression`: Ekspresi tanggal atau stempel waktu untuk mengekstrak komponen tahun.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan cara menggunakan `$year` operator untuk mengekstrak komponen tahun dari bidang tanggal.

Buat dokumen sampel

```
db.events.insertMany([
  { "_id": 1, "date": ISODate("2023-04-15T00:00:00Z") },
  { "_id": 3, "date": ISODate("2021-12-31T00:00:00Z") }
]);
```

Contoh kueri

```
db.events.aggregate([
  { $project: { year: { $year: "$date" } } }
]);
```

Keluaran

```
[
  { "_id": 1, "year": 2023 },
  { "_id": 3, "year": 2021 }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$year` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

`$zip`Operator di Amazon DocumentDB memungkinkan Anda untuk menggabungkan beberapa array ke dalam satu array tupel (pasangan kunci-nilai). Ini berguna ketika Anda perlu membuat dokumen atau objek baru dengan menggabungkan data dari berbagai sumber atau array dalam dokumen.

Parameter

- `inputs`: Sebuah array ekspresi yang menyelesaikan array. Array ini akan digabungkan menjadi satu array tupel.

- `useLongestLength`: (opsional) Jika`true`, array output akan memiliki panjang array input terpanjang, padding array yang lebih pendek dengan `null` nilai. Jika`false`, array output akan memiliki panjang array input terpendek.
- `defaults`: (opsional) Sebuah array nilai default untuk digunakan untuk tupel jika array input yang sesuai lebih pendek dari array input terpanjang dan `useLongestLength` adalah. `true`

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$zip` operator untuk menggabungkan dua array ke dalam array tunggal tupel.

Buat dokumen sampel

```
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"]
  }
])
```

Contoh kueri

```
db.grades.aggregate([
  {
    $project: {
      "name": 1,
      "scoredClasses": {
        $zip: {
          inputs: ["$scores", "$classes"],
          useLongestLength: true,
          defaults: [null, null]
        }
      }
```

```
    }
  }
])
```

Keluaran

```
[
  {
    "_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"]
    ]
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$zip` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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()
```

Geospasial

Bagian ini memberikan informasi rinci tentang operator geospasial yang didukung oleh Amazon DocumentDB.

Topik

- $geometry
- $geointersects
- $geowithin
- $maxDistance
- $minDistance
- $near
- $nearSphere

$geometry

`$geometry`Operator di Amazon DocumentDB digunakan untuk menentukan objek geometri GeoJSON sebagai bagian dari kueri geospasial. Operator ini digunakan bersama dengan operator kueri geospasial lainnya seperti `$geoWithin` dan `$geoIntersects` untuk melakukan kueri spasial pada data Anda.

Di Amazon DocumentDB, operator mendukung jenis `$geometry` geometri GeoJSON berikut:

- Poin
- LineString

- Polygon
- MultiPoint
- MultiLineString
- MultiPolygon
- GeometryCollection

Parameter

- `type`: Jenis objek geometri GeoJSON, misalnya,,, dll. `Point` `Polygon`
- `coordinates`: Sebuah array koordinat yang mewakili geometri. Struktur array koordinat tergantung pada jenis geometri.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan cara menggunakan `$geometry` operator untuk melakukan `$geoIntersects` query di Amazon DocumentDB.

Buat dokumen sampel

```
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]
```

```
      ]]
    }
  }
]);
```

Contoh kueri

```
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]
        ]]
      }
    }
  }
})
```

Keluaran

```
[
  {
    "_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]
        ]
      ]
    }
  }
```

```
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$geometry` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

`$geoIntersects`Operator di Amazon DocumentDB digunakan untuk menemukan dokumen yang data geospasialnya berpotongan dengan objek GeoJSON tertentu. Operator ini berguna untuk aplikasi yang memerlukan identifikasi dokumen berdasarkan hubungan spasialnya dengan bentuk geografis tertentu, seperti poligon atau multipoligon.

Parameter

- `$geometry`: Sebuah objek GeoJSON yang mewakili bentuk untuk memeriksa persimpangan. Jenis objek GeoJSON yang didukung `Point` adalah`LineString`,,, `Polygon` dan. `MultiPolygon`

Contoh (MongoDB Shell)

Contoh berikut menunjukkan cara menggunakan `$geoIntersects` operator untuk menemukan nama negara untuk kumpulan koordinat tertentu di Amazon DocumentDB.

Buat dokumen sampel

```
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]
      ]]
    }
  }
]);
```

Contoh kueri

```
var location = [-73.965355, 40.782865];

db.states.find({
  "loc": {
    "$geoIntersects": {
      "$geometry": {
        "type": "Point",
        "coordinates": location
      }
    }
  }
}, {
  "name": 1
});
```

Keluaran

```
{ "_id" : ObjectId("536b0a143004b15885c91a2c"), "name" : "New York" }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$geoIntersects` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$geoWithin`Operator di Amazon DocumentDB digunakan untuk menemukan dokumen yang data lokasinya (direpresentasikan sebagai objek GeoJSON) sepenuhnya terkandung dalam bentuk tertentu, seperti poligon atau multipoligon. Ini berguna untuk menanyakan objek yang terletak di dalam wilayah geografis tertentu.

Parameter

- `$geometry`: Sebuah objek GeoJSON yang mewakili bentuk untuk query terhadap.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$geoWithin` operator untuk menemukan semua bandara yang terletak di negara bagian New York.

Buat dokumen sampel

```
// 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]
        }
    }
]);
```

Contoh kueri

```
var state = db.states.findOne({"name": "New York"});

db.airports.find({
    "loc": {
        "$geoWithin": {
            "$geometry": state.loc
        }
    }
}, {
    "name": 1,
    "type": 1,
    "code": 1,
    "_id": 0
});
```

Keluaran

```
[
  {
    "name": "John F. Kennedy International Airport",
    "type": "airport",
    "code": "JFK"
  },
  {
    "name": "LaGuardia Airport",
    "type": "airport",
    "code": "LGA"
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$geoWithin` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$maxDistance`Operator di Amazon DocumentDB digunakan untuk menentukan jarak maksimum (dalam meter) dari titik GeoJSON di mana dokumen harus berada di dalamnya untuk dimasukkan dalam hasil kueri. Operator ini digunakan bersama dengan `$nearSphere` operator untuk melakukan kueri geospasial.

Parameter

- `$maxDistance`: Jarak maksimum (dalam meter) dari titik referensi di mana dokumen harus berada di dalamnya untuk dimasukkan dalam hasil kueri.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan cara menggunakan `$maxDistance` operator di Amazon DocumentDB untuk menemukan semua ibu kota negara bagian dalam jarak 100 kilometer dari Boston.

Buat dokumen sampel

```
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] } }
]);
```

Contoh kueri

```
db.capitals.find(
```

```
  {
    location: {
      $nearSphere: {
        $geometry: { type: "Point", coordinates: [-71.0589, 42.3601] },
        $maxDistance: 100000
      }
    }
  },
  { state: 1, city: 1, _id: 0 }
);
```

Keluaran

```
[
  { "state": "Rhode Island", "city": "Providence" },
  { "state": "New Hampshire", "city": "Concord" }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$maxDistance` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$minDistance`adalah operator find yang digunakan bersama dengan `$nearSphere` atau `$geoNear` untuk memfilter dokumen yang setidaknya pada jarak minimum yang ditentukan dari

titik pusat. Operator ini didukung di Amazon DocumentDB dan fungsinya mirip dengan mitranya di MongoDB.

Parameter

- `$minDistance`: Jarak minimum (dalam meter) dari titik pusat untuk memasukkan dokumen dalam hasil.

Contoh (MongoDB Shell)

Dalam contoh ini, kita akan menemukan semua restoran dalam radius 2 kilometer dari lokasi tertentu di Seattle, Washington.

Buat dokumen sampel

```
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]
    }
  }
]);
```

Contoh kueri

```
db.usarestaurants.find({
  "location": {
    "$nearSphere": {
      "$geometry": {
        "type": "Point",
        "coordinates": [-122.3516, 47.6156]
      },
      "$minDistance": 1,
      "$maxDistance": 2000
    }
  }
}, {
  "name": 1
});
```

Keluaran

```
{ "_id" : ObjectId("611f3da985009a81ad38e74b"), "name" : "Noodle House" }
{ "_id" : ObjectId("611f3da985009a81ad38e74c"), "name" : "Pike Place Grill" }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$minDistance` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$near`Operator di Amazon DocumentDB digunakan untuk menemukan dokumen yang secara geografis dekat titik tertentu. Ia mengembalikan dokumen yang dipesan berdasarkan jarak, dengan dokumen terdekat terlebih dahulu. Operator ini memerlukan indeks geospasial 2dsphere dan berguna untuk kueri kedekatan pada data lokasi.

Parameter

- `$geometry`: Sebuah objek GeoJSON Point yang mendefinisikan titik pusat untuk query dekat.
- `$maxDistance`: (opsional) Jarak maksimum dalam meter dari titik yang ditentukan bahwa dokumen dapat untuk mencocokkan kueri.
- `$minDistance`: (opsional) Jarak minimum dalam meter dari titik yang ditentukan bahwa dokumen dapat untuk mencocokkan kueri.

Persyaratan Indeks

- `2dsphere index`: Diperlukan untuk kueri geospasial pada data GeoJSON Point.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$near` operator untuk menemukan restoran terdekat ke lokasi tertentu di Seattle, Washington.

Buat dokumen sampel

```
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] }
  }
]);
```

Buat indeks 2dsphere

```
db.usarestaurants.createIndex({ "location": "2dsphere" });
```

Contoh kueri dengan GeoJSON Point

```
db.usarestaurants.find({
  location: {
    $near: {
      $geometry: {
        type: "Point",
        coordinates: [-122.3516, 47.6156]
      },
      $maxDistance: 100,
      $minDistance: 10
    }
  }
});
```

Keluaran

```
{
  "_id" : ObjectId("69031ec9ea1c2922a1ce5f4a"),
  "name" : "Noodle House",
  "city" : "Seattle",
  "state" : "Washington",
  "rating" : 4.8,
  "location" : {
    "type" : "Point",
    "coordinates" : [ -122.3517, 47.6159 ]
  }
}
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$near` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$nearSphere`Operator di Amazon DocumentDB digunakan untuk menemukan dokumen yang berada dalam jarak tertentu dari titik geospasial. Operator ini sangat berguna untuk kueri geo-spasial, seperti menemukan semua restoran dalam radius tertentu dari lokasi tertentu.

Parameter

- `$geometry`: Sebuah objek GeoJSON yang mewakili titik referensi. Harus menjadi `Point` objek dengan `type` dan `coordinates` bidang.
- `$minDistance`: (opsional) Jarak minimum (dalam meter) dari titik referensi yang harus dimiliki dokumen.
- `$maxDistance`: (opsional) Jarak maksimum (dalam meter) dari titik referensi yang harus dimiliki dokumen.

Contoh (MongoDB Shell)

Dalam contoh ini, kita akan menemukan semua restoran dalam jarak 2 kilometer (2000 meter) dari lokasi tertentu di Seattle, WA.

Buat dokumen sampel

```
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] }
  }
]);
```

Contoh kueri

```
db.usarestaurants.find({
```

```
  location: {
    $nearSphere: {
      $geometry: {
        type: "Point",
        coordinates: [-122.3516, 47.6156]
      },
      $minDistance: 1,
      $maxDistance: 2000
    }
  }
}, {
  name: 1
});
```

Keluaran

```
{ "_id" : ObjectId("611f3da985009a81ad38e74b"), "name" : "Noodle House" }
{ "_id" : ObjectId("611f3da985009a81ad38e74c"), "name" : "Pike Place Grill" }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$nearSphere` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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()
```

Operator kueri dan proyeksi

Bagian ini memberikan informasi terperinci tentang operator kueri dan proyeksi yang didukung oleh Amazon DocumentDB.

Topik

- $
- $ semua
- $and
- $ bit AllClear
- $ bit AllSet
- $ bit AnyClear
- $ bit AnySet
- $comment
- $elemMatch
- $eq
- $exists
- $expr
- $gt
- $gte
- $in
- $jsonSchema
- $lt
- $mod
- $lte
- $meta
- $ne
- $nin
- $nor
- $not

- $or
- $regex
- $slice
- $size
- $text
- $type

$

Operator $ proyeksi membatasi isi bidang array untuk mengembalikan hanya elemen pertama yang cocok dengan kondisi kueri. Hal ini digunakan untuk memproyeksikan elemen array pencocokan tunggal.

Parameter

- `field.$`: Bidang array dengan operator posisi untuk memproyeksikan elemen pencocokan pertama.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan menggunakan operator $ proyeksi untuk mengembalikan hanya elemen array yang cocok.

Buat dokumen sampel

```
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] }
]);
```

Contoh kueri

```
db.students.find(
  { grades: { $gte: 90 } },
  { name: 1, "grades.$": 1 }
);
```

Keluaran

```
{ "_id" : 1, "name" : "Alice", "grades" : [ 92 ] }
{ "_id" : 2, "name" : "Bob", "grades" : [ 92 ] }
{ "_id" : 3, "name" : "Charlie", "grades" : [ 95 ] }
```

Dalam contoh ini, hanya nilai satu yang lebih besar dari atau sama dengan 90 yang dikembalikan untuk setiap siswa.

Contoh kode

Untuk melihat contoh kode untuk menggunakan operator $ proyeksi, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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()
```

$ semua

`$all`Operator di Amazon DocumentDB digunakan untuk mencocokkan dokumen di mana nilai bidang adalah array dan berisi semua elemen yang ditentukan, terlepas dari urutan elemen dalam array.

Parameter

- `field`: Nama bidang yang akan diperiksa.
- `[value1, value2, ...]`: Daftar nilai yang cocok dalam array.

Menggunakan **`$elemMatch`** dalam **`$all`** ekspresi

Lihat [Menggunakan $elemMatch dalam ekspresi $all] batasan mengenai penggunaan `$elemMatch` operator dalam sebuah `$all` ekspresi.

Dolar ($) dalam nama bidang

Lihat [Dolar ($) dan titik (.) dalam nama bidang] batasan terkait kueri bidang $ awalan `$all` di objek bersarang.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$all` operator untuk mengambil dokumen di mana bidang "Warna" adalah array yang berisi "Merah" dan "Biru".

Buat dokumen sampel

```
db.example.insertMany([
  { "Item": "Pen", "Colors": ["Red", "Blue", "Green"] },
  { "Item": "Notebook", "Colors": ["Blue", "White"] },
  { "Item": "Poster Paint", "Colors": ["Red", "Yellow", "White"] }
])
```

Contoh kueri

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

Keluaran

```
{
  "_id" : ObjectId("6137d6c5b3a1d35e0b6ee6ad"),
  "Item" : "Pen",
  "Colors" : [
          "Red",
          "Blue",
          "Green"
  ]
}
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$all` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

`$and`Operator di Amazon DocumentDB digunakan untuk menggabungkan beberapa ekspresi dan mengevaluasinya sebagai satu kondisi. Ia kembali `true` jika semua ekspresi yang disediakan mengevaluasi`true`, dan `false` sebaliknya. Operator ini berguna untuk menerapkan beberapa kriteria ke kueri.

Parameter

- `expression1`: Ekspresi wajib yang mengevaluasi nilai boolean.
- `expression2`: Ekspresi wajib yang mengevaluasi nilai boolean.

- `...`: Ekspresi tambahan yang diperlukan yang mengevaluasi nilai boolean.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$and` operator untuk menemukan semua dokumen dalam koleksi “pengguna” di mana bidang “usia” lebih besar dari 18 dan bidang “status” adalah “aktif”.

Buat dokumen sampel

```
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" }
]);
```

Contoh kueri

```
db.users.find({
  $and: [
    { age: { $gt: 18 } },
    { status: "active" }
  ]
});
```

Keluaran

```
[
  { "_id" : ObjectId("614e3c4b63f5892e7c4e2345"), "name" : "John", "age" : 25,
 "status" : "active" },
  { "_id" : ObjectId("614e3c4b63f5892e7c4e2347"), "name" : "Alice", "age" : 22,
 "status" : "active" }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$and` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function findActiveUsersOlderThan18() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const users = db.collection('users');

  const activeUsersOlderThan18 = await users.find({
    $and: [
      { age: { $gt: 18 } },
      { status: 'active' }
    ]
  }).toArray();

  console.log(activeUsersOlderThan18);
  await client.close();
}

findActiveUsersOlderThan18();
```

Python

```
from pymongo import MongoClient

def find_active_users_older_than_18():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    users = db['users']

    active_users_older_than_18 = list(users.find({
        '$and': [
            {'age': {'$gt': 18}},
            {'status': 'active'}
        ]
    }))

    print(active_users_older_than_18)
```

```
    client.close()

find_active_users_older_than_18()
```

$ bit AllClear

`$bitsAllClear`Operator di Amazon DocumentDB digunakan untuk mencocokkan dokumen di mana semua bit yang ditentukan dalam bidang dihapus (diatur ke 0). Operator ini dapat berguna untuk melakukan operasi bitwise pada data yang disimpan.

Parameter

- `field`: Bidang untuk memeriksa bit tertentu yang dihapus.
- `value`: Bitmask numerik yang menentukan bit mana yang harus diperiksa, atau daftar posisi bit yang akan diperiksa. Bitmask numerik dapat berupa biner (0b...), desimal, heksadesimal (0x...), oktal (0o...), atau biner (). BinData Dalam daftar posisi bit, posisi bit yang paling tidak signifikan adalah 0.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$bitsAllClear` operator di Amazon DocumentDB.

Buat dokumen sampel

```
db.collection.insertMany([
  { _id: 1, bits: 0b1010 },
  { _id: 2, bits: 0b1100 },
  { _id: 3, bits: 0b0101 }
]);
```

Contoh kueri

```
db.collection.find({
  bits: { $bitsAllClear: 0b0011 }
})
```

Keluaran

```
{ "_id" : 2, "bits" : 12 }
```

Kueri memeriksa apakah semua bit yang ditentukan oleh bitmask `0b0011` (dua bit yang paling tidak signifikan) dihapus di `bits` lapangan. Dokumen dengan `_id` 2 memenuhi kondisi ini, karena `bits` bidangnya memiliki bit-bit yang dihapus.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$bitsAllClear` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('collection');

  const result = await collection.find({
    bits: { $bitsAllClear: 0b0011 }
  }).toArray();

  console.log(result);

  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client['test']
    collection = db['collection']
```

```
    result = list(collection.find({
        'bits': { '$bitsAllClear': 0b0011 }
    }))

    print(result)

    client.close()

example()
```

$ bit AllSet

`$bitsAllSet`Operator di Amazon DocumentDB digunakan untuk menanyakan dokumen di mana sekumpulan bit tertentu diatur ke 1 di bidang. Operator ini memungkinkan Anda untuk melakukan operasi bitwise pada nilai bidang dan dapat berguna ketika Anda perlu memeriksa status masing-masing bit dalam nilai numerik.

Parameter

- `field`: Nama bidang numerik untuk melakukan operasi bitwise pada.
- `value`: Bitmask numerik yang menentukan bit mana yang harus diperiksa, atau daftar posisi bit yang akan diperiksa. Bitmask numerik dapat berupa biner (0b...), desimal, heksadesimal (0x...), oktal (0o...), atau biner (). BinData Dalam daftar posisi bit, posisi bit yang paling tidak signifikan adalah 0.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$bitsAllSet` operator untuk menemukan dokumen di mana `flags` bidang memiliki semua bit set yang ditentukan oleh bitmask.

Buat dokumen sampel

```
db.collection.insert([
  { _id: 1, flags: 0b1010 },
  { _id: 2, flags: 0b1100 },
  { _id: 3, flags: 0b1110 }
])
```

Contoh kueri

```
db.collection.find({ flags: { $bitsAllSet: 0b1100 } })
```

Keluaran

```
{ "_id": 2, "flags": 12 },
{ "_id": 3, "flags": 14 }
```

Dalam contoh ini, kueri memeriksa dokumen di mana `flags` bidang memiliki semua bit yang ditetapkan oleh bitmask `0b1100` (yang mewakili nilai desimal 12). Dokumen dengan `_id` 2 dan 3 cocok dengan kriteria ini, karena nilai `flags` bidangnya memiliki semua bit yang diperlukan (bit paling tidak signifikan ke-3 dan ke-4).

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$bitsAllSet` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function findWithBitsAllSet() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('collection');

  const result = await collection.find({ flags: { $bitsAllSet:
 0b1100 } }).toArray();
  console.log(result);

  await client.close();
}

findWithBitsAllSet();
```

Python

```
from pymongo import MongoClient
```

```
def find_with_bits_all_set():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.collection

    result = list(collection.find({ 'flags': { '$bitsAllSet': 0b1100 } }))
    print(result)

    client.close()

find_with_bits_all_set()
```

$ bit AnyClear

`$bitsAnyClear`Operator di Amazon DocumentDB digunakan untuk mencocokkan dokumen di mana salah satu bit yang ditentukan dalam bidang dihapus (disetel ke 0). Ini dapat berguna untuk melakukan operasi bitwise pada nilai bidang yang disimpan dalam dokumen.

Parameter

- `field`: Bidang untuk memeriksa.
- `value`: Bitmask numerik yang menentukan bit mana yang harus diperiksa, atau daftar posisi bit yang akan diperiksa. Bitmask numerik dapat berupa biner (0b...), desimal, heksadesimal (0x...), oktal (0o...), atau biner (). BinData Dalam daftar posisi bit, posisi bit yang paling tidak signifikan adalah 0.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$bitsAnyClear` operator untuk memeriksa apakah ada bit yang jelas di `status` bidang `items` koleksi.

Buat dokumen sampel

```
db.items.insertMany([
  { "_id": 1, "status": 7 },
  { "_id": 2, "status": 15 },
```

```
  { "_id": 3, "status": 31 }
]);
```

Contoh kueri

```
db.items.find({ "status": { $bitsAnyClear: 8 } })
```

Keluaran

```
{ "_id" : 1, "status" : 7 }
```

Dalam contoh ini, kueri memeriksa dokumen di mana `status` bidang memiliki bit yang jelas (0) di bitmask 8 (biner`1000`). Dokumen dengan `status` nilai 7 (biner`111`) cocok dengan kueri, karena setidaknya satu bit jelas dalam bitmask yang disediakan. Bit bening yang cocok adalah bit ke-4 yang paling tidak signifikan.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$bitsAnyClear` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<cluster-endpoint>:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false');
  const db = client.db('test');
  const collection = db.collection('items');

  const result = await collection.find({ "status": { $bitsAnyClear:
 8 } }).toArray();
  console.log(result);

  await client.close();
}

example();
```

Python

```
from pymongo import MongoClient

def example():
    client = MongoClient('mongodb://<username>:<password>@<cluster-endpoint>:27017/?
tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred&retryWrites=false')
    db = client.test
    collection = db.items

    result = list(collection.find({ "status": { "$bitsAnyClear": 8 } }))
    print(result)

    client.close()

example()
```

$ bit AnySet

`$bitsAnySet`Operator di Amazon DocumentDB digunakan untuk menanyakan dokumen di mana setidaknya satu bit diatur ke 1 dalam bit yang ditentukan di bidang. Operator ini memungkinkan Anda melakukan operasi bitwise pada nilai yang disimpan di bidang, memungkinkan kueri dan analisis data yang efisien dengan karakteristik bitwise.

Parameter

- `field`: Nama bidang untuk menerapkan operasi bitwise ke.
- `value`: Bitmask numerik yang menentukan bit mana yang harus diperiksa, atau daftar posisi bit yang akan diperiksa. Bitmask numerik dapat berupa biner (0b...), desimal, heksadesimal (0x...), oktal (0o...), atau biner (). BinData Dalam daftar posisi bit, posisi bit yang paling tidak signifikan adalah 0.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$bitsAnySet` operator untuk menemukan dokumen di mana setidaknya satu bit diatur di `flags` lapangan.

Buat dokumen sampel

```
db.collection.insertMany([
  { _id: 1, flags: 0b1010 },
  { _id: 2, flags: 0b1100 },
  { _id: 3, flags: 0b0011 },
  { _id: 4, flags: 0b0100 }
]);
```

Contoh kueri

```
db.collection.find({
  flags: { $bitsAnySet: 0b1010 }
})
```

Keluaran

```
{ "_id" : 1, "flags" : 10 }
{ "_id" : 2, "flags" : 12 }
{ "_id" : 3, "flags" : 3 }
```

Query mengembalikan dokumen di mana setidaknya salah satu bit yang ditentukan dalam bitmask `0b1010` diatur di `flags` lapangan.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$bitsAnySet` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$comment`Operator di Amazon DocumentDB digunakan untuk melampirkan komentar ke kueri. Ini dapat berguna untuk memberikan konteks atau informasi tambahan tentang kueri, yang dapat membantu untuk tujuan debugging atau pendokumentasian. Komentar terlampir akan muncul sebagai bagian dari output operasi seperti db.currentTop ().

Parameter

- `string`: Komentar yang dilampirkan pada kueri.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$comment` operator di Amazon DocumentDB.

Buat dokumen sampel

```
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" }
]);
```

Contoh kueri

```
db.users.find({ age: { $gt: 25 } }, { _id: 0, name: 1, age: 1 }).comment("Retrieve
 users older than 25");
```

Keluaran

```
{ "name" : "John Doe", "age" : 30 }
{ "name" : "Bob Johnson", "age" : 35 }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$comment` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$elemMatch`Operator di Amazon DocumentDB digunakan untuk menanyakan bidang array dan mengembalikan dokumen di mana setidaknya satu elemen dalam array cocok dengan kriteria yang ditentukan. Operator ini sangat berguna ketika Anda memiliki struktur data yang kompleks dengan array bersarang atau dokumen yang disematkan.

Perencana versi 2.0 menambahkan dukungan indeks untuk`$elemMatch`.

Parameter

- `field`: Bidang array untuk query.
- `query`: Kriteria untuk mencocokkan dengan elemen array.

Menggunakan **`$elemMatch`** dalam **`$all`** ekspresi

Lihat [Menggunakan $elemMatch dalam ekspresi $all] batasan mengenai penggunaan `$elemMatch` operator dalam sebuah `$all` ekspresi.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$elemMatch` operator untuk menemukan dokumen di mana `parts` array memiliki setidaknya satu elemen yang cocok dengan kriteria yang ditentukan.

Buat dokumen sampel

```
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 }] }
]);
```

Contoh kueri

```
db.col.find({
  parts: { "$elemMatch": { part: "xyz", qty: { $lt: 11 } } }
})
```

Keluaran

```
{ "_id" : 1, "parts" : [ { "part" : "xyz", "qty" : 10 }, { "part" : "abc", "qty" :
 20 } ] }
{ "_id" : 2, "parts" : [ { "part" : "xyz", "qty" : 5 }, { "part" : "abc", "qty" :
 10 } ] }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$elemMatch` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('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

`$eq`Operator di Amazon DocumentDB digunakan untuk mencocokkan dokumen di mana nilai bidang sama dengan nilai yang ditentukan. Operator ini biasanya digunakan dalam `find()` metode untuk mengambil dokumen yang memenuhi kriteria yang ditentukan.

Parameter

- `field`: Bidang untuk memeriksa kondisi kesetaraan.
- `value`: Nilai untuk dibandingkan dengan bidang.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$eq` operator untuk menemukan semua dokumen di mana `name` bidang sama dengan`"Thai Curry Palace"`.

Buat dokumen sampel

```
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"] }
]);
```

Contoh kueri

```
db.restaurants.find({ name: { $eq: "Thai Curry Palace" } });
```

Keluaran

```
{ "_id" : ObjectId("68ee586f916df9d39f3d9414"), "name" : "Thai Curry Palace",
 "cuisine" : "Thai", "features" : [ "Private Dining" ] }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$eq` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$exists`Operator digunakan untuk memeriksa apakah bidang ada dalam dokumen atau tidak. `$exists`sangat berguna saat bekerja dengan indeks jarang di Amazon DocumentDB, di mana bidang yang diindeks mungkin tidak ada di semua dokumen.

Untuk menggunakan indeks jarang yang telah Anda buat dalam kueri, Anda harus menggunakan klausa `$exists` pada bidang yang mencakup indeks tersebut. Jika Anda menghilangkan`$exists`, Amazon DocumentDB tidak akan menggunakan indeks jarang untuk kueri.

Parameter

- `field`: Nama bidang untuk memeriksa keberadaan.
- `value`: Nilai boolean (`true`atau`false`) yang menentukan apakah bidang harus ada (`true`) atau tidak ada (`false`) dalam dokumen yang cocok.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$exists` operator dengan indeks jarang pada `special_diets` bidang dalam koleksi. `food`

Buat dokumen sampel

```
db.food.insertMany([
  { _id: 1, name: "Apple", special_diets: ["vegetarian", "gluten-free"] },
  { _id: 2, name: "Broccoli" },
  { _id: 3, name: "Chicken", special_diets: ["dairy-free"] }
]);
```

Buat indeks jarang pada bidang `special_diets`

```
db.food.createIndex({ "special_diets": 1 }, { sparse: true, name:
 "special_diets_sparse_asc" });
```

Contoh kueri

```
db.food.find({ "special_diets": { $exists: true } });
```

Keluaran

```
[
  { "_id" : 1, "name" : "Apple", "special_diets" : [ "vegetarian", "gluten-free" ] },
  { "_id" : 3, "name" : "Chicken", "special_diets" : [ "dairy-free" ] }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$exists` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

Baru dari versi 4.0.

Tidak didukung oleh cluster elastis.

`$expr`Operator di Amazon DocumentDB memungkinkan Anda menggunakan ekspresi agregasi dalam bahasa kueri. Ini memungkinkan Anda untuk melakukan perbandingan dan perhitungan yang kompleks pada bidang dalam dokumen, mirip dengan cara Anda menggunakan tahapan pipeline agregasi.

Parameter

- `expression`: Ekspresi yang mengembalikan nilai boolean, memungkinkan Anda untuk melakukan perbandingan dan perhitungan pada bidang dokumen.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$expr` operator untuk menemukan semua dokumen di mana `manufacturingCost` bidang lebih besar dari `price` bidang.

Buat dokumen sampel

```
db.inventory.insertMany([
  { item: "abc", manufacturingCost: 500, price: 100 },
  { item: "def", manufacturingCost: 300, price: 450 },
  { item: "ghi", manufacturingCost: 400, price: 120 }
]);
```

Contoh kueri

```
db.inventory.find({
  $expr: {
    $gt: ["$manufacturingCost", "$price"]
  }
})
```

Keluaran

```
{ "_id" : ObjectId("60b9d4d68d2cac581bc5a89a"), "item" : "abc", "manufacturingCost" :
 500, "price" : 100 },
{ "_id" : ObjectId("60b9d4d68d2cac581bc5a89c"), "item" : "ghi", "manufacturingCost" :
 400, "price" : 120 }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$expr` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

`$gt`Operator di Amazon DocumentDB digunakan untuk memilih dokumen di mana nilai bidang yang ditentukan lebih besar dari nilai yang ditentukan. Operator ini berguna untuk memfilter dan menanyakan data berdasarkan perbandingan numerik.

Parameter

- `field`: Bidang untuk membandingkan.
- `value`: Nilai untuk dibandingkan.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$gt` operator untuk menemukan semua dokumen di mana `age` bidang lebih besar dari 30.

Buat dokumen sampel

```
db.users.insertMany([
  { name: "John", age: 25 },
  { name: "Jane", age: 32 },
  { name: "Bob", age: 45 },
  { name: "Alice", age: 28 }
```

```
]);
```

Contoh kueri

```
db.users.find({ age: { $gt: 30 } });
```

Keluaran

```
{ "_id" : ObjectId("6249e5c22a5d39884a0a0001"), "name" : "Jane", "age" : 32 },
{ "_id" : ObjectId("6249e5c22a5d39884a0a0002"), "name" : "Bob", "age" : 45 }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$gt` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$gte`Operator di Amazon DocumentDB digunakan untuk mencocokkan nilai yang lebih besar dari atau sama dengan nilai yang ditentukan. Operator ini berguna untuk memfilter dan menanyakan data berdasarkan perbandingan numerik.

Parameter

- `field`: Bidang untuk memeriksa nilai yang diberikan.
- `value`: Nilai untuk dibandingkan dengan bidang.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$gte` operator di Amazon DocumentDB untuk menemukan semua dokumen di mana bidang “usia” lebih besar dari atau sama dengan 25.

Buat dokumen sampel

```
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 }
]);
```

Contoh kueri

```
db.users.find({ age: { $gte: 25 } }, { _id: 0, name: 1, age: 1 });
```

Keluaran

```
{ "name" : "Jane", "age" : 25 }
```

```
{ "name" : "Bob", "age" : 30 }
{ "name" : "Alice", "age" : 35 }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$gte` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$in`Operator di Amazon DocumentDB adalah operator kueri logis yang memungkinkan Anda menemukan dokumen di mana nilai bidang sama dengan nilai yang ditentukan dalam array.

Parameter

- `field`: Bidang untuk memeriksa array yang disediakan.
- `[value1, value2, ...]`: Array nilai untuk dicocokkan dengan bidang yang ditentukan.

Dollar (**$**) dalam nama bidang

Lihat Dolar ($) dan titik (.) dalam nama bidang batasan terkait kueri bidang $ awalan `$in` di objek bersarang.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$in` operator untuk menemukan dokumen di mana `color` bidang adalah salah satu nilai dalam array yang disediakan.

Buat dokumen sampel

```
db.colors.insertMany([
  { "_id": 1, "color": "red" },
  { "_id": 2, "color": "green" },
  { "_id": 3, "color": "blue" },
  { "_id": 4, "color": "yellow" },
  { "_id": 5, "color": "purple" }
])
```

Contoh kueri

```
db.colors.find({ "color": { "$in": ["red", "blue", "purple"] } })
```

Keluaran

```
{ "_id": 1, "color": "red" },
{ "_id": 3, "color": "blue" },
{ "_id": 5, "color": "purple" }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$in` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

Baru dari versi 4.0.

Tidak didukung oleh cluster elastis.

`$jsonSchema`Operator di Amazon DocumentDB digunakan untuk memfilter dokumen berdasarkan skema JSON yang ditentukan. Operator ini memungkinkan Anda untuk menanyakan dokumen yang cocok dengan skema JSON tertentu, memastikan bahwa dokumen yang diambil mematuhi persyaratan struktural dan tipe data tertentu.

Dengan menggunakan operator kueri `$jsonSchema` evaluasi sebagai bagian dari pembuatan koleksi, Anda dapat memvalidasi skema dokumen yang dimasukkan ke dalam koleksi. Lihat Menggunakan validasi skema JSON untuk informasi tambahan.

Parameter

- `required`(array): Menentukan bidang yang diperlukan dalam dokumen.
- `properties`(object): Mendefinisikan tipe data dan kendala lainnya untuk setiap bidang dalam dokumen.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$jsonSchema` operator untuk memfilter `employees` koleksi untuk hanya mengambil dokumen yang memiliki`name`, `employeeId` dan `age` bidang, dan `employeeId` bidang adalah jenis. `string`

Buat dokumen sampel

```
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" }
]);
```

Contoh kueri

```
db.employees.aggregate([
  { $match: {
    $jsonSchema: {
      required: ["name", "employeeId", "age"],
      properties: { "employeeId": { "bsonType": "string" } }
    }
  }}
]);
```

Keluaran

```
{ "_id" : ObjectId("6908e8b61f77fc26b2ecd26f"), "name" : { "firstName" : "Emily",
 "lastName" : "Brown" }, "employeeId" : "2", "age" : 25 }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$jsonSchema` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$lt`Operator di Amazon DocumentDB digunakan untuk mencocokkan nilai yang kurang dari nilai yang ditentukan. Operator ini berguna untuk memfilter dan menanyakan data berdasarkan perbandingan numerik.

Parameter

- `field`: Nama bidang untuk memeriksa.
- `value`: Nilai untuk dibandingkan.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$lt` operator untuk mengambil dokumen di mana `Inventory.OnHand` nilai kurang dari 50.

Buat dokumen sampel

```
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"] }
]);
```

Contoh kueri

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

Keluaran

```
{ "_id" : 1, "Inventory" : { "OnHand" : 25, "Sold" : 60 }, "Colors" : [ "Red",
 "Blue" ] }
{ "_id" : 3, "Inventory" : { "OnHand" : 10, "Sold" : 85 }, "Colors" : [ "Red",
 "Green" ] }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$lt` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Operator $mod kueri memilih dokumen di mana nilai bidang dibagi dengan pembagi memiliki sisa tertentu. Ini berguna untuk memfilter dokumen berdasarkan kondisi aritmatika modulo.

Parameter

- `divisor`: Angka untuk dibagi dengan.
- `remainder`: Nilai sisa yang diharapkan.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan menggunakan $mod operator untuk menemukan semua pesanan di mana kuantitas adalah angka ganjil.

Buat dokumen sampel

```
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 }
]);
```

Contoh kueri

```
db.orders.find({ quantity: { $mod: [2, 1] } });
```

Keluaran

```
{ "_id" : 1, "item" : "Widget", "quantity" : 15 }
{ "_id" : 3, "item" : "Tool", "quantity" : 7 }
{ "_id" : 5, "item" : "Part", "quantity" : 9 }
```

Kueri ini mengembalikan dokumen di mana kuantitas dibagi 2 memiliki sisa 1, secara efektif memilih semua jumlah ganjil.

Contoh kode

Untuk melihat contoh kode untuk menggunakan operator $mod kueri, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

`$lte`Operator di Amazon DocumentDB digunakan untuk mencocokkan dokumen di mana nilai bidang tertentu kurang dari atau sama dengan nilai yang ditentukan. Operator ini berguna untuk memfilter dan menanyakan data berdasarkan perbandingan numerik.

Parameter

- `field`: Bidang untuk membandingkan.
- `value`: Nilai untuk dibandingkan.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$lte` operator untuk mengambil dokumen di mana `quantity` bidang kurang dari atau sama dengan 10.

Buat dokumen sampel

```
db.inventory.insertMany([
  { item: "canvas", qty: 100 },
  { item: "paint", qty: 50 },
  { item: "brush", qty: 10 },
  { item: "paper", qty: 5 }
]);
```

Contoh kueri

```
db.inventory.find({ qty: { $lte: 10 } });
```

Keluaran

```
{ "_id" : ObjectId("..."), "item" : "brush", "qty" : 10 },
{ "_id" : ObjectId("..."), "item" : "paper", "qty" : 5 }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$lte` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$meta`Operator digunakan untuk mengakses metadata yang terkait dengan eksekusi kueri saat ini. Operator ini terutama digunakan untuk operasi pencarian teks, di mana metadata dapat memberikan informasi tentang relevansi dokumen yang cocok.

Parameter

- `textScore`: Mengambil skor pencarian teks untuk dokumen. Skor ini menunjukkan relevansi dokumen dengan kueri pencarian teks.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan cara menggunakan `$meta` operator untuk mengambil skor pencarian teks untuk dokumen yang cocok dengan permintaan pencarian teks.

Buat dokumen sampel

```
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." }
]);
```

Buat indeks teks

```
db.documents.createIndex({ content: "text" });
```

Contoh kueri

```
db.documents.find(
  { $text: { $search: "coffee" } },
  { _id: 0, title: 1, content: 1, score: { $meta: "textScore" } }
).sort({ score: { $meta: "textScore" } });
```

Keluaran

```
[
  {
    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
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$meta` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$ne`Operator digunakan untuk mencocokkan dokumen di mana nilai bidang tidak sama dengan nilai yang ditentukan. Ini adalah operator perbandingan yang dapat digunakan dalam predikat kueri untuk memfilter dokumen.

Perencana versi 2.0 menambahkan dukungan indeks untuk`$ne`.

Parameter

- `field`: Bidang untuk memeriksa.
- `value`: Nilai yang harus diperiksa.

Contoh (MongoDB Shell)

Dalam contoh ini, kita akan menemukan semua dokumen dalam `users` koleksi di mana `status` bidangnya tidak sama dengan`"active"`.

Buat dokumen sampel

```
db.users.insertMany([
  { name: "John", status: "active" },
  { name: "Jane", status: "inactive" },
  { name: "Bob", status: "suspended" },
  { name: "Alice", status: "active" }
]);
```

Contoh kueri

```
db.users.find({ status: { $ne: "active" } });
```

Keluaran

```
[
  {
    _id: ObjectId('...'),
    name: 'Jane',
    status: 'inactive'
  },
  {
    _id: ObjectId('...'),
    name: 'Bob',
    status: 'suspended'
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan $ne perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$nin`Operator digunakan untuk mencocokkan nilai yang tidak ada dalam array yang ditentukan. Ini adalah kebalikan dari `$in` operator, yang cocok dengan nilai-nilai yang ada dalam array yang ditentukan.

Perencana versi 2.0 menambahkan dukungan indeks untuk`$nin`.

Parameter

- `field`: Bidang untuk memeriksa.
- `array`: Array nilai untuk memeriksa terhadap.

Dollar (**$**) dalam nama bidang

Lihat Dolar ($) dan titik (.) dalam nama bidang batasan terkait kueri bidang $ awalan `$nin` di objek bersarang.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$nin` operator untuk menemukan dokumen di mana `category` bidang tidak sama dengan “Fiksi” atau “Misteri”.

Buat dokumen sampel

```
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" }
])
```

Contoh kueri

```
db.books.find({
  category: {
    $nin: ["Fiction", "Mystery"]
  }
})
```

Keluaran

```
[
  {
    _id: ObjectId('...'),
    title: 'The Martian',
    author: 'Andy Weir',
    category: 'Science Fiction'
  },
  {
    _id: ObjectId('...'),
    title: 'The Alchemist',
    author: 'Paulo Coelho',
    category: 'Philosophy'
  }
```

```
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$nin` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

$norOperator digunakan untuk mencocokkan dokumen di mana tidak ada kondisi kueri yang ditentukan yang benar. Ini mirip dengan operasi “NOR” logis, di mana hasilnya benar jika tidak ada operan yang benar.

Parameter

- `expression1`: Ekspresi pertama untuk mengevaluasi.
- `expression2`: Ekspresi kedua untuk mengevaluasi.
- `expressionN`: Ekspresi tambahan untuk dievaluasi.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan $nor operator dengan mengambil dokumen di mana `qty` bidang tidak kurang dari 20 dan `size` bidang tidak sama dengan “XL”.

Buat dokumen sampel

```
db.items.insertMany([
  { qty: 10, size: "M" },
  { qty: 15, size: "XL" },
  { qty: 25, size: "L" },
  { qty: 30, size: "XL" }
])
```

Contoh kueri

```
db.items.find({
  $nor: [
    { qty: { $lt: 20 } },
    { size: "XL" }
  ]
})
```

Keluaran

```
[
  { "_id" : ObjectId("..."), "qty" : 25, "size" : "L" }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$nor` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

`$not`Operator digunakan untuk meniadakan hasil dari ekspresi yang diberikan. Ini memungkinkan Anda untuk memilih dokumen di mana kondisi yang ditentukan tidak cocok.

Perencana versi 2.0 menambahkan dukungan indeks untuk `$not {eq}` dan`$not {in}`.

Parameter

- `expression`: Ekspresi untuk meniadakan.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$not` operator untuk menemukan dokumen di mana `status` bidang tidak sama dengan “aktif”.

Buat dokumen sampel

```
db.users.insertMany([
  { name: "John", status: "active" },
  { name: "Jane", status: "inactive" },
  { name: "Bob", status: "pending" },
  { name: "Alice", status: "active" }
]);
```

Contoh kueri

```
db.users.find({ status: { $not: { $eq: "active" } } });
```

Keluaran

```
[
  {
    _id: ObjectId('...'),
    name: 'Jane',
    status: 'inactive'
  },
  {
    _id: ObjectId('...'),
    name: 'Bob',
    status: 'pending'
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$not` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$or`Operator digunakan untuk melakukan operasi OR logis pada array dari dua atau lebih ekspresi. Ia mengembalikan dokumen yang cocok dengan setidaknya salah satu ekspresi. Operator ini berguna ketika Anda perlu menanyakan dokumen yang memenuhi salah satu dari beberapa kondisi.

Parameter

- `expression1`: Ekspresi pertama untuk mengevaluasi.
- `expression2`: Ekspresi kedua untuk mengevaluasi.
- `...`: Ekspresi tambahan untuk dievaluasi (opsional).

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$or` operator untuk menemukan dokumen di mana `make` adalah "TruckForYou" dengan model “Heavy H1" atau" SportForYou "dengan model “Bolid 1".

Buat dokumen sampel

```
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 }
]);
```

Contoh kueri

```
db.cars.find({
  $or: [
    { make: "TruckForYou", model: "Heavy H1" },
    { make: "SportForYou", model: "Bolid 1" }
  ]
});
```

Keluaran

```
[
  {
    _id: ObjectId('...'),
    make: 'TruckForYou',
    model: 'Heavy H1',
    year: 2020
  },
  {
    _id: ObjectId('...'),
    make: 'SportForYou',
    model: 'Bolid 1',
    year: 2021
  }
]
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$or` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$regex`Operator memungkinkan Anda untuk melakukan pencocokan ekspresi reguler pada bidang string. Ini adalah alat yang ampuh untuk mencari dan memfilter dokumen berdasarkan pola yang kompleks.

Parameter

- `regular expression`: Pola ekspresi reguler untuk mencocokkan dengan bidang.
- `$options`: (opsional) Menyediakan opsi untuk memodifikasi perilaku pencarian, seperti sensitivitas huruf kecil, pencocokan global, dll.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$regex` operator untuk mencari dokumen di mana bidang “nama” cocok dengan pola tertentu.

Buat dokumen sampel

```
db.users.insertMany([
  { name: "John Doe" },
  { name: "Jane Smith" },
  { name: "Alice Johnson" },
  { name: "Bob Williams" },
  { name: "Charlie Davis" }
]);
```

Contoh kueri

```
db.users.find({ name: { $regex: /^A/ } })
```

Keluaran

```
[
```

```
  { "_id" : ObjectId("..."), "name" : "Alice Johnson" }
]
```

Kueri ini akan mengembalikan semua dokumen di mana bidang “nama” dimulai dengan huruf “A”.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$regex` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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()
```

$slice

Operator `$slice` proyeksi membatasi jumlah elemen array yang dikembalikan dalam hasil query. Ini memungkinkan Anda untuk mengambil sejumlah elemen tertentu dari awal atau akhir bidang array tanpa memuat seluruh array.

Parameter

- `field`: Bidang array untuk proyek.
- `count`: Jumlah elemen yang akan dikembalikan. Nilai positif mengembalikan elemen dari awal, nilai negatif dari akhir.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan operator `$slice` proyeksi untuk mengembalikan hanya dua item pertama dari bidang array.

Buat dokumen sampel

```
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"] }
]);
```

Contoh kueri

```
db.inventory.find(
  {},
  { item: 1, tags: { $slice: 2 } }
)
```

Keluaran

```
{ "_id" : 1, "item" : "notebook", "tags" : [ "office", "school" ] }
{ "_id" : 2, "item" : "pen", "tags" : [ "office", "writing" ] }
{ "_id" : 3, "item" : "folder", "tags" : [ "office", "supplies" ] }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan operator `$slice` proyeksi, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Operator `$size` kueri cocok dengan dokumen di mana bidang array memiliki persis jumlah elemen yang ditentukan. Ini berguna untuk memfilter dokumen berdasarkan panjang array.

Parameter

- `field`: Bidang array untuk memeriksa.
- `count`: Jumlah pasti elemen array harus berisi.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan menggunakan `$size` operator untuk menemukan semua produk yang memiliki tepat tiga tag.

Buat dokumen sampel

```
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"] }
]);
```

Contoh kueri

```
db.products.find({ tags: { $size: 3 } });
```

Keluaran

```
{ "_id" : 1, "name" : "Laptop", "tags" : [ "electronics", "computers", "portable" ] }
{ "_id" : 3, "name" : "Desk", "tags" : [ "furniture", "office", "workspace" ] }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan operator `$size` kueri, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

`$text`Operator digunakan untuk melakukan pencarian teks lengkap pada bidang yang diindeks teks dalam koleksi dokumen. Operator ini memungkinkan Anda untuk mencari dokumen yang berisi kata atau frasa tertentu, dan dapat dikombinasikan dengan operator kueri lain untuk memfilter hasil berdasarkan kriteria tambahan.

Parameter

- `$search`: String teks untuk mencari.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan cara menggunakan `$text` operator untuk mencari dokumen yang berisi kata “minat” dan memfilter hasil berdasarkan bidang “star_rating”.

Buat dokumen sampel

```
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" }
]);
```

Buat indeks teks

```
db.test.createIndex({ comments: "text" });
```

Contoh kueri

```
db.test.find({$and: [{star_rating: 5}, {$text: {$search: "interest"}}]})
```

Keluaran

```
{ "_id" : 5, "star_rating" : 5, "comments" : "interesting couch" }
{ "_id" : 6, "star_rating" : 5, "comments" : "interested in couch for sale, year
 2022" }
```

Perintah di atas mengembalikan dokumen dengan bidang yang diindeks teks yang berisi segala bentuk “minat” dan “star_rating” sama dengan 5.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$text` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$type`Operator digunakan untuk memeriksa tipe data bidang dalam dokumen. Ini dapat digunakan ketika operasi atau validasi tipe spesifik diperlukan. `$type`Operator mengembalikan tipe BSON dari ekspresi yang dievaluasi. Tipe yang dikembalikan adalah string, yang sesuai dengan jenis bidang atau ekspresi.

Perencana versi 2.0 menambahkan dukungan indeks untuk`$type`.

Parameter

- `expression`: Ekspresi untuk mengevaluasi.

Contoh (MongoDB Shell)

Buat dokumen sampel

```
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 }
]);
```

Contoh kueri

```
db.documents.find({
  $or: [
    { age: { $type: "number" } },
    { email: { $type: "string" } },
    { name: { $type: "string" } }
  ]
})
```

Keluaran

```
[
  { "_id": 1, "name": "John", "age": 30, "email": "john@example.com" },
  { "_id": 2, "name": "Jane", "age": "25", "email": 123456 }
]
```

Kueri ini akan mengembalikan dokumen di mana `age` bidangnya bertipe “angka”, `email` bidangnya bertipe “string”, dan `name` bidangnya bertipe “string”.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$type` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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()
```

Perbarui operator

Bagian ini memberikan informasi terperinci tentang operator pembaruan yang didukung oleh Amazon DocumentDB.

Topik

- $
- $[]

- $ [<pengidentifikasi>]
- $ tambahkan ToSet
- $bit
- $currentDate
- $each
- $inc
- $max
- $min
- $mul
- $pop
- $position
- $pull
- $pullAll
- $push
- $rename
- $set
- $ set OnInsert
- $slice
- $sort
- $unset

$

Operator $ posisi memperbarui elemen array pertama yang cocok dengan kondisi kueri. Ini bertindak sebagai placeholder untuk posisi elemen array yang cocok.

Parameter

- `field.$`: Bidang array dengan operator posisi untuk memperbarui elemen pencocokan pertama.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan menggunakan operator $ posisi untuk memperbarui elemen array tertentu.

Buat dokumen sampel

```
db.inventory.insertMany([
  { _id: 1, item: "Widget", quantities: [10, 20, 30] },
  { _id: 2, item: "Gadget", quantities: [5, 15, 25] }
]);
```

Contoh kueri

```
db.inventory.updateOne(
  { _id: 1, quantities: 20 },
  { $set: { "quantities.$": 22 } }
);
```

Keluaran

```
{
  "_id" : 1,
  "item" : "Widget",
  "quantities" : [ 10, 22, 30 ]
}
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan operator $ posisi, pilih tab untuk bahasa yang ingin Anda gunakan:

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()
```

$[]

Operator `$[]` semua posisi memperbarui semua elemen dalam array. Hal ini digunakan ketika Anda perlu memodifikasi setiap elemen dalam bidang array.

Parameter

- `field.$[]`: Bidang array dengan operator semua posisi untuk memperbarui semua elemen.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan menggunakan `$[]` operator untuk memperbarui semua elemen array.

Buat dokumen sampel

```
db.products.insertOne({
  _id: 1,
  name: "Laptop",
  prices: [1000, 1100, 1200]
});
```

Contoh kueri

```
db.products.updateOne(
  { _id: 1 },
  { $inc: { "prices.$[]": 50 } }
);
```

Keluaran

```
{
  "_id" : 1,
  "name" : "Laptop",
  "prices" : [ 1050, 1150, 1250 ]
}
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$[]` operator, pilih tab untuk bahasa yang ingin Anda gunakan:

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()
```

$ [<pengidentifikasi>]

Operator posisi yang `$[<identifier>]` difilter memperbarui semua elemen array yang cocok dengan kondisi filter yang ditentukan. Hal ini digunakan dengan `arrayFilters` pilihan untuk selektif memperbarui elemen array.

Parameter

- `field.$[identifier]`: Bidang array dengan operator posisi yang disaring.
- `arrayFilters`: Array kondisi filter yang menentukan elemen mana yang akan diperbarui.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan menggunakan `$[<identifier>]` operator untuk memperbarui elemen array tertentu berdasarkan kondisi.

Buat dokumen sampel

```
db.students.insertOne({
  _id: 1,
  name: "Alice",
  grades: [
    { subject: "Math", score: 85 },
    { subject: "Science", score: 92 },
    { subject: "History", score: 78 }
  ]
});
```

Contoh kueri

```
db.students.updateOne(
  { _id: 1 },
  { $inc: { "grades.$[elem].score": 5 } },
  { arrayFilters: [{ "elem.score": { $gte: 80 } }] }
);
```

Keluaran

```
{
```

```
  "_id" : 1,
  "name" : "Alice",
  "grades" : [
    { "subject" : "Math", "score" : 90 },
    { "subject" : "Science", "score" : 97 },
    { "subject" : "History", "score" : 78 }
  ]
}
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$[<identifier>]` operator, pilih tab untuk bahasa yang ingin Anda gunakan:

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()
```

$ tambahkan ToSet

`$addToSet`Operator di Amazon DocumentDB digunakan untuk menambahkan nilai ke array hanya jika nilainya belum ada dalam array. Ini berguna untuk memastikan bahwa array berisi elemen unik.

Parameter

- `field`: Bidang untuk memperbarui.
- `value`: Nilai untuk ditambahkan ke bidang array. Ini bisa berupa nilai tunggal atau ekspresi.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$addToSet` operator untuk menambahkan elemen unik ke array.

Buat dokumen sampel

```
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"] }
])
```

Contoh kueri

```
db.products.update(
  { "item": "apple" },
  { $addToSet: { "tags": "green" } }
)
```

Keluaran

```
{ "_id": 1, "item": "apple", "tags": ["fruit", "red", "round", "green"] }
```

Dalam contoh ini, `$addToSet` operator menambahkan tag “hijau” ke array “tag” dokumen di mana bidang “item” adalah “apel”. Karena “hijau” belum ada dalam array, itu ditambahkan.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$addToSet` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

`$bit`Operator di Amazon DocumentDB memungkinkan Anda melakukan operasi bitwise pada bit bidang tertentu. Ini dapat berguna untuk tugas-tugas seperti pengaturan, pembersihan, atau memeriksa status bit individu dalam angka.

Parameter

- `field`: Bidang untuk melakukan operasi bitwise pada.
- `and`: Nilai integer yang digunakan untuk melakukan operasi AND bitwise di lapangan.
- `or`: Nilai integer yang digunakan untuk melakukan operasi OR bitwise di lapangan.
- `xor`: Nilai integer yang digunakan untuk melakukan operasi XOR bitwise di lapangan.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$bit` operator untuk melakukan operasi bitwise pada bidang numerik.

Buat dokumen sampel

```
db.numbers.insert([
  { "_id": 1, "number": 5 },
  { "_id": 2, "number": 12 }
])
```

Contoh kueri

```
db.numbers.update(
  { "_id": 1 },
  { "$bit": { "number": { "and": 3 } } }
)
```

Keluaran

```
{
  "_id": 1,
  "number": 1
}
```

Dalam contoh ini, `$bit` operator digunakan untuk melakukan operasi bitwise AND pada bidang “angka” dokumen dengan 1. `_id` Hasilnya adalah bahwa nilai bidang “angka” diatur ke 1, yang merupakan hasil dari operasi bitwise AND antara nilai asli 5 dan nilai 3.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$bit` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$currentDate`Operator digunakan untuk mengatur nilai bidang ke tanggal dan waktu saat ini. Operator ini berguna untuk memperbarui bidang secara otomatis dengan stempel waktu saat ini ketika dokumen dimasukkan atau diperbarui.

Parameter

- `field`: Bidang untuk memperbarui dengan tanggal dan waktu saat ini.
- `type`: (opsional) Menentukan jenis BSON untuk digunakan untuk tanggal saat ini. Bisa salah satu `date` ataU`timestamp`.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan cara menggunakan `$currentDate` operator untuk mengatur `lastModified` bidang ke tanggal dan waktu saat ini ketika dokumen baru dimasukkan.

Buat dokumen sampel

```
db.users.insert({
  name: "John Doe",
  email: "john.doe@example.com"
})
```

Contoh kueri

```
db.users.updateOne(
  { name: "John Doe" },
  { $currentDate: { lastModified: true } }
)
```

Lihat dokumen yang diperbarui

```
db.users.findOne({ name: "John Doe" })
```

Keluaran

```
{
```

```
  _id: ObjectId('...'),
  name: 'John Doe',
  email: 'john.doe@example.com',
  lastModified: ISODate('2025-10-25T22:50:29.963Z')
}
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$currentDate` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$each`Operator digunakan bersama dengan operator pembaruan lainnya, seperti `$push` dan`$addToSet`, untuk menambahkan beberapa nilai ke bidang array. Ini memungkinkan menambahkan beberapa elemen ke array dalam satu operasi, daripada harus menjalankan beberapa operasi pembaruan.

Parameter

- `value`: Array nilai untuk ditambahkan ke bidang array.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan menggunakan `$each` operator dengan `$push` operator untuk menambahkan beberapa elemen ke bidang array.

Buat dokumen sampel

```
db.fruits.insertOne({
  _id: 1,
  fruits: ["apple", "banana"]
})
```

Contoh kueri

```
db.fruits.updateOne(
  { _id: 1 },
  { $push: { fruits: { $each: ["cherry", "durian", "elderberry"] } } }
```

```
)
```

Lihat dokumen yang diperbarui

```
db.fruits.findOne({ _id: 1 })
```

Keluaran

```
{
  _id: 1,
  fruits: [ 'apple', 'banana', 'cherry', 'durian', 'elderberry' ]
}
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$each` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

`$inc`Operator digunakan untuk menambah nilai bidang dengan jumlah tertentu. Ini digunakan untuk memperbarui bidang numerik, seperti penghitung atau peringkat, tanpa harus mengambil nilai saat ini, menghitung nilai baru, dan kemudian memperbarui bidang.

Parameter

- `field`: Nama bidang untuk kenaikan.
- `amount`: Jumlah yang digunakan untuk menambah bidang. Ini bisa menjadi nilai positif atau negatif.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$inc` operator untuk menambah `age` bidang dokumen.

Buat dokumen sampel

```
db.users.insertOne({_id: 123, name: "John Doe", age: 30})
```

Contoh kueri

```
db.users.updateOne({_id: 123}, {$inc: {age: 1}})
```

Lihat dokumen yang diperbarui

```
db.users.findOne({_id: 123})
```

Keluaran

```
{ "_id" : 123, "name" : "John Doe", "age" : 31 }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$inc` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

Operator `$max` pembaruan memperbarui nilai bidang hanya jika nilai yang ditentukan lebih besar dari nilai bidang saat ini. Operator ini berguna untuk mempertahankan nilai maksimum di seluruh pembaruan.

Parameter

- `field`: Bidang untuk memperbarui.
- `value`: Nilai untuk membandingkan dengan nilai bidang saat ini.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan menggunakan `$max` operator untuk memperbarui skor tertinggi yang tercatat untuk pemain.

Buat dokumen sampel

```
db.scores.insertMany([
  { _id: 1, player: "Alice", highScore: 85 },
  { _id: 2, player: "Bob", highScore: 92 },
```

```
  { _id: 3, player: "Charlie", highScore: 78 }
])
```

Perbarui contoh

```
db.scores.updateOne(
  { _id: 1 },
  { $max: { highScore: 95 } }
)
```

Hasil

`highScore`Bidang untuk Alice diperbarui menjadi 95 karena 95 lebih besar dari nilai saat ini 85.

```
{ "_id": 1, "player": "Alice", "highScore": 95 }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$max` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

Operator `$min` pembaruan memperbarui nilai bidang hanya jika nilai yang ditentukan kurang dari nilai bidang saat ini. Operator ini berguna untuk mempertahankan nilai minimum di seluruh pembaruan.

Parameter

- `field`: Bidang untuk memperbarui.
- `value`: Nilai untuk membandingkan dengan nilai bidang saat ini.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan menggunakan `$min` operator untuk memperbarui suhu terendah yang tercatat untuk stasiun cuaca.

Buat dokumen sampel

```
db.weather.insertMany([
  { _id: 1, station: "Station A", lowestTemp: 15 },
```

```
  { _id: 2, station: "Station B", lowestTemp: 20 },
  { _id: 3, station: "Station C", lowestTemp: 18 }
])
```

Perbarui contoh

```
db.weather.updateOne(
  { _id: 1 },
  { $min: { lowestTemp: 12 } }
)
```

Hasil

`lowestTemp`Bidang untuk Stasiun A diperbarui ke 12 karena 12 kurang dari nilai saat ini 15.

```
{ "_id": 1, "station": "Station A", "lowestTemp": 12 }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$min` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

`$mul`Operator di Amazon DocumentDB digunakan untuk mengalikan nilai bidang dengan nomor tertentu. Ini dapat berguna untuk memperbarui beberapa dokumen secara atomik dan konsisten, seperti memperbarui mil penerbangan berdasarkan status kartu kredit.

Parameter

- `field`: Bidang yang akan dikalikan.
- `multiplier`: Angka untuk mengalikan nilai bidang dengan.

Contoh (MongoDB Shell)

Contoh ini menunjukkan bagaimana menggunakan `$mul` operator untuk menggandakan `flight_miles` nilai untuk semua dokumen di mana `credit_card` bidang tersebut`true`.

Buat dokumen sampel

```
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] }
]);
```

Contoh kueri

```
db.miles.update(
  { "credit_card": { "$eq": true } },
  { "$mul": { "flight_miles.$[]": NumberInt(2) } },
  { "multi": true }
);
```

Keluaran

```
{ "_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 ] }
```

Untuk pelanggan yang memiliki kartu kredit, mil penerbangan mereka telah berlipat ganda.

Operator array `$[]` posisional digunakan untuk menerapkan `$mul` operasi ke setiap elemen dalam `flight_miles` array.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$mul` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

$popOperator di Amazon DocumentDB digunakan untuk menghapus elemen pertama atau terakhir dari bidang array. Ini sangat berguna ketika Anda perlu mempertahankan array ukuran tetap atau menerapkan struktur data seperti antrian dalam dokumen.

Parameter

- `field`: Nama bidang array untuk menghapus elemen dari.
- `value`: Nilai integer yang menentukan posisi elemen untuk menghapus. Nilai 1 menghapus elemen terakhir, sementara nilai `-1` menghapus elemen pertama.

Contoh (MongoDB Shell)

Contoh ini menunjukkan bagaimana menggunakan $pop operator untuk menghapus elemen pertama dan terakhir dari bidang array.

Buat dokumen sampel

```
db.users.insertMany([
  { "_id": 1, "name": "John Doe", "hobbies": ["reading", "swimming", "hiking"] },
  { "_id": 2, "name": "Jane Smith", "hobbies": ["cooking", "gardening", "painting"] }
])
```

Contoh kueri

```
// 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 } })
```

Keluaran

```
{ "_id" : 1, "name" : "John Doe", "hobbies" : [ "swimming", "hiking" ] }
{ "_id" : 2, "name" : "Jane Smith", "hobbies" : [ "cooking", "gardening" ] }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan $pop perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
  const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

`$position`Pengubah di Amazon DocumentDB menentukan lokasi dalam array di `$push` mana operator menyisipkan elemen. Tanpa `$position` modifier, `$push` operator menyisipkan elemen ke akhir array.

Parameter

- `field`: Bidang array untuk memperbarui.
- num: Posisi dalam array di mana elemen harus dimasukkan, berdasarkan pengindeksan berbasis nol.

Catatan: Untuk menggunakan `$position` pengubah, itu harus muncul dengan `$each` pengubah.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$position` operator untuk menyisipkan tugas pada posisi tertentu dalam sistem manajemen proyek.

Buat dokumen sampel

```
db.projects.insertOne({ "_id": 1, "name": "Website Redesign", "tasks": ["Design
 mockups"] })
```

Contoh kueri 1 - Tambahkan tugas mendesak di awal

```
db.projects.updateOne(
   { _id: 1 },
```

```
   {
     $push: {
        tasks: {
           $each: ["Security audit", "Performance review"],
           $position: 0
        }
     }
   }
)
```

Keluaran 1

```
{ "_id": 1, "name": "Website Redesign", "tasks": ["Security audit", "Performance
 review", "Design mockups"] }
```

Contoh kueri 2 - Tambahkan tugas pada posisi tertentu

```
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
        }
     }
   }
)
```

Keluaran 2

```
{ "_id": 2, "name": "Mobile App", "tasks": ["Setup project", "Create wireframes", "Code
 review", "Testing phase", "Deploy to store"] }
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$position` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$pull`Operator digunakan untuk menghapus dari array semua contoh nilai atau nilai yang cocok dengan kondisi tertentu. Operator ini berguna ketika Anda perlu menghapus elemen tertentu dari bidang array dalam dokumen.

Parameter

- `field`: Nama bidang array dari mana untuk menghapus nilai (s).
- `value`: Nilai atau kondisi yang menentukan elemen mana yang akan dihapus dari array.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$pull` operator untuk menghapus elemen dari bidang array.

Buat dokumen sampel

```
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"]
  }
])
```

Contoh kueri

```
db.restaurants.updateMany(
  { cuisine: "Italian" },
  { $pull: { features: "Takeout" } }
)
```

Keluaran

```
{
  "acknowledged" : true,
  "matchedCount" : 1,
  "modifiedCount" : 1
}
```

Kueri di atas menghapus fitur “Takeout” dari semua dokumen yang `cuisine` bidangnya “Italia”.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$pull` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$pullAll`Operator di Amazon DocumentDB digunakan untuk menghapus semua instance nilai yang ditentukan dari bidang array. Ini sangat berguna ketika Anda perlu menghapus beberapa elemen dari array dalam satu operasi.

Parameter

- `field`: Nama bidang array dari mana untuk menghapus elemen.
- `value`: Sebuah array nilai untuk menghapus dari bidang array.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$pullAll` operator untuk menghapus beberapa elemen dari bidang array.

Buat dokumen sampel

```
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"]
  }
])
```

Contoh kueri

```
db.restaurants.update(
  { "name": "Taj Mahal" },
  { $pullAll: { "features": ["Private Dining", "Live Music"] } }
```

```
)
```

Keluaran

```
{
  "name": "Taj Mahal",
  "cuisine": "Indian",
  "features": []
}
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$pullAll` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$push`Operator di Amazon DocumentDB digunakan untuk menambahkan item ke bidang array dalam dokumen. Operator ini sangat berguna ketika Anda perlu menambahkan data baru ke array yang ada tanpa menimpa seluruh array.

Parameter

- `field`: Nama bidang array yang elemen baru harus ditambahkan.
- `value`: Nilai yang akan ditambahkan ke array.
- `position`: (opsional) Sebuah modifier yang menentukan posisi dalam array di mana elemen baru harus ditambahkan. Pengubah yang didukung termasuk $ (tambahkan ke akhir array) dan `$[]` (tambahkan ke akhir array, mengabaikan filter array apa pun).

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$push` operator untuk menambahkan elemen baru ke bidang array dalam dokumen.

Buat dokumen sampel

```
db.users.insert([
  { _id: 1, name: "John Doe", hobbies: ["reading", "swimming"] },
  { _id: 2, name: "Jane Smith", hobbies: ["gardening", "cooking"] }
])
```

Contoh kueri

```
db.users.updateOne(
  { _id: 1 },
  { $push: { hobbies: "hiking" } }
)
```

Keluaran

```
{
  "acknowledged" : true,
  "matchedCount" : 1,
  "modifiedCount" : 1
}
```

Setelah menjalankan pembaruan, dokumen dengan hobbies array _id: 1 akan diperbarui["reading", "swimming", "hiking"].

Contoh kode

Untuk melihat contoh kode untuk menggunakan $push perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$rename`Operator di Amazon DocumentDB digunakan untuk mengganti nama bidang dalam dokumen. Operator ini dapat sangat berguna ketika Anda perlu memperbarui struktur dokumen Anda atau menyelaraskannya dengan model data baru.

Parameter

- `field`: Bidang yang akan diganti namanya.

- `newName`: Nama baru untuk bidang.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan cara menggunakan `$rename` operator untuk mengganti nama bidang menjadi `"Date.DayOfWeek"` dalam dokumen dengan `"Date.DoW"` bidang yang disetel ke. `"DocName" "Document 1"`

Buat dokumen sampel

```
db.example.insertOne({
    "DocName": "Document 1",
    "Date": {
        "Month": 4,
        "Day": 18,
        "Year": 1987,
        "DoW": "Saturday"
    },
    "Words": 2482
})
```

Contoh kueri

```
db.example.update(
    { "DocName": "Document 1" },
    { $rename: { "Date.DoW": "Date.DayOfWeek" } }
)
```

Keluaran

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

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$rename` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

Node.js

```
const { MongoClient } = require('mongodb');

async function example() {
    const client = await MongoClient.connect('mongodb://
<username>:<password>@<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

`$set`Operator di Amazon DocumentDB digunakan untuk memperbarui nilai bidang tertentu dalam dokumen. Operator ini memungkinkan Anda untuk menambahkan bidang baru atau memodifikasi yang sudah ada dalam dokumen. Ini adalah operator pembaruan mendasar dalam driver MongoDB Java, yang kompatibel dengan Amazon DocumentDB.

Parameter

- `field`: Bidang untuk memperbarui.
- `value`: Nilai baru untuk bidang.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan bagaimana menggunakan `$set` operator untuk memperbarui `Item` bidang dalam dokumen.

Buat dokumen sampel

```
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
    }
  }
])
```

Contoh kueri

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

Keluaran

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

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$set` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$setOnInsert`Operator di Amazon DocumentDB digunakan untuk menetapkan nilai bidang jika dokumen dimasukkan, tetapi tidak berpengaruh jika dokumen sedang diperbarui.

Parameter

- `field`: Bidang yang akan diatur.
- `value`: Nilai untuk menetapkan ke bidang.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan penggunaan `$setOnInsert` operator di Amazon DocumentDB. Ini membuat dokumen baru jika dokumen belum ada, tetapi tidak berpengaruh jika dokumen sedang diperbarui.

Buat dokumen sampel

```
db.users.insertOne({
  _id: 1,
  name: "John Doe",
  age: 30
})
```

Contoh kueri 1 - Perbarui dokumen yang ada

```
db.users.update(
  { _id: 1 },
  {
    $set: { age: 31 },
    $setOnInsert: { createdAt: new Date() }
  },
  { upsert: true }
)
```

Keluaran 1

```
{
  _id: 1,
  name: "John Doe",
  age: 31
}
```

Output menunjukkan bahwa dokumen telah diperbarui, tetapi `createdAt` bidang TIDAK ditambahkan karena dokumen sudah ada. `$setOnInsert`Operator hanya berlaku saat memasukkan dokumen baru.

Contoh kueri 2 - Sisipkan dokumen baru (upsert)

```
db.users.update(
  { _id: 2 },
  {
    $set: { name: "Jane Smith", age: 25 },
    $setOnInsert: { createdAt: new Date() }
  },
  { upsert: true }
)
```

Keluaran 2

```
{
  _id: 2,
  name: "Jane Smith",
  age: 25,
  createdAt: ISODate("2025-10-31T09:57:52.459Z")
}
}
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$setOnInsert` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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

Operator `$slice` pembaruan memodifikasi array dengan membatasi ukurannya. Ketika digunakan dengan `$push` operator, ini membatasi jumlah elemen dalam array, hanya menyimpan jumlah tertentu dari elemen terbaru atau tertua.

Parameter

- `field`: Bidang array untuk memodifikasi.
- `count`: Jumlah maksimum elemen yang harus disimpan. Nilai positif menjaga elemen N pertama, nilai negatif menjaga elemen N terakhir.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan cara menggunakan operator `$slice` pembaruan `$push` untuk mempertahankan array ukuran tetap dari skor terbaru.

Buat dokumen sampel

```
db.students.insertOne({
  _id: 1,
  name: "Alice",
  scores: [85, 90, 78]
});
```

Contoh kueri

```
db.students.updateOne(
  { _id: 1 },
  {
    $push: {
      scores: {
        $each: [92, 88],
        $slice: -3
      }
    }
  }
)
```

Keluaran

```
{
  "_id" : 1,
  "name" : "Alice",
  "scores" : [ 78, 92, 88 ]
}
```

Dalam contoh ini, `$slice: -3` pengubah hanya menyimpan tiga elemen terakhir setelah mendorong nilai baru ke array.

Contoh kode

Untuk melihat contoh kode untuk menggunakan operator `$slice` pembaruan, pilih tab untuk bahasa yang ingin Anda gunakan:

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

Pengubah `$sort` pembaruan memesan elemen array saat digunakan dengan `$push` operator. Ini mengatur elemen array dalam urutan naik atau turun berdasarkan nilai bidang tertentu atau elemen itu sendiri.

Parameter

- `field`: Bidang array untuk memodifikasi.

- `order`: Gunakan 1 untuk urutan menaik atau `-1` untuk urutan menurun.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan menggunakan `$sort` pengubah dengan `$push` untuk menambahkan skor kuis baru dan menjaga mereka diurutkan dalam urutan menurun.

Buat dokumen sampel

```
db.students.insertOne({
  _id: 1,
  name: "Bob",
  quizzes: [
    { score: 85, date: "2024-01-15" },
    { score: 92, date: "2024-02-10" }
  ]
});
```

Contoh kueri

```
db.students.updateOne(
  { _id: 1 },
  {
    $push: {
      quizzes: {
        $each: [{ score: 78, date: "2024-03-05" }],
        $sort: { score: -1 }
      }
    }
  }
)
```

Keluaran

```
{
  "_id" : 1,
  "name" : "Bob",
  "quizzes" : [
    { "score" : 92, "date" : "2024-02-10" },
    { "score" : 85, "date" : "2024-01-15" },
    { "score" : 78, "date" : "2024-03-05" }
  ]
```

```
}
```

Contoh kode

Untuk melihat contoh kode untuk menggunakan pengubah `$sort` pembaruan, pilih tab untuk bahasa yang ingin Anda gunakan:

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

`$unset`Operator di Amazon DocumentDB digunakan untuk menghapus bidang tertentu dari dokumen. Ketika bidang dihapus menggunakan`$unset`, bidang dihapus dari dokumen, dan ukuran dokumen dikurangi sesuai. Ini dapat berguna ketika Anda ingin menghapus data yang tidak perlu dari dokumen Anda.

Parameter

- `field`: Bidang untuk menghapus dari dokumen. Ini bisa berupa bidang tunggal atau jalur putus-putus ke bidang bersarang.

Contoh (MongoDB Shell)

Contoh berikut menunjukkan cara menggunakan `$unset` operator untuk menghapus `Words` bidang dari dokumen dalam `example` koleksi.

Buat dokumen sampel

```
db.example.insert({
    "DocName": "Document 1",
    "Date": {
        "Month": 4,
        "Day": 18,
        "Year": 1987,
        "DoW": "Saturday"
    },
    "Words": 2482
})
```

Contoh kueri

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

Keluaran

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

Dalam contoh ini, `$unset` operator digunakan untuk menghapus `Words` bidang dari dokumen dengan `DocName` sama dengan “Dokumen 1". Dokumen yang dihasilkan tidak lagi berisi `Words` bidang.

Contoh kode

Untuk melihat contoh kode untuk menggunakan `$unset` perintah, pilih tab untuk bahasa yang ingin Anda gunakan:

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()
```

Amazon DocumentDB kecerdasan buatan generatif

Amazon DocumentDB menawarkan kemampuan untuk memungkinkan model pembelajaran mesin (ML) dan kecerdasan buatan generatif (AI) untuk bekerja dengan data yang disimpan di Amazon DocumentDB secara real time. Pelanggan tidak lagi harus menghabiskan waktu mengelola infrastruktur terpisah, menulis kode untuk terhubung dengan layanan lain, dan menduplikasi data dari database utama mereka.

Untuk informasi lebih lanjut tentang kecerdasan buatan dan bagaimana AWS dapat mendukung kebutuhan AI Anda, lihat artikel What-is"" ini.

Topik

- No-code pembelajaran mesin dengan Amazon SageMaker AI Canvas
- Pencarian vektor untuk Amazon DocumentDB

No-code pembelajaran mesin dengan Amazon SageMaker AI Canvas

Amazon SageMaker AI Canvas memungkinkan Anda membuat AI/ML model Anda sendiri tanpa harus menulis satu baris kode pun. Anda dapat membuat model ML untuk kasus penggunaan umum seperti regresi dan peramalan dan dapat mengakses dan mengevaluasi model pondasi (FM) dari Amazon Bedrock. Anda juga dapat mengakses FM publik dari Amazon SageMaker AI JumpStart untuk pembuatan konten, ekstraksi teks, dan ringkasan teks untuk mendukung solusi AI generatif.

Cara membuat model MLtanpa kode dengan SageMaker AI Canvas

Amazon DocumentDB sekarang terintegrasi dengan SageMaker Amazon AI Canvas untuk mengaktifkan pembelajaran mesin tanpa kode (ML) dengan data yang disimpan di Amazon DocumentDB. Anda sekarang dapat membangun model ML untuk kebutuhan regresi dan peramalan dan menggunakan model dasar untuk ringkasan dan pembuatan konten menggunakan data yang disimpan di Amazon DocumentDB tanpa menulis satu baris kode pun.

SageMaker AI Canvas menyediakan antarmuka visual yang memungkinkan pelanggan Amazon DocumentDB menghasilkan prediksi tanpa memerlukan keahlian AI/ML apa pun atau menulis satu baris kode. Pelanggan sekarang dapat meluncurkan ruang kerja SageMaker AI Canvas dari Konsol Manajemen AWS, mengimpor, dan bergabung dengan data Amazon DocumentDB untuk persiapan

data dan pelatihan model. Data di Amazon DocumentDB sekarang dapat digunakan SageMaker di AI Canvas untuk membangun dan menambah model untuk memprediksi churn pelanggan, mendeteksi penipuan, memprediksi kegagalan pemeliharaan, memperkirakan metrik bisnis, dan menghasilkan konten. Pelanggan sekarang dapat mempublikasikan dan berbagi ML-driven wawasan di seluruh tim menggunakan integrasi asli SageMaker AI Canvas dengan Quick. Saluran konsumsi data di SageMaker AI Canvas berjalan di instans sekunder Amazon DocumentDB secara default, memastikan bahwa kinerja aplikasi dan SageMaker beban kerja konsumsi Kanvas AI tidak terhalang.

Pelanggan Amazon DocumentDB dapat memulai SageMaker dengan AI Canvas dengan menavigasi ke halaman Amazon DocumentDB MS Console baru dan menghubungkan ke ruang No-Code kerja AI Canvas baru atau yang tersedia. SageMaker

Mengkonfigurasi domain SageMaker AI dan profil pengguna

Anda dapat terhubung ke cluster Amazon DocumentDB SageMaker dari domain AI yang berjalan dalam mode VPC Only. Dengan meluncurkan domain SageMaker AI di VPC Anda, Anda dapat mengontrol aliran data dari lingkungan SageMaker AI Studio dan Canvas Anda. Ini memungkinkan Anda untuk membatasi akses internet, memantau dan memeriksa lalu lintas menggunakan AWS jaringan standar dan kemampuan keamanan, dan terhubung ke AWS sumber daya lain melalui titik akhir VPC. Lihat Amazon SageMaker AI Canvas Memulai dan Mengonfigurasi Amazon SageMaker AI Canvas di VPC tanpa akses internet yang terletak di Panduan Pengembang Amazon SageMaker AI untuk membuat domain SageMaker AI Anda agar terhubung ke cluster Amazon DocumentDB Anda.

Mengkonfigurasi izin akses IAM untuk Amazon DocumentDB dan AI Canvas SageMaker

Pengguna Amazon DocumentDB yang `AmazonDocDBConsoleFullAccess` telah melekat pada peran dan identitas terkait mereka dapat mengakses file. Konsol Manajemen AWS Tambahkan tindakan berikut ke peran atau identitas yang disebutkan di atas untuk menyediakan akses ke pembelajaran mesin tanpa kode dengan Amazon SageMaker AI Canvas.

```
"sagemaker:CreatePresignedDomainUrl",
"sagemaker:DescribeDomain",
"sagemaker:ListDomains",
"sagemaker:ListUserProfiles"
```

Membuat pengguna database dan peran untuk SageMaker AI Canvas

Anda dapat membatasi akses ke tindakan yang dapat dilakukan pengguna pada database menggunakan kontrol akses berbasis peran (RBAC) di Amazon DocumentDB. RBAC bekerja dengan memberikan satu atau lebih peran kepada pengguna. Peran ini menentukan operasi yang dapat dilakukan pengguna pada sumber daya database.

Sebagai pengguna Canvas, Anda terhubung ke database Amazon DocumentDB dengan kredensi nama pengguna dan kata sandi. Anda dapat membuat database user/role untuk pengguna Canvas yang memiliki akses baca ke database tertentu menggunakan fungsionalitas Amazon DocumentDB RBAC.

Misalnya, gunakan `createUser` operasi:

```
db.createUser({
user: "canvas_user",
pwd: "<insert-password>",
roles: [{role: "read", db: "sample-database-1"}]
})
```

Ini menciptakan `canvas_user` yang telah membaca izin ke `sample-database-1` database. Analis Canvas Anda dapat menggunakan kredensi ini untuk mengakses data di klaster Amazon DocumentDB Anda. Lihat Akses database menggunakan Role-Based Access Control untuk mempelajari lebih lanjut.

Wilayah yang tersedia

Integrasi tanpa kode tersedia di wilayah di mana Amazon DocumentDB SageMaker dan Amazon AI Canvas didukung. Daerah-daerah tersebut meliputi:

- us-east-1 (Virginia Utara)
- us-east-2 (Ohio)
- us-west-2 (Oregon)
- ap-northeast-1 (Tokyo)
- ap-northeast-2 (Seoul)
- ap-south-1 (Mumbai)
- ap-southeast-1 (Singapura)
- ap-southeast-2 (Sydney)

- eu-central-1 (Frankfurt am Main)
- eu-west-1 (Irlandia)

Lihat Amazon SageMaker AI Canvas di Panduan Pengembang Amazon SageMaker AI untuk ketersediaan wilayah terbaru.

Pencarian vektor untuk Amazon DocumentDB

Pencarian vektor adalah metode yang digunakan dalam pembelajaran mesin untuk menemukan titik data yang serupa dengan titik data tertentu dengan membandingkan representasi vektor mereka menggunakan metrik jarak atau kesamaan. Semakin dekat kedua vektor berada di ruang vektor, semakin mirip item yang mendasarinya. Teknik ini membantu menangkap makna semantik dari data. Pendekatan ini berguna dalam berbagai aplikasi, seperti sistem rekomendasi, pemrosesan bahasa alami, dan pengenalan gambar.

Pencarian vektor untuk Amazon DocumentDB menggabungkan fleksibilitas dan kemampuan query yang kaya dari database dokumen JSON-based dengan kekuatan pencarian vektor. Jika Anda ingin menggunakan data Amazon DocumentDB yang ada atau struktur data dokumen yang fleksibel untuk membangun pembelajaran mesin dan kasus penggunaan AI generatif, seperti pengalaman pencarian semantik, rekomendasi produk, personalisasi, chatbots, deteksi penipuan, dan deteksi anomali, maka pencarian vektor untuk Amazon DocumentDB adalah pilihan ideal untuk Anda. Pencarian vektor tersedia di cluster berbasis instans Amazon DocumentDB 5.0.

Topik

- Memasukkan vektor
- Membuat indeks vektor
- Mendapatkan definisi indeks
- Menanyakan vektor
- Fitur dan keterbatasan
- Praktik terbaik

Memasukkan vektor

Untuk menyisipkan vektor ke dalam database Amazon DocumentDB, Anda dapat menggunakan metode penyisipan yang ada:

Contoh

Dalam contoh berikut, kumpulan lima dokumen dalam database pengujian dibuat. Setiap dokumen mencakup dua bidang: nama produk dan penyematan vektor yang sesuai.

```
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]}
]);
```

Membuat indeks vektor

Amazon DocumentDB mendukung pengindeksan Hierarchical Navigable Small World (HNSW) dan metode pengindeksan File Terbalik dengan Kompresi Datar (iVFFlat). Indeks iVFFlat memisahkan vektor ke dalam daftar dan selanjutnya mencari subset terpilih dari daftar yang paling dekat dengan vektor kueri. Di sisi lain, indeks HNSW mengatur data vektor ke dalam grafik berlapis-lapis. Meskipun HNSW memiliki waktu pembuatan yang lebih lambat dibandingkan dengan iVFFlat, HNSW memberikan kinerja kueri dan penarikan yang lebih baik. Tidak seperti iVFFlat, HNSW tidak memiliki langkah pelatihan yang terlibat, memungkinkan indeks dihasilkan tanpa beban data awal. Untuk sebagian besar kasus penggunaan, sebaiknya gunakan tipe indeks HNSW untuk pencarian vektor.

Jika Anda tidak membuat indeks vektor, Amazon DocumentDB melakukan pencarian tetangga terdekat yang tepat, memastikan penarikan yang sempurna. Namun, dalam skenario produksi, kecepatan sangat penting. Kami merekomendasikan penggunaan indeks vektor, yang dapat memperdagangkan beberapa penarikan untuk meningkatkan kecepatan. Penting untuk dicatat bahwa menambahkan indeks vektor dapat menyebabkan hasil kueri yang berbeda.

Template

Anda dapat menggunakan `runCommand` templat `createIndex` atau templat berikut untuk membuat indeks vektor pada bidang vektor:

Using createIndex

> Pada driver tertentu, seperti mongosh dan Java, menggunakan `vectorOptions` parameter di `createIndex` dapat mengakibatkan kesalahan. Dalam kasus seperti itu, kami sarankan menggunakan`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

Pada driver tertentu, seperti mongosh dan Java, menggunakan `vectorOptions` parameter di `createIndex` dapat mengakibatkan kesalahan. Dalam kasus seperti itu, kami sarankan menggunakan`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"
      }]
  }
);
```

Parameter	Persyaratan	Jenis data	Deskripsi	Nilai
`name`	opsional	string	Menentukan nama indeks.	Alfanumerik
`type`	opsional		Menentukan jenis indeks.	Didukung: hnsw atau ivfflat Default: HNSW (patch mesin 3.0.4574 dan seterusnya)
`dimensions`	wajib	integer	Menentukan jumlah dimensi dalam data vektor.	Maksimal 2.000 dimensi.
`similarity`	wajib	string	Menentukan metrik jarak yang digunakan untuk perhitungan kesamaan.	• **`euclidean`** • **`cosine`** • **`dotProduct`**
`lists`	diperlukan untuk iVFFlat	integer	Menentukan jumlah cluster yang indeks iVFFlat menggunakan untuk mengelompokkan data vektor. Pengaturan yang disarankan adalah # dari documents/1000 hingga 1 juta	Minimal: 1 Maksimum: Lihat daftar per tabel jenis instans di Fitur dan keterbatasan bawah ini.

Parameter	Persyaratan	Jenis data	Deskripsi	Nilai
			dokumen dan `sqrt(# of documents)` untuk lebih dari 1 juta dokumen.	
`m`	opsional	integer	Menentukan jumlah max koneksi untuk indeks HNSW	Default: 16 Rentang [2, 100]
`efConstruction`	opsional	integer	Menentukan ukuran daftar kandidat dinamis untuk membangun grafik untuk indeks HNSW. `efConstruction` harus lebih besar dari atau sama dengan (2 * m)	Default: 64 Rentang [4, 1000]

Penting bagi Anda untuk menetapkan nilai sub-parameter seperti `lists` untuk iVFFlat dan `m` dan `efConstruction` untuk HNSW dengan tepat karena akan memengaruhi, waktu pembuatan accuracy/recall, dan kinerja pencarian Anda. Nilai daftar yang lebih tinggi meningkatkan kecepatan kueri karena mengurangi jumlah vektor di setiap daftar, menghasilkan wilayah yang lebih kecil. Namun, ukuran wilayah yang lebih kecil dapat menyebabkan lebih banyak kesalahan penarikan, menghasilkan akurasi yang lebih rendah. Untuk HNSW, meningkatkan nilai `m` dan `efConstruction` meningkatkan akurasi, tetapi juga meningkatkan waktu dan ukuran pembuatan indeks. Lihat contoh berikut:

Contoh

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
    }
  }
)
```

Mendapatkan definisi indeks

Anda dapat melihat detail indeks Anda, termasuk indeks vektor, menggunakan perintah: `getIndexes`

Contoh

```
db.collection.getIndexes()
```

Contoh keluaran

```
[
```

```
 {
  "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"
 }
]
```

Menanyakan vektor

Amazon DocumentDB mendukung dua operator pencarian vektor untuk menanyakan vektor:

Operator pencarian vektor klasik

Gunakan template berikut untuk menanyakan vektor:

```
db.collection.aggregate([
  {
    $search: {
      "vectorSearch": {
        "vector": <query vector>,
        "path": "<vectorField>",
        "similarity": "<distance metric>",
        "k": <number of results>,
        "probes":<number of probes> [applicable for IVFFlat],
        "efSearch":<size of the dynamic list during search> [applicable for HNSW]
      }
```

```
    }
  }
]);
```

Parameter	Persyaratan	Tipe	Deskripsi	Nilai
vectorSearch	wajib	operator	Digunakan di dalam perintah $search untuk query vektor.	
vector	wajib	array	Menunjukkan vektor kueri yang akan digunakan untuk menemukan vektor serupa.	
path	wajib	string	Mendefinisikan nama bidang vektor.	
k	wajib	integer	Menentukan jumlah hasil yang pencarian kembali.	
similarity	wajib	string	Menentukan metrik jarak yang digunakan untuk perhitungan kesamaan.	• **euclidean** • **cosine** • **dotProduct**
probes	opsional	integer	Jumlah cluster yang Anda inginkan pencarian vektor untuk diperiksa.	Default: 1

Parameter	Persyaratan	Tipe	Deskripsi	Nilai
			Nilai yang lebih tinggi memberikan penarikan yang lebih baik dengan mengorbankan kecepatan. Ini dapat diatur ke jumlah daftar untuk pencarian tetangga terdekat yang tepat (di mana perencana tidak akan menggunakan indeks). Pengaturan yang disarankan untuk memulai fine-tuning adalah. `sqrt(# of lists)`	

Parameter	Persyaratan	Tipe	Deskripsi	Nilai
`efSearch`	opsional	integer	Menentukan ukuran daftar kandidat dinamis yang digunakan indeks HNSW selama pencarian. Nilai yang lebih tinggi `efSearch` memberikan penarikan yang lebih baik dengan mengorbankan kecepatan.	Default: 40 Rentang [1, 1000]

Penting untuk menyempurnakan nilai `efSearch` (HNSW) atau `probes` (iVFFlat) untuk mencapai kinerja dan akurasi yang Anda inginkan. Lihat contoh operasi berikut:

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
      }
    }
  }
]);
```

Contoh keluaran

Keluaran dari operasi ini terlihat seperti berikut ini:

```
{ "_id" : ObjectId("653d835ff96bee02cad7323c"), "product_name" : "Product A",
 "vectorEmbedding" : [ 0.2, 0.5, 0.8 ] }
{ "_id" : ObjectId("653d835ff96bee02cad7323e"), "product_name" : "Product C",
 "vectorEmbedding" : [ 0.1, 0.2, 0.5 ] }
```

$VectorSearch operator (tersedia di Amazon DocumentDB 8.0 dan seterusnya)

Gunakan template berikut untuk menanyakan vektor:

```
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],
  }
}])
```

Fitur dan keterbatasan

Kompatibilitas versi

- Pencarian vektor untuk Amazon DocumentDB hanya tersedia di Amazon DocumentDB 5.0+ cluster berbasis instans.

vektor

- Amazon DocumentDB dapat mengindeks vektor hingga 2.000 dimensi. Namun, hingga 16.000 dimensi dapat disimpan tanpa indeks.

Indeks

- Untuk pembuatan indeks iVFFlat, pengaturan yang disarankan untuk parameter daftar adalah jumlah documents/1000 hingga 1 juta dokumen dan `sqrt(# of documents)` untuk lebih dari 1 juta dokumen. Karena batas memori yang berfungsi, Amazon DocumentDB mendukung nilai maksimum tertentu dari parameter daftar tergantung pada jumlah dimensi. Untuk referensi Anda, tabel berikut memberikan nilai maksimum parameter daftar untuk vektor 500, 1000, dan 2.000 dimensi:

Tipe instans	Daftar dengan 500 dimensi	Daftar dengan 1000 dimensi	Daftar dengan 2000 dimensi
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

Tipe instans	Daftar dengan 500 dimensi	Daftar dengan 1000 dimensi	Daftar dengan 2000 dimensi
r5.24xl	19,585	19,335	18,845

- Tidak ada opsi indeks lain seperti`compound`, `sparse` atau `partial` didukung dengan indeks vektor.
- Build indeks paralel tidak didukung untuk indeks HNSW di Amazon DocumentDB 5.0.

kueri vektor

- Untuk kueri pencarian vektor, penting untuk menyempurnakan parameter seperti `probes` atau `efSearch` untuk hasil yang optimal. Semakin tinggi nilai `probes` atau `efSearch` parameter, semakin tinggi recall dan semakin rendah kecepatannya. Pengaturan yang disarankan untuk mulai menyempurnakan parameter probe adalah. `sqrt(# of lists)`

Praktik terbaik

Pelajari praktik terbaik untuk bekerja dengan pencarian vektor di Amazon DocumentDB. Bagian ini terus diperbarui saat praktik terbaik baru diidentifikasi.

- Pembuatan indeks Inverted File with Flat Compression (iVFFlat) melibatkan pengelompokan dan pengorganisasian titik data berdasarkan kesamaan. Oleh karena itu, agar indeks menjadi lebih efektif, kami menyarankan Anda setidaknya memuat beberapa data sebelum membuat indeks.
- Untuk kueri pencarian vektor, penting untuk menyempurnakan parameter seperti `probes` atau `efSearch` untuk hasil yang optimal. Semakin tinggi nilai `efSearch` parameter `probes` atau, semakin tinggi recall dan semakin rendah kecepatannya. Pengaturan yang disarankan untuk memulai fine tuning `probes` parameter adalah`sqrt(lists)`.

Sumber Daya

- Pencarian vektor apa posting blog baru
- Contoh kode pencarian semantik
- Sampel kode pencarian vektor Amazon DocumentDB

Migrasi ke Amazon DocumentDB

Amazon DocumentDB adalah layanan database yang dikelola sepenuhnya yang kompatibel dengan MongoDB API. Anda dapat memigrasikan data Anda ke Amazon DocumentDB dari basis data MongoDB yang berjalan di lokasi atau di Amazon Elastic Compute Cloud (Amazon EC2) menggunakan proses yang diperinci pada bagian ini.

Topik

- Migrasi ke Amazon DocumentDB menggunakan AWS Database Migration Service (DMS): Panduan Memulai Cepat
- Runbook migrasi Amazon DocumentDB
- Migrasi dari Couchbase Server

Migrasi ke Amazon DocumentDB menggunakan AWS Database Migration Service (DMS): Panduan Memulai Cepat

Topik

- Siapkan sumber DMS
- Pengaturan DMS
- Aktifkan kompresi DocumentDB
- Buat tugas replikasi
- Memantau kemajuan
- Informasi tambahan

Siapkan sumber DMS

Lihat Mengaktifkan aliran perubahan untuk mengaktifkan aliran perubahan DocumentDB atau mengaktifkan Oplog MongoDB untuk mendukung DMS Change Data Capture (CDC).

- Sumber DMS harus mempertahankan semua perubahan yang sedang berlangsung sampai beban penuh DMS selesai untuk semua koleksi yang disertakan.

- Aliran perubahan DocumentDB berbasis waktu. Pastikan `change_stream_log_retention_duration` pengaturan Anda cukup besar untuk menutupi waktu untuk menyelesaikan beban penuh.
- MongoDB Oplog adalah ukuran tetap. Pastikan ukurannya untuk menahan semua operasi selama beban penuh.

Pengaturan DMS

Buat instance DMS, sumber, dan titik akhir target dan uji setiap titik akhir.

Aktifkan kompresi DocumentDB

Aktifkan kompresi dengan melampirkan grup parameter kustom ke cluster DocumentDB Anda dan memperbarui parameter default_collection_compression untuk diaktifkan. Untuk informasi selengkapnya, lihat Mengelola kompresi dokumen tingkat koleksi.

Buat tugas replikasi

1. Di konsol DMS, di panel navigasi, pilih Migrasi atau replikasi, lalu pilih Tugas.
2. Pilih Buat tugas.
3. Pada halaman Buat tugas, di bagian Konfigurasi tugas:
   - Masukkan pengidentifikasi Tugas yang unik dan bermakna (misalnya, “mongodb-docdb-replication”).
   - Pilih titik akhir sumber yang Anda buat sebelumnya di menu drop-down titik akhir database Sumber.
   - Pilih titik akhir target yang Anda buat sebelumnya di menu tarik-turun titik akhir database Target.
   - Untuk jenis Tugas, pilih Migrasi dan replikasi.
4. Di bagian Pengaturan:
   - Untuk log Tugas, centang kotak Aktifkan CloudWatch log.
   - Untuk mode Editing (di bagian atas bagian), pilih editor JSON dan atur atribut berikut:
     - Setel `ParallelApplyThreads` ke 5 (di bawah`TargetMetadata`). Ini memungkinkan insert/update ~1000/delete ops per detik di CDC.

- Setel `MaxFullLoadSubTasks` ke 16 (di bawah`FullLoadSettings`). Pertimbangkan untuk meningkatkan ini tergantung pada ukuran instans Anda.
- Untuk koleksi besar (lebih dari 100 GB), aktifkan partisi otomatis (di bawah Pemetaan Tabel dan di bawah `parallel-load` atribut):
  - “type”: “partisi-otomatis”
  - “jumlah partisi”: 16

Memantau kemajuan

Gunakan AWS DMS konsol atau buat dasbor khusus (alat dasbor) untuk melacak migrasi. Fokus pada metrik berikut:

- FullLoadThroughputBandwidthTarget— Mengukur bandwidth jaringan (in KB/second) yang digunakan oleh DMS saat mentransfer data ke database target selama fase migrasi beban penuh.
- CDCLatencyTarget— Mengukur waktu tunda (dalam detik) saat terjadi perubahan dalam database sumber dan kapan perubahan itu diterapkan ke database target.
- CDCThroughputRowsTarget— Mengukur jumlah baris per detik yang diterapkan DMS ke database target selama fase replikasi migrasi yang sedang berlangsung.

Informasi tambahan

Untuk informasi selengkapnya tentang Amazon DocumentDB AWS DMS dan, lihat: Lihat untuk informasi selengkapnya.

- Runbook migrasi Amazon DocumentDB
- Migrasi dari MongoDB ke Amazon DocumentDB

Runbook migrasi Amazon DocumentDB

Runbook ini menyediakan panduan komprehensif untuk memigrasi database MongoDB ke Amazon DocumentDB menggunakan (DMS). AWS Database Migration Service Ini dirancang untuk mendukung administrator database, insinyur cloud, dan pengembang sepanjang perjalanan migrasi ujung ke ujung — dari penemuan awal hingga validasi pasca-migrasi.

Mengingat perbedaan dalam implementasi dan fitur yang didukung antara MongoDB dan Amazon DocumentDB, runbook ini menekankan pendekatan terstruktur dan sistematis. Ini menguraikan penilaian pra-migrasi yang penting, menyoroti pertimbangan kompatibilitas, dan merinci tugas-tugas utama yang diperlukan untuk memastikan migrasi yang berhasil dengan gangguan minimal.

Runbook diatur ke dalam topik-topik berikut:

- Kompatibilitas— Memahami fitur MongoDB yang didukung dan tipe data di Amazon DocumentDB, dan mengidentifikasi potensi ketidakcocokan.
- Penemuan beban kerja— Menganalisis beban kerja MongoDB yang ada, read/write termasuk pola, volume data, dan baseline kinerja.
- Migrasi indeks— Analisis strategi untuk mengekstraksi dan mengubah indeks MongoDB untuk kinerja optimal di Amazon DocumentDB.
- Migrasi pengguna— Detail pendekatan untuk memigrasi pengguna database, peran, dan kontrol akses ke Amazon DocumentDB.
- Migrasi data— Mencakup berbagai metode untuk migrasi data yang digunakan AWS DMS, termasuk full load dan change data capture (CDC).
- Memantau— Detail berbagai pendekatan pemantauan saat bermigrasi menggunakan DMS atau alat asli.
- ValidasiMenyediakan prosedur untuk pemeriksaan integritas data, validasi fungsional, dan perbandingan kinerja pasca-migrasi.

Dengan mengikuti panduan dalam runbook ini, tim dapat memastikan transisi yang lancar, aman, dan efisien ke Amazon DocumentDB, sambil menjaga fungsionalitas aplikasi dan meminimalkan risiko.

Kompatibilitas

Topik

- Kompatibilitas fitur inti
- Alat penilaian kompatibilitas Amazon DocumentDB

Saat bermigrasi dari MongoDB ke Amazon DocumentDB, penilaian awal yang menyeluruh dan pemeriksaan kompatibilitas fitur sangat penting untuk migrasi yang berhasil. Proses ini dimulai dengan inventaris lengkap fitur MongoDB Anda, termasuk operator pipeline agregasi, pola kueri, indeks, dan model data.

Karena Amazon DocumentDB kompatibel dengan MongoDB 3.6, 4.0, 5.0, dan 8.0 API, aplikasi yang menggunakan fitur yang lebih baru mungkin memerlukan refactoring. MongoDB-specific Area kritis untuk dievaluasi termasuk mekanisme sharding (Amazon DocumentDB menggunakan pendekatan yang berbeda), implementasi transaksi, fungsionalitas aliran perubahan, dan jenis indeks (terutama indeks jarang dan sebagian).

Karakteristik kinerja juga berbeda, dengan Amazon DocumentDB dioptimalkan untuk beban kerja perusahaan dengan kinerja yang dapat diprediksi. Pengujian harus melibatkan menjalankan beban kerja yang representatif terhadap kedua sistem untuk mengidentifikasi pola kueri yang mungkin memerlukan pengoptimalan.

Memantau rencana eksekusi untuk mendeteksi potensi kesenjangan kinerja penting selama fase penilaian. Ini membantu membuat peta jalan migrasi yang jelas, mengidentifikasi perubahan aplikasi yang diperlukan dan menetapkan jadwal yang realistis untuk transisi yang mulus.

Kompatibilitas fitur inti

Dukungan fitur yang komprehensif

- Operasi CRUD - Nikmati dukungan penuh untuk semua operasi pembuatan, baca, perbarui, dan hapus dasar, termasuk operator massal dan kueri - menyediakan kompatibilitas aplikasi yang mulus.
- Kemampuan pengindeksan yang kaya — Manfaatkan dukungan komprehensif untuk indeks bidang tunggal, gabungan, TTL, parsi, jarang, dan 2dsphere, untuk mengoptimalkan kinerja kueri dan indeks teks Anda (versi 5) untuk pencarian berbasis teks.
- Enterprise-grade replikasi — Manfaatkan mekanisme failover otomatis yang kuat dengan replika baca untuk ketersediaan tinggi yang unggul tanpa overhead operasional.
- Solusi pencadangan lanjutan - Beristirahatlah dengan sistem pencadangan otomatis yang menampilkan Point-in-Time Pemulihan (PITR) dan snapshot manual sesuai permintaan untuk perlindungan data.

Ditingkatkan AWS-fitur terintegrasi

- Agregasi yang efisien — Manfaatkan tahapan agregasi yang paling umum digunakan (`$match`,,,, `$group` `$sort` `$project`, dll.) Dengan kinerja yang dioptimalkan untuk beban kerja perusahaan.
- Dukungan transaksi - Menerapkan transaksi multi-dokumen dan multi-koleksi, sempurna untuk sebagian besar kebutuhan aplikasi bisnis.

- Real-time Pelacakan data — Aktifkan aliran perubahan dengan perintah sederhana dan tingkatkan periode retensi aliran perubahan melalui pengaturan grup parameter sederhana untuk pemantauan perubahan data waktu nyata.
- Location-based layanan - Menerapkan aplikasi geospasial dengan dukungan untuk `$geoNear` operator dan indeks 2dsphere.
- Kemampuan pencarian teks - Memanfaatkan fungsionalitas pencarian teks bawaan untuk kebutuhan penemuan konten.

Keunggulan arsitektur modern

- Cloud-native desain — Nikmati arsitektur AWS yang dioptimalkan yang menggantikan fitur lama seperti MapReduce dengan operasi pipa agregasi yang lebih efisien.
- Keamanan yang ditingkatkan — Manfaat dari AWS Identity and Access Management (IAM),,, otentikasi X.509 sertifikat SCRAM-SHA-1 SCRAM-SHA-256, dan otentikasi berbasis kata sandi.
- Kinerja yang dapat diprediksi - Rasakan kinerja konsisten yang dioptimalkan secara khusus untuk beban kerja perusahaan.

Untuk gambaran menyeluruh tentang kemampuan Amazon DocumentDB, lihat API MongoDB yang didukung, operasi, dan tipe data di Amazon DocumentDB dan Perbedaan fungsional: Amazon DocumentDB dan MongoDB untuk memaksimalkan potensi database Anda.

Amazon DocumentDB tidak mendukung semua indeks yang ditawarkan oleh MongoDB. Kami menyediakan alat indeks gratis untuk memeriksa kompatibilitas. Kami merekomendasikan menjalankan alat indeks untuk menilai ketidakcocokan dan merencanakan solusi yang sesuai.

Alat penilaian kompatibilitas Amazon DocumentDB

Alat Kompatibilitas MongoDB ke Amazon DocumentDB adalah utilitas sumber terbuka yang tersedia GitHub yang membantu mengevaluasi kompatibilitas beban kerja MongoDB dengan Amazon DocumentDB dengan menganalisis log MongoDB atau kode sumber aplikasi.

Fitur utama

- Mengidentifikasi pola penggunaan API MongoDB di beban kerja Anda
- Menandai potensi masalah kompatibilitas sebelum migrasi
- Menghasilkan laporan kompatibilitas terperinci dengan rekomendasi
- Tersedia sebagai utilitas mandiri yang dapat dijalankan secara lokal

Metode penilaian

Log-based penilaian

- Kelebihan:
  - Menangkap perilaku runtime aktual dan pola kueri
  - Mengidentifikasi frekuensi penggunaan dunia nyata dan karakteristik kinerja
  - Mendeteksi kueri dinamis yang mungkin tidak terlihat dalam kode sumber
  - Tidak diperlukan akses ke kode sumber aplikasi
- Kontra:
  - Memerlukan akses ke log MongoDB dengan profil diaktifkan
  - Hanya menangkap operasi yang terjadi selama periode penebangan
  - Mungkin melewatkan fitur yang jarang digunakan atau beban kerja musiman

Analisis kode sumber

- Kelebihan:
  - Cakupan komprehensif dari semua operasi MongoDB potensial dalam basis kode
  - Dapat mengidentifikasi masalah di jalur kode yang jarang dieksekusi
  - Mendeteksi logika sisi klien yang mungkin terpengaruh oleh perbedaan Amazon DocumentDB
  - Tidak perlu menjalankan aplikasi untuk melakukan penilaian
- Kontra:
  - Dapat menandai kode yang ada tetapi tidak pernah dieksekusi dalam produksi
  - Membutuhkan akses ke kode sumber aplikasi lengkap
  - Kemampuan terbatas untuk menganalisis kueri yang dibangun secara dinamis

Untuk hasil terbaik, sebaiknya gunakan kedua metode penilaian bila memungkinkan untuk mendapatkan gambaran lengkap tentang tantangan kompatibilitas sebelum migrasi.

Penemuan beban kerja

Migrasi dari MongoDB ke Amazon DocumentDB memerlukan pemahaman menyeluruh tentang beban kerja database yang ada. Penemuan beban kerja adalah proses menganalisis pola

penggunaan database Anda, struktur data, kinerja kueri, dan dependensi operasional untuk memastikan transisi yang mulus dengan gangguan minimal. Bagian ini menguraikan langkah-langkah kunci yang terlibat dalam penemuan beban kerja untuk memfasilitasi migrasi yang efektif dari MongoDB ke Amazon DocumentDB.

Topik

- Menilai penerapan MongoDB yang ada
- Mengidentifikasi perbedaan model data
- Analisis kueri dan kinerja
- Tinjauan keamanan dan kontrol akses
- Pertimbangan operasional dan pemantauan

Menilai penerapan MongoDB yang ada

Sebelum migrasi, sangat penting untuk mengevaluasi lingkungan MongoDB saat ini, termasuk:

- Arsitektur cluster — Identifikasi jumlah node, set replika, dan konfigurasi sharding. Saat bermigrasi dari MongoDB ke Amazon DocumentDB, memahami konfigurasi sharding MongoDB Anda penting karena Amazon DocumentDB tidak mendukung sharding yang dikendalikan pengguna. Aplikasi yang dirancang untuk lingkungan MongoDB sharded akan membutuhkan perubahan arsitektur, karena Amazon DocumentDB menggunakan pendekatan penskalaan yang berbeda dengan arsitektur berbasis penyimpanan. Anda harus menyesuaikan strategi distribusi data Anda dan mungkin mengkonsolidasikan koleksi sharded saat pindah ke Amazon DocumentDB.
- Penyimpanan dan volume data — Ukur ukuran data total dan ukuran indeks klaster Anda. Lengkapi ini dengan alat ulasan Oplog untuk memahami pola tulis dan kecepatan pertumbuhan data. Untuk informasi selengkapnya tentang ukuran cluster Anda, lihatUkuran instans.
- Pola beban kerja - Menganalisis throughput baca dan tulis, frekuensi eksekusi kueri, dan efisiensi pengindeksan.
- Ketergantungan operasional - Dokumentasikan semua aplikasi, layanan, dan integrasi yang mengandalkan MongoDB.

Mengidentifikasi perbedaan model data

Meskipun Amazon DocumentDB, MongoDB-compatible ada perbedaan dalam fitur yang didukung, seperti:

- Transaksi — Amazon DocumentDB mendukung transaksi ACID tetapi dengan beberapa. Batasan
- Desain skema - Pastikan struktur dokumen, dokumen yang disematkan, dan referensi selaras dengan praktik terbaik Amazon DocumentDB.

Analisis kueri dan kinerja

Memahami perilaku kueri membantu mengoptimalkan migrasi dan performa pasca-migrasi. Bidang utama untuk dianalisis meliputi:

- Kueri lambat - Identifikasi kueri dengan waktu eksekusi tinggi menggunakan alat profil MongoDB.
- Pola kueri — Mengkategorikan jenis kueri umum, termasuk operasi dan agregasi CRUD.
- Penggunaan indeks — Menilai apakah indeks digunakan secara efektif atau perlu optimasi di Amazon DocumentDB. Untuk menilai penggunaan indeks dan mengoptimalkan kinerja di Amazon DocumentDB, gunakan tahap pipeline agregasi `$indexStats` yang dikombinasikan dengan `explain()` metode pada kueri penting Anda. Mulailah dengan menjalankan `db.collection.aggregate([{$indexStats{}}])` untuk mengidentifikasi indeks mana yang sedang digunakan. Anda dapat melakukan analisis yang lebih rinci dengan mengeksekusi kueri yang paling sering Anda gunakan. `explainPlan`
- Distribusi konkurensi & beban kerja - Mengevaluasi rasio baca dan tulis, penyatuan koneksi, dan kemacetan kinerja.

Tinjauan keamanan dan kontrol akses

Otentikasi dan otorisasi

- MongoDB RBAC ke Amazon DocumentDB IAM dan RBAC - Peta pengguna kontrol akses berbasis peran MongoDB dan peran ke kebijakan (IAM) dan pengguna otentikasi Amazon DocumentDB SCRAM. AWS Identity and Access Management
- Strategi migrasi pengguna — Rencanakan untuk memigrasikan pengguna database, peran khusus, dan hak istimewa ke mekanisme autentikasi yang didukung Amazon DocumentDB.
- Perbedaan hak istimewa - Identifikasi hak istimewa MongoDB tanpa setara Amazon DocumentDB langsung (misalnya, peran administrasi klaster).
- Otentikasi aplikasi - Perbarui string koneksi dan manajemen kredensi untuk kebijakan kata sandi Amazon DocumentDB. Anda dapat menggunakan manajer rahasia untuk menyimpan kredensil Anda dan memutar kata sandi.

- Manajemen akun layanan - Menetapkan proses untuk mengelola kredensi akun layanan di. AWS Secrets Manager
- Implementasi hak istimewa terkecil - Tinjau dan perbaiki kontrol akses untuk menerapkan prinsip hak istimewa terkecil di lingkungan baru.

Enkripsi

Pastikan enkripsi saat istirahat dan dalam perjalanan selaras dengan persyaratan kepatuhan.

Konfigurasi jaringan

Rencanakan pengaturan Virtual Private Cloud (VPC) dan aturan grup keamanan.

Pertimbangan operasional dan pemantauan

Untuk menjaga keandalan sistem, penemuan beban kerja juga harus mencakup:

- Strategi Backup dan Restore — Evaluasi metode backup yang ada dan kemampuan backup Amazon DocumentDB.
- AWS Backup Integrasi — AWS Backup Memanfaatkan manajemen cadangan terpusat di seluruh AWS layanan termasuk Amazon DocumentDB.
- CloudWatch metrik — Petakan metrik pemantauan MongoDB ke metrik Amazon DocumentDB untuk CPU, CloudWatch memori, koneksi, dan penyimpanan.
- Performance Insights — Menerapkan Amazon DocumentDB Performance Insights untuk memvisualisasikan beban database dan menganalisis masalah kinerja dengan analitik kueri terperinci.
- Profiler - Konfigurasikan profiler Amazon DocumentDB untuk menangkap operasi yang berjalan lambat (mirip dengan profiler MongoDB tetapi dengan pengaturan Amazon). DocumentDB-specific
  - Aktifkan melalui grup parameter dengan ambang batas yang sesuai.
  - Menganalisis data profiler untuk mengidentifikasi peluang optimasi
- CloudWatch Acara - Siapkan pemantauan berbasis peristiwa untuk peristiwa klaster Amazon DocumentDB.
  - Konfigurasikan notifikasi untuk acara pencadangan, jendela pemeliharaan, dan kegagalan.
  - Integrasikan dengan Amazon SNS untuk peringatan dan respons otomatis AWS Lambda .
- Audit logging - Merencanakan konfigurasi pencatatan audit untuk melacak aktivitas pengguna dan peristiwa yang relevan dengan keamanan.

- Pemantauan yang disempurnakan - Aktifkan pemantauan yang disempurnakan untuk OS-level metrik granular pada interval 1 detik.

Migrasi indeks

Migrasi dari MongoDB ke Amazon DocumentDB melibatkan transfer tidak hanya data tetapi juga indeks untuk mempertahankan kinerja kueri dan mengoptimalkan operasi database. Bagian ini menguraikan proses langkah demi langkah terperinci untuk memigrasikan indeks dari MongoDB ke Amazon DocumentDB sambil memastikan kompatibilitas dan efisiensi.

Menggunakan alat indeks Amazon DocumentDB

Kloning alat indeks

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

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

Ekspor indeks dari MongoDB (jika bermigrasi dari MongoDB)

```
python3 migrationtools/documentdb_index_tool.py --dump-indexes --dir
 mongodb_index_export --uri
'mongodb://localhost:27017'
```

Ekspor indeks dari Amazon DocumentDB (jika bermigrasi dari Amazon DocumentDB)

```
python3 migrationtools/documentdb_index_tool.py --dump-indexes --dir docdb_index_export
 --uri
'mongodb://user:password@mydocdb.cluster-cdtjj00yfi95.eu-west-
2.docdb.amazonaws.com:27017/?tls=true&tlsCAFile=rds-combined-ca-
bundle.pem&replicaSet=rs0&retryWrites=false'
```

Indeks impor

```
python3 migrationtools/documentdb_index_tool.py --restore-indexes --skip-incompatible
 --dir
mongodb_index_export --uri 'mongodb://user:password@mydocdb.cluster-cdtjj00yfi95.eu-
west-
```

```
2.docdb.amazonaws.com:27017/?tls=true&tlsCAFile=rds-combined-ca-
bundle.pem&replicaSet=rs0&retryWrites=false'
```

Verifikasi indeks

```
python3 migrationtools/documentdb_index_tool.py --show-issues --dir
 mongodb_index_export
```

Migrasi pengguna

Migrasi pengguna dari MongoDB ke Amazon DocumentDB sangat penting untuk menjaga kontrol akses, otentikasi, dan keamanan database. Bagian ini menguraikan langkah-langkah terperinci untuk berhasil memigrasikan pengguna MongoDB sambil mempertahankan peran dan izin mereka menggunakan alat pengguna ekspor Amazon DocumentDB.

Menggunakan alat pengguna ekspor Amazon DocumentDB

Pengguna `Export Users tool`ekspor dan peran dari MongoDB atau Amazon JavaScript DocumentDB ke file, yang kemudian dapat digunakan untuk membuatnya kembali di cluster lain.

Prasyarat

```
# 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
```

Langkah 1: Ekspor pengguna dan peran

```
# Export users and roles to JavaScript files
python3 docdbExportUsers.py \
--users-file mongodb-users.js \
--roles-file mongodb-roles.js \
--uri "mongodb://admin:password@source-host:27017/"
```

Langkah 2: Edit File Pengguna

```
// Example of how to update the users.js file
// Find each user creation statement and add the password
db.getSiblingDB("admin").createUser({
user: "appuser",
// Add password here
pwd: "newpassword",
roles: [
{ role: "readWrite", db: "mydb" }
]
})
```

Langkah 3: Kembalikan Peran Kustom ke Amazon DocumentDB

```
# Import roles first
mongo --ssl \
--host target-host:27017 \
--sslCAFile rds-combined-ca-bundle.pem \
--username admin \
--password password \
mongodb-roles.js
```

Langkah 4: Kembalikan Pengguna ke Amazon DocumentDB

```
# Import users after roles are created
mongo --ssl \
--host target-host:27017 \
--sslCAFile rds-combined-ca-bundle.pem \
--username admin \
--password password \
mongodb-users.js
```

Catatan penting

- Kata sandi tidak diekspor untuk alasan keamanan dan harus ditambahkan secara manual ke file users.js.
- Peran harus diimpor sebelum pengguna untuk memastikan penetapan peran yang tepat.
- Alat ini menghasilkan JavaScript file yang dapat langsung dieksekusi dengan shell mongo.
- Peran khusus dan hak istimewa mereka dipertahankan selama migrasi.
- Pendekatan ini memungkinkan peninjauan dan modifikasi izin pengguna sebelum mengimpor.

Metode ini menyediakan pendekatan yang aman dan fleksibel untuk memigrasikan pengguna dan peran dari MongoDB ke Amazon DocumentDB sambil memungkinkan pengaturan ulang kata sandi selama proses migrasi.

Migrasi data

Topik

- Migrasi online
- Migrasi offline
- Prasyarat
- Siapkan cluster Amazon DocumentDB
- Lakukan dump data (mongodump)
- Transfer file dump ke lingkungan restorasi
- Mengembalikan data ke Amazon DocumentDB (mongorestore)

Migrasi online

Bagian ini memberikan langkah-langkah rinci untuk melakukan migrasi online dari MongoDB ke Amazon DocumentDB AWS DMS menggunakan untuk mengaktifkan downtime minimal dan replikasi berkelanjutan. Untuk memulai, Anda menyiapkan klaster Amazon DocumentDB sebagai target dan memastikan instans MongoDB Anda dikonfigurasi dengan benar sebagai sumber, biasanya memerlukan mode set replika untuk pengambilan data perubahan. Selanjutnya, Anda membuat instance replikasi DMS dan menentukan titik akhir sumber dan target dengan detail koneksi yang diperlukan. Setelah memvalidasi titik akhir, Anda mengonfigurasi dan memulai tugas migrasi yang dapat menyertakan pemuatan data penuh, replikasi berkelanjutan, atau keduanya.

Konfigurasikan target (Amazon DocumentDB)

> **Note**
>
> Jika Anda sudah menyediakan klaster Amazon DocumentDB untuk bermigrasi, Anda dapat melewati langkah ini.

Buat grup parameter kustom

Lihat Konsol Manajemen AWS atau AWS CLI prosedur diMembuat grup parameter cluster Amazon DocumentDB.

Buat cluster Amazon DocumentDB

> **Note**
>
> Meskipun ada prosedur lain untuk membuat cluster Amazon DocumentDB dalam panduan ini, langkah-langkah di bagian ini berlaku khusus untuk tugas migrasi sejumlah besar data ke cluster baru.

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Klaster.

   > **Tip**
   >
   > Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu (≡) di pojok kiri atas halaman.

3. Di konsol manajemen Amazon DocumentDB, di bawah Klaster, pilih Buat.
4. Pada halaman cluster Create Amazon DocumentDB, di bagian tipe Cluster, Instance-based pilih cluster (ini adalah opsi default).
5. Di bagian konfigurasi Cluster:
   - Untuk pengidentifikasi Cluster, masukkan nama unik, seperti**`mydocdbcluster`**. Perhatikan bahwa konsol akan mengubah semua nama cluster menjadi huruf kecil terlepas dari bagaimana mereka dimasukkan.
   - Untuk versi Engine, pilih 5.0.0.
6. Di bagian konfigurasi penyimpanan Cluster, biarkan pengaturan Standar Amazon DocumentDB apa adanya (ini adalah opsi default).
7. Di bagian konfigurasi Instans:
   - Untuk kelas instans DB, pilih kelas yang dioptimalkan memori (termasuk kelas r) (ini adalah default).

- Untuk kelas Instance, pilih kelas instance berdasarkan beban kerja. Contoh:
  - db.r6g.large: untuk beban kerja yang lebih kecil
  - db.r6g.4xlarge: untuk beban kerja yang lebih besar

  Sebagai praktik terbaik, sebaiknya pilih instans sebesar yang Anda mampu untuk throughput beban penuh terbaik, dan turunkan skala setelah migrasi selesai.
- Untuk Jumlah instance, pilih 1 instance. Memilih satu instans membantu meminimalkan biaya. Kami menyarankan Anda menskalakan ke tiga instans untuk ketersediaan tinggi setelah migrasi muatan penuh selesai.

8. Di bagian Autentikasi, masukkan nama pengguna untuk pengguna utama, lalu pilih Self managed. Masukkan kata sandi, lalu konfirmasikan.
9. Di bagian Pengaturan jaringan, pilih grup VPC dan subnet, lalu konfigurasikan grup keamanan VPC. Pastikan grup keamanan Amazon DocumentDB Anda mengizinkan koneksi masuk dari grup keamanan instans DMS dengan memperbarui aturan masuk.
10. Di Encryption-at-restbagian ini, aktifkan enkripsi (disarankan) dan pilih atau masukkan kunci KMS.
11. Di bagian Backup, atur periode retensi cadangan (1-35 hari).
12. Tinjau konfigurasi Anda dan pilih Buat klaster.

    Waktu penyebaran biasanya memakan waktu antara 10 dan 15 menit,

Konfigurasikan sumber

MongoDB dan Amazon DocumentDB keduanya dapat berfungsi sebagai sumber migrasi, tergantung pada skenario Anda:

- MongoDB sebagai sumber - Umum saat bermigrasi dari MongoDB lokal atau yang dikelola sendiri ke Amazon DocumentDB atau layanan database lainnya. AWS Memerlukan berjalan dalam mode set replika dengan oplog berukuran cukup (pastikan ukurannya untuk menahan semua operasi selama Beban Penuh) untuk mendukung pengambilan data perubahan selama migrasi.
- Amazon DocumentDB sebagai sumber - Biasanya digunakan untuk replikasi lintas wilayah, peningkatan versi, atau migrasi ke layanan database lain seperti MongoDB Atlas. Mengaktifkan aliran perubahanDiperlukan dengan menyetel `change_stream_log_retention_duration` parameter dalam grup parameter cluster untuk menangkap perubahan yang sedang berlangsung selama migrasi. Pastikan `change_stream_log_retention_duration` pengaturan Anda cukup besar untuk menutupi waktu yang dibutuhkan untuk menyelesaikan Beban Penuh.

Sebelum memulai migrasi, konfigurasikan sumber Anda untuk mengizinkan AWS DMS akses.

Buat pengguna MongoDB dengan izin yang tepat:

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

Konfigurasikan jaringan dan otentikasi.

Saat mengonfigurasi konektivitas jaringan untuk MongoDB ke migrasi DMS:

EC2-hosted Sumber MongoDB

- Ubah grup keamanan EC2 untuk mengizinkan lalu lintas masuk dari grup keamanan instans replikasi DMS.
- Tambahkan aturan untuk port TCP 27017 (atau port MongoDB kustom Anda).
- Gunakan ID grup keamanan instans replikasi DMS sebagai sumber untuk kontrol akses yang tepat.
- Pastikan subnet instans EC2 memiliki rute ke subnet instance replikasi DMS.

On-premises Sumber MongoDB

- Konfigurasikan firewall Anda untuk memungkinkan koneksi masuk dari alamat IP publik instans replikasi DMS.
- Jika menggunakan Direct Connect atau VPN, pastikan perutean yang tepat antara jaringan Anda dan VPC yang berisi instance DMS.
- Uji konektivitas menggunakan perintah telnet atau nc dari subnet DMS ke server MongoDB Anda.

Sumber MongoDB Atlas

- Tambahkan alamat IP contoh replikasi DMS ke daftar izin MongoDB Atlas IP.
- Konfigurasikan peering VPC antara VPC dan AWS MongoDB Atlas VPC jika Atlas berjalan. AWS
- Siapkan AWS PrivateLink untuk konektivitas pribadi (Tingkat perusahaan), jika berjalan di penyedia cloud lain.
- Buat pengguna khusus dengan read/write izin yang sesuai.

- Gunakan string koneksi MongoDB Atlas dengan Mode SSL diatur ke “verify-full”.
- Pastikan ukuran oplog yang cukup untuk durasi migrasi.

Sumber Amazon DocumentDB

Konfigurasikan grup keamanan Amazon DocumentDB sumber Anda untuk mengizinkan lalu lintas masuk dari grup keamanan instans replikasi DMS.

Buat contoh replikasi DMS

Sebaiknya gunakan DMS Buddy untuk menyediakan infrastruktur DMS Anda karena menciptakan infrastruktur migrasi yang optimal dengan pengaturan DMS dan ukuran instans yang optimal.

Jika Anda lebih suka mengkonfigurasi secara manual, ikuti langkah-langkah ini:

1. Buka konsol AWS DMS dan pilih Create replication instance.
2. Masukkan detail contoh replikasi:
    - Nama instance: Pilih nama yang unik.
    - Kelas instance: Pilih berdasarkan beban kerja. Contoh: dms.r5.large (beban kerja kecil), dms.r5.4xlarge (beban kerja besar).
    - Versi mesin: 3.5.4
    - Penyimpanan yang dialokasikan: Default adalah 100GB (meningkat jika diperlukan). Ini ditentukan oleh ukuran dokumen, updates/second dan durasi muat penuh.
    - Multi-AZ Deployment: Aktifkan ketersediaan tinggi, jika diperlukan.
    - Pilih VPC yang sama dengan Amazon DocumentDB.
    - Pastikan grup Keamanan mengizinkan lalu lintas masuk dari sumber dan Amazon DocumentDB.
3. Pilih Buat instance replikasi dan tunggu statusnya tersedia.

Buat titik akhir DMS

Buat titik akhir sumber

Untuk sumber MongoDB

1. Di konsol DMS, di panel navigasi, pilih Migrasi atau replikasi, lalu pilih Endpoints.

2. Pilih Buat titik akhir.
3. Pada halaman Create endpoint, pilih Source endpoint.
4. Di bagian konfigurasi Endpoint:

   - Masukkan pengenal Endpoint yang unik dan bermakna (misalnya, “mongodb-source”).
   - Pilih MongoDB sebagai mesin Sumber.
   - Untuk Mengakses ke basis data titik akhir, memilih Memberikan informasi untuk mengakses secara manual.
   - Untuk nama Server, masukkan*MongoDB server DNS name/IP address*.
   - Untuk Port, masukkan 27017 (port MongoDB default).
   - Untuk mode Otentikasi, pilih mode yang sesuai untuk aplikasi Anda (password/SSL) (default adalah manajer rahasia).
   - Jika mode Otentikasi adalah Kata Sandi, berikan:
     - Nama Pengguna dan Kata Sandi: Masukkan kredenal MongoDB.
     - Nama database: Nama basis data sumber Anda.
     - Mekanisme otentikasi: SCRAM-SHA-1 (default) atau mekanisme yang sesuai
5. Untuk mode Metadata, tinggalkan pengaturan default dokumen.
6. Atribut koneksi tambahan:

   - authSource=admin (jika database otentikasi berbeda)
   - Replicaset= <your-replica-set-name>(diperlukan untuk CDC)

Untuk sumber Amazon DocumentDB

1. Di konsol DMS, di panel navigasi, pilih Migrasi atau replikasi, lalu pilih Endpoints.
2. Pilih Buat titik akhir.
3. Pada halaman Create endpoint, pilih Source endpoint.
4. Di bagian konfigurasi Endpoint:

   - Masukkan pengenal Endpoint yang unik dan bermakna (misalnya, “docdb-source”).
   - Pilih Amazon DocumentDB sebagai mesin Sumber.
   - Untuk Mengakses ke basis data titik akhir, memilih Memberikan informasi untuk mengakses secara manual.

- Untuk nama Server, masukkan*source Amazon DocumentDB cluster endpoint*.
- Untuk Port, masukkan 27017 (port Amazon DocumentDB default).
- Untuk mode SSL, pilih verify-full (direkomendasikan untuk Amazon DocumentDB).
- Untuk Sertifikat CA, pilih sertifikat CA root Amazon RDS.
- Untuk mode Otentikasi, pilih mode yang sesuai untuk aplikasi Anda (password/SSL) (default adalah manajer rahasia).
- Jika mode Otentikasi adalah Kata Sandi, berikan:
  - Nama Pengguna dan Kata Sandi: Masukkan kredenal Amazon DocumentDB.
  - Nama database: Nama basis data sumber Anda.
  - Mekanisme otentikasi: SCRAM-SHA-1 (default) atau mekanisme yang sesuai

5. Untuk mode Metadata, tinggalkan pengaturan default dokumen.

Buat titik akhir target (Amazon DocumentDB)

1. Di konsol DMS, di panel navigasi, pilih Migrasi atau replikasi, lalu pilih Endpoints.
2. Pilih Buat titik akhir.
3. Pada halaman Create endpoint, pilih Target endpoint.
4. Di bagian konfigurasi Endpoint:

   - Masukkan pengenal Endpoint yang unik dan bermakna (misalnya, “docdb-target”).
   - Pilih Amazon DocumentDB sebagai mesin Target.
   - Untuk Access to endpoint database, pilih metode yang ingin Anda gunakan untuk mengautentikasi akses ke database:
     - Jika Anda memilih AWS Secrets Manager, pilih rahasia tempat Anda menyimpan kredensi Amazon DocumentDB Anda di bidang Rahasia.
     - Jika Anda memilih Menyediakan informasi akses secara manual:
       - Untuk nama Server, masukkan*target Amazon DocumentDB cluster endpoint*.
       - Untuk Port, masukkan 27017 (port Amazon DocumentDB default).
       - Untuk mode SSL, pilih verify-full (direkomendasikan untuk Amazon DocumentDB).
       - Untuk Sertifikat CA, unduh dan tentukan bundel sertifikat CA untuk verifikasi SSL.
       - Untuk mode Otentikasi, pilih mode yang sesuai untuk aplikasi Anda (password/SSL) (default adalah manajer rahasia).

- Jika mode Otentikasi adalah Kata Sandi, berikan:
  - Nama Pengguna dan Kata Sandi: Masukkan kredenal Amazon DocumentDB.
  - Nama database: Nama basis data sumber Anda.
  - Mekanisme otentikasi: SCRAM-SHA-1 (default) atau mekanisme yang sesuai

5. Untuk mode Metadata, tinggalkan pengaturan default dokumen.

Buat tugas replikasi

1. Di konsol DMS, di panel navigasi, pilih Migrasi atau replikasi, lalu pilih Tugas.
2. Pilih Buat tugas.
3. Pada halaman Buat tugas, di bagian Konfigurasi tugas:
   - Masukkan pengidentifikasi Tugas yang unik dan bermakna (misalnya, “mongodb-docdb-replication”).
   - Pilih titik akhir sumber yang Anda buat sebelumnya di menu drop-down titik akhir database Sumber.
   - Pilih titik akhir target yang Anda buat sebelumnya di menu tarik-turun titik akhir database Target.
   - Untuk jenis Tugas, pilih Migrasi dan replikasi.
4. Di bagian Pengaturan:
   - Untuk mode persiapan tabel Target, tinggalkan pengaturan default.
   - Untuk tugas Stop setelah beban penuh selesai, tinggalkan pengaturan default.
   - Untuk pengaturan kolom LOB, biarkan pengaturan mode LOB Terbatas apa adanya.
   - Untuk validasi Data, biarkan pengaturan default Matikan.
   - Untuk log Tugas, centang kotak Aktifkan CloudWatch log.
   - Untuk Batch-optimized menerapkan, biarkan pengaturan default tidak dicentang (mati).
5. Kembali di bagian atas Pengaturan tugas, dalam mode Pengeditan, pilih editor JSON dan atur atribut berikut:

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

```
    "MaxFullLoadSubTasks": 16
  }
}
```

6. Di bagian Pemetaan tabel, tambahkan aturan pemilihan baru:
   - Untuk nama Skema, tambahkan database sumber untuk bermigrasi. Gunakan% untuk menentukan beberapa database.
   - Untuk nama tabel Skema, tambahkan koleksi sumber untuk bermigrasi. Gunakan% untuk menentukan beberapa koleksi.
   - Untuk Tindakan, tinggalkan pengaturan default Sertakan
7. Untuk koleksi besar (lebih dari 100GB), tambahkan aturan pengaturan Tabel:
   - Untuk nama Skema, tambahkan database sumber untuk bermigrasi. Gunakan% untuk menentukan beberapa database.
   - Untuk nama tabel Skema, tambahkan koleksi sumber untuk bermigrasi. Gunakan% untuk menentukan beberapa koleksi.
   - Untuk Jumlah partisi, masukkan 16 (harus kurang dari`MaxFullLoadSubTask`).
8. Di bagian penilaian Premigrasi, pastikan dimatikan.

Migrasi offline

Bagian ini menguraikan proses untuk melakukan migrasi offline dari instance MongoDB yang dikelola sendiri ke Amazon DocumentDB menggunakan alat MongoDB asli: dan. `mongodump` `mongorestore`

Prasyarat

Sumber persyaratan MongoDB

- Akses ke instance MongoDB sumber dengan izin yang sesuai.
- Instal`mongodump`. jika diperlukan (diinstal selama instalasi MongoDB).
- Pastikan ada cukup ruang disk untuk file dump.

Targetkan persyaratan Amazon DocumentDB

- Pastikan Anda memiliki klaster Amazon DocumentDB yang disediakan.

- Pastikan ada instans EC2 di VPC yang sama dengan Amazon DocumentDB untuk memfasilitasi migrasi.
- Konektivitas jaringan harus tersedia antara lingkungan sumber Anda dan Amazon DocumentDB.
- mongorestore harus diinstal pada instance EC2 migrasi.
- Izin IAM yang sesuai harus dikonfigurasi untuk mengakses Amazon DocumentDB,

Persyaratan umum

- AWS CLI harus dikonfigurasi (jika menggunakan AWS layanan untuk penyimpanan menengah)
- Bandwidth yang cukup harus tersedia untuk transfer data.
- Jendela downtime harus disetujui (jika melakukan migrasi langsung, pertimbangkan pendekatan lain)

Siapkan cluster Amazon DocumentDB

Buat cluster Amazon DocumentDB di: AWS

- Tepat ukuran instans berdasarkan beban kerja Anda.
- Konfigurasikan VPC, subnet, dan grup keamanan.
- Aktifkan parameter yang diperlukan melalui grup parameter.

Lakukan dump data (mongodump)

Pilih salah satu opsi berikut untuk membuat file dump:

- Opsi 1: Dasar

```
mongodump --
uri="mongodb://<source_user>:<source_password>@<source_host>:<source_port>/
<database>" --
out=/path/to/dump
```

- Opsi 2: Kontrol dan kinerja yang lebih baik

```
mongodump \
--uri="mongodb://<source_user>:<source_password>@<sourcehost>:<source_port>" \
```

```
--out=/path/to/dump \
--gzip \# Compress output
--numParallelCollections=4 \# Parallel collections dump
--ssl \# If using SSL
--authenticationDatabase=admin \ # If auth is required
--readPreference=secondaryPreferred # If replica set
```

- Opsi 3: Database besar

```
mongodump \
--host=<source_host> \
--port=<source_port> \
--username=<source_user> \
--password=<source_password> \
--db=<specific_db> \# Only dump specific DB
--collection=<specific_collection> \ # Only dump specific collection
--query='{ "date": { "$gt": "2020-01-01" } }' \ # Filter documents
--archive=/path/to/archive.gz \# Single archive output
--gzip \
--ssl
```

Transfer file dump ke lingkungan restorasi

Pilih metode yang sesuai berdasarkan ukuran dump Anda:

- Kecil - Langsung salin ke mesin migrasi Anda (instans EC2 yang Anda buat sebelumnya):

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

- Sedang - Gunakan Amazon S3 sebagai penyimpanan perantara:

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

- Besar — Untuk database yang sangat besar, pertimbangkan AWS DataSync atau transfer fisik.

Mengembalikan data ke Amazon DocumentDB (mongorestore)

Sebelum memulai proses pemulihan, buat indeks di Amazon DocumentDB. Anda dapat menggunakan alat Amazon DocumentDB Index untuk mengekspor dan mengimpor indeks.

Pilih salah satu opsi berikut untuk memulihkan data:

- Opsi 1: Pemulihan dasar

```
mongorestore --uri="mongodb://<docdb_user>:<docdb_password>@<docdb_endpoint>:27017"
/path/to/dump
```

- Opsi 2: Kontrol dan kinerja yang lebih baik

```
mongorestore \
--uri="mongodb://<docdb_user>:<docdb_password>@<docdb_endpoint>:27017" \
--ssl \
--sslCAFile=/path/to/rds-combined-ca-bundle.pem \ # DocumentDB CA cert
--gzip \# If dumped with gzip
--numParallelCollections=4 \# Parallel restoration
--numInsertionWorkersPerCollection=4 \# Parallel documents insertion
--noIndexRestore \# skip indexes as they are pre-created
/path/to/dump
```

- Opsi 3: Database besar atau kontrol khusus

```
mongorestore \
--host=<docdb_endpoint> \
--port=27017 \
--username=<docdb_user> \
--password=<docdb_password> \
--ssl \
--sslCAFile=/path/to/rds-combined-ca-bundle.pem \
--archive=/path/to/archive.gz \# If using archive format
--gzip \
--nsInclude="db1.*" \# Only restore specific namespaces
--nsExclude="db1.sensitive_data" \ # Exclude specific collections if needed
--noIndexRestore \# skip indexes as they are pre-created
--writeConcern="{w: 'majority'}" # Ensure write durability
```

Memantau

Bagian ini menyediakan proses pemantauan terperinci untuk melacak kemajuan, kinerja, dan kesehatan migrasi yang sedang berlangsung dari:

MongoDB ke Amazon DocumentDB

atau

Amazon DocumentDB ke Amazon DocumentDB

Langkah-langkah pemantauan berlaku terlepas dari metode migrasi (AWS DMS, mongodump/mongorestore, atau alat lainnya).

AWS DMS Pemantauan migrasi (jika ada)

Pantau CloudWatch metrik utama berikut:

Metrik fase beban penuh

- FullLoadThroughputBandwidthTarget— Bandwidth jaringan (KB/second) selama beban penuh
- FullLoadThroughputRowsTarget— Jumlah rows/documents dimuat per detik
- FullLoadThroughputTablesTarget— Jumlah tables/collections selesai per menit
- FullLoadProgressPercent— Persentase beban penuh selesai
- TablesLoaded— Jumlah yang tables/collections berhasil dimuat
- TablesLoading— Jumlah pemuatan tables/collections saat ini
- TablesQueued— Jumlah tables/collections menunggu untuk dimuat
- TablesErrored— Jumlah tables/collections yang gagal dimuat

Metrik fase CDC

- CDCLatencyTarget— Waktu tunda (detik) antara perubahan sumber dan aplikasi target
- CDCLatencySource— Waktu tunda (detik) antara perubahan sumber dan DMS membacanya
- CDCThroughputRowsTarget— Baris per detik diterapkan selama replikasi yang sedang berlangsung
- CDCThroughputBandwidthTarget— Bandwidth jaringan (KB/second) selama CDC
- CDCIncomingChanges— Jumlah peristiwa perubahan yang diterima dari sumber
- CDCChangesMemoryTarget— Memori yang digunakan (MB) untuk menyimpan perubahan di sisi target

Metrik sumber daya

- CPUUutilization — penggunaan CPU dari instance replikasi
- FreeableMemory— Memori yang tersedia pada contoh replikasi

- FreeStorageSpace— Penyimpanan yang tersedia pada contoh replikasi
- NetworkTransmitThroughput— Throughput jaringan untuk contoh replikasi
- NetworkReceiveThroughput— Throughput jaringan untuk contoh replikasi

Metrik kesalahan

- ErrorsCount— Jumlah total kesalahan selama migrasi
- TableErrorsCount— Jumlah kesalahan khusus tabel
- RecordsErrorsCount— Jumlah kesalahan khusus catatan

Buat CloudWatch alarm untuk metrik penting seperti `CDCLatencyTarget` dan `CPUUtilization` untuk menerima pemberitahuan jika kinerja migrasi menurun.

Log DMS (CloudWatch log)

1. Buka konsol Amazon CloudWatch Logs.
2. Temukan dan pilih pada grup log Anda. Ini akan terlihat mirip dengan “dms-tasks —”.
3. Cari aliran log yang mungkin berisi informasi kesalahan:
   - Streaming dengan “kesalahan” dalam nama
   - Streaming dengan ID tugas atau nama titik akhir
   - Aliran log terbaru selama migrasi Anda
4. Dalam aliran ini, cari kata kunci seperti:
   - “kesalahan”
   - “pengecualian”
   - “Gagal”
   - “peringatan”

Status tugas DMS (menggunakan AWS CLI)

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

Aliran status yang diharapkan:

membuat → siap → berjalan → berhenti → berhenti (atau gagal)

Pantau menggunakan **`docdb-dashboarder`**

`docdb-dashboarder`Alat ini menyediakan pemantauan komprehensif untuk cluster Amazon DocumentDB dengan secara otomatis CloudWatch menghasilkan dasbor dengan metrik kinerja penting. Dasbor ini menampilkan metrik tingkat klaster kritis (lag replika, penghitung operasi), metrik tingkat instance (CPU, memori, koneksi), dan metrik penyimpanan (penggunaan volume, penyimpanan cadangan). Untuk skenario migrasi, alat ini menawarkan dasbor khusus yang melacak kemajuan migrasi dengan metrik seperti lag replikasi CDC dan kecepatan operasi. Dasbor dapat memantau beberapa cluster secara bersamaan dan menyertakan dukungan untuk NVMe-backed instance. Dengan memvisualisasikan metrik ini, tim dapat secara proaktif mengidentifikasi kemacetan kinerja, mengoptimalkan alokasi sumber daya, dan memastikan kelancaran pengoperasian penerapan Amazon DocumentDB mereka. Alat ini menghilangkan kebutuhan akan pembuatan dasbor manual sambil memberikan pemantauan yang konsisten di semua lingkungan. Untuk petunjuk penyiapan dan opsi konfigurasi lanjutan, lihat repositori Alat Dasbor Amazon DocumentDB. GitHub

Validasi

Topik

- Daftar periksa validasi
- Skema dan validasi indeks
- Pengambilan sampel data dan validasi tingkat lapangan
- Validasi menggunakan alat DataDiffer

Bagian ini menyediakan proses validasi terperinci untuk memastikan konsistensi data, integritas, dan kompatibilitas aplikasi setelah bermigrasi dari:

MongoDB ke Amazon DocumentDB

atau

Amazon DocumentDB ke Amazon DocumentDB

Langkah-langkah validasi berlaku terlepas dari metode migrasi (AWS DMS, mongodump/mongorestore, atau alat lainnya).

Daftar periksa validasi

Verifikasi bahwa jumlah dokumen di setiap koleksi cocok antara sumber dan target:

Sumber MongoDB

```
mongo --host <source_host> --port <port> --username <user> -- password <password> --
eval
"db.<collection>.count()"
```

Target Amazon DocumentDB

```
mongo --host <target_host> --port <port> --username <user> -- password <password> --
eval
"db.<collection>.count()"
```

Skema dan validasi indeks

Pastikan bahwa:

- semua koleksi ada di target.
- indeks direplikasi dengan benar.
- definisi skema (jika ditegakkan) identik.

Periksa koleksi (sumber vs. target)

```
mongo --host <source_host> --eval "show collections"
mongo --host <target_host> --ssl --eval "show collections"
```

periksa indeks (Sumber vs Target)

```
mongo --host <source_host> --eval" db.<collection>.getIndexes()"
mongo --host <target_host> --ssl –eval" db.<collection>.getIndexes()"
```

Bandingkan daftar koleksi untuk memastikan tidak ada koleksi yang hilang atau tambahan.

Verifikasi indeks dengan memeriksa nama indeks, definisi kunci, kendala unik, dan indeks TTL (jika ada).

Periksa aturan validasi skema (jika menggunakan validasi skema di MongoDB)

```
mongo --host <source_host> --eval" db.getCollectionInfos({name: '<collection>'})
[0].options.validator"
   mongo --host <target_host> --ssl -eval" db.getCollectionInfos({name:
 '<collection>'})[0].options.validator"
```

Pengambilan sampel data dan validasi tingkat lapangan

Anda dapat secara acak sampel dokumen dan membandingkan bidang antara sumber dan target.

Pengambilan sampel manual

Ambil lima dokumen acak (sumber):

```
mongo --host <source_host> --eval "db.<collection>.aggregate([{ \$sample: { size:
 5 } }])"
```

Ambil ID dokumen yang sama (target):

```
mongo --host <target_host> --ssl -eval "db.<collection>.find({ _id: { \$in:
 [<list_of_ids>] } })"
```

Pengambilan sampel otomatis

```
import pymongo
# Connect to source and target
source_client = pymongo.MongoClient("<source_uri>")
target_client = pymongo.MongoClient("<target_uri>", ssl=True)
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']}")
```

Validasi menggunakan alat DataDiffer

DataDiffer Alat ini menyediakan cara yang andal untuk membandingkan data antara basis data sumber dan target.

Prasyarat

Prasyarat berikut harus dipenuhi sebelum menginstal alat: DataDiffer

- Python 3.7+
- PyMongo perpustakaan
- Konektivitas jaringan ke MongoDB sumber dan target cluster Amazon DocumentDB

Penyiapan dan instalasi

Kloning repositori dan arahkan ke direktori DataDiffer

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

Instal dependensi yang diperlukan

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

Menjalankan validasi data

Buat file konfigurasi (misalnya, config.json) dengan detail koneksi

```
{
"source": {
"uri": "mongodb://username:password@source-mongodb-
host:27017/?replicaSet=rs0",
"db": "your_database",
"collection": "your_collection"
},
"target": {
"uri": "mongodb://username:password@target-docdb-
cluster.region.docdb.amazonaws.com:27017/?tls=true&tlsCAFile=global-
bundle.pem&replicaSet=rs0",
"db": "your_database",
"collection": "your_collection"
```

```
},
"options": {
"batch_size": 1000,
"threads": 4,
"sample_size": 0,
"verbose": true
}
}
```

Jalankan DataDiffer alat

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

Untuk koleksi besar, gunakan sampling untuk memvalidasi subset data

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

Untuk memvalidasi beberapa koleksi, buat file konfigurasi terpisah atau gunakan mode batch

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

Menafsirkan hasil

Alat ini akan menampilkan:

- Total dokumen dalam sumber dan target
- Jumlah dokumen yang cocok
- Jumlah dokumen yang hilang
- Jumlah dokumen dengan perbedaan
- Laporan rinci perbedaan (jika ada)

Praktik terbaik

Berikut ini adalah praktik terbaik saat menggunakan DataDiffer alat ini:

- Jalankan secara bertahap — Pertama validasi jumlah dokumen, lalu sampel dokumen kunci, dan akhirnya jalankan perbandingan lengkap, jika diperlukan.
- Periksa perbedaan skema - Amazon DocumentDB memiliki beberapa keterbatasan dibandingkan dengan MongoDB. Alat ini akan menyoroti tipe atau struktur data yang tidak kompatibel.

- Validasi selama periode tenang - Jalankan validasi saat operasi tulis minimal untuk memastikan konsistensi.
- Memantau penggunaan sumber daya — Proses perbandingan bisa intensif sumber daya. Sesuaikan ukuran batch dan jumlah utas yang sesuai.
- Validasi indeks - Setelah validasi data, pastikan semua indeks yang diperlukan telah dibuat pada klaster Amazon DocumentDB target.
- Hasil validasi dokumen — Simpan catatan hasil validasi untuk setiap koleksi sebagai bagian dari dokumentasi migrasi Anda.

Migrasi dari Couchbase Server

Topik

- Pengantar
- Perbandingan dengan Amazon DocumentDB
- Penemuan
- Perencanaan
- Migrasi
- Validasi

Pengantar

Panduan ini menyajikan poin-poin penting yang perlu dipertimbangkan saat bermigrasi dari Couchbase Server ke Amazon DocumentDB. Ini menjelaskan pertimbangan untuk fase penemuan, perencanaan, pelaksanaan, dan validasi migrasi Anda. Ini juga menjelaskan cara melakukan migrasi offline dan online.

Perbandingan dengan Amazon DocumentDB

	Server Couchbase	Amazon DocumentDB
Organisasi Data	Dalam versi 7.0 dan yang lebih baru, data diatur ke dalam ember, cakupan, dan koleksi. Dalam versi	Data diatur ke dalam database dan koleksi.

	Server Couchbase	Amazon DocumentDB
	sebelumnya, data diatur ke dalam ember.	
Kompatibilitas	Ada API terpisah untuk setiap layanan (misalnya data, indeks, pencarian, dll.). Pencarian sekunder menggunakan SQL++ (sebelumnya dikenal sebagai N1QL); bahasa query berdasarkan ANSI-standard SQL sehingga akrab bagi banyak pengembang.	Amazon DocumentDB kompatibel dengan MongoDB API.
Arsitektur	Penyimpanan dilampirkan ke setiap instance cluster. Anda tidak dapat menskalakan komputasi secara independen dari penyimpanan.	Amazon DocumentDB dirancang untuk cloud dan untuk menghindari keterbatasan arsitektur database tradisional. Lapisan komputasi dan penyimpanan dipisahkan di Amazon DocumentDB dan lapisan komputasi dapat diskalakan secara independen dari penyimpanan.
Tambahkan kapasitas baca sesuai permintaan	Cluster dapat ditingkatkan dengan menambahkan instance. Karena penyimpanan dilampirkan ke instance di mana layanan berjalan, waktu yang diperlukan untuk skala keluar tergantung pada jumlah data yang perlu dipindahkan ke instance baru, atau diseimbangkan kembali.	Anda dapat mencapai penskalaan baca untuk klaster Amazon DocumentDB Anda dengan membuat hingga 15 replika Amazon DocumentDB di cluster. Tidak ada dampak pada lapisan penyimpanan.

	Server Couchbase	Amazon DocumentDB
Pulihkan dengan cepat dari kegagalan node	Cluster memiliki kemampuan failover otomatis tetapi waktu untuk mengembalikan cluster ke kekuatan penuh tergantung pada jumlah data yang perlu dipindahk an ke instance baru.	Amazon DocumentDB dapat melakukan failover primer biasanya dalam 30 detik dan mengembalikan cluster kembali ke kekuatan penuh dalam 8-10 menit terlepas dari jumlah data dalam cluster.
Skalakan penyimpan an saat data tumbuh	Untuk penyimpanan cluster yang dikelola sendiri dan iOS tidak dapat diskalakan secara otomatis.	Penyimpanan Amazon DocumentDB dan skala iOS secara otomatis.
Backup data tanpa mempengaruhi kinerja	Cadangan dilakukan oleh layanan cadangan dan tidak diaktifkan secara default. Karena penyimpan an dan komputasi tidak terpisah, dapat berdampak pada kinerja.	Pencadangan Amazon DocumentDB diaktifkan secara default dan tidak dapat dimatikan . Cadangan ditangani oleh lapisan penyimpanan, sehingga berdampak nol pada lapisan komputasi. Amazon DocumentD B mendukung pemulihan dari snapshot cluster dan memulihkan ke titik waktu.
Daya tahan data	Bisa ada maksimal 3 salinan replika data dalam sebuah cluster dengan total 4 salinan. Setiap contoh di mana layanan data berjalan akan memiliki salinan aktif dan 1, 2, atau 3 replika data.	Amazon DocumentDB menyimpan 6 salinan data tidak peduli berapa banyak instance komputasi yang ada dengan kuorum tulis 4 dan persistst true. Klien menerima pengakuan setelah lapisan penyimpanan bertahan 4 salinan data.

	Server Couchbase	Amazon DocumentDB
Konsistensi	Konsistensi langsung untuk K/V operasi didukung. Couchbase SDK merutekan K/V permintaan ke instance tertentu yang berisi salinan aktif data sehingga setelah pembaruan diakui, klien dijamin untuk membaca pembaruan tersebut. Replikasi pembaruan ke layanan lain (indeks, pencarian, analisis, eventing) pada akhirnya konsisten.	Replika Amazon DocumentDB pada akhirnya konsisten. Jika pembacaan konsistensi langsung diperlukan, klien dapat membaca dari contoh utama.
Replikasi	Cross-Data Center Replication (XDCR) menyediakan replikasi data aktif yang disaring dan aktif dalam banyak passive/active topologi.	Cluster global Amazon DocumentDB menyediakan replikasi aktif-pasif dalam topologi 1:many (hingga 10).

Penemuan

Migrasi ke Amazon DocumentDB memerlukan pemahaman menyeluruh tentang beban kerja database yang ada. Penemuan beban kerja adalah proses menganalisis konfigurasi klaster Couchbase dan karakteristik operasional Anda — kumpulan data, indeks, dan beban kerja — untuk membantu memastikan transisi yang mulus dengan gangguan minimal.

Konfigurasi klaster

Couchbase menggunakan arsitektur service-centric di mana setiap kemampuan sesuai dengan layanan. Jalankan perintah berikut terhadap cluster Couchbase Anda untuk menentukan layanan mana yang digunakan (lihat Mendapatkan Informasi tentang Node):

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

Contoh output:

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

Layanan Couchbase meliputi:

Layanan data (kv)

Layanan data menyediakan read/write akses ke data dalam memori dan pada disk.

Amazon DocumentDB K/V mendukung operasi pada data JSON melalui MongoDB API.

Layanan kueri (n1ql)

Layanan query mendukung query data JSON melalui SQL++.

Amazon DocumentDB mendukung kueri data JSON melalui MongoDB API.

Layanan indeks (indeks)

Layanan indeks membuat dan memelihara indeks pada data, memungkinkan kueri lebih cepat.

Amazon DocumentDB mendukung indeks primer default dan pembuatan indeks sekunder pada data JSON melalui MongoDB API.

Layanan pencarian (fts)

Layanan pencarian mendukung pembuatan indeks untuk pencarian teks lengkap.

Fitur pencarian teks lengkap asli Amazon DocumentDB memungkinkan Anda melakukan pencarian teks pada kumpulan data tekstual besar menggunakan indeks teks tujuan khusus melalui MongoDB API. Untuk kasus penggunaan penelusuran lanjutan, integrasi Amazon DocumentDB Zero-ETL dengan OpenSearch Amazon Service menyediakan kemampuan pencarian lanjutan, seperti pencarian fuzzy, pencarian lintas koleksi, dan pencarian multibahasa, pada data Amazon DocumentDB.

Layanan Analytics (cbas)

Layanan analitik mendukung analisis data JSON dalam waktu dekat.

Amazon DocumentDB mendukung kueri ad-hoc pada data JSON melalui MongoDB API. Anda juga dapat menjalankan kueri kompleks pada data JSON Anda di Amazon DocumentDB menggunakan Apache Spark yang berjalan di Amazon EMR.

Layanan eventing (eventing)

Layanan eventing mengeksekusi logika bisnis yang ditentukan pengguna dalam menanggapi perubahan data.

Amazon DocumentDB mengotomatiskan beban kerja berbasis peristiwa dengan menjalankan fungsi setiap kali data berubah dengan cluster AWS Lambda Amazon DocumentDB Anda.

Layanan Backup (backup)

Layanan cadangan menjadwalkan pencadangan data penuh dan tambahan dan penggabungan cadangan data sebelumnya.

Amazon DocumentDB terus mencadangkan data Anda ke Amazon S3 dengan periode retensi 1—35 hari sehingga Anda dapat dengan cepat memulihkan ke titik mana pun dalam periode retensi cadangan. Amazon DocumentDB juga mengambil snapshot otomatis data Anda sebagai bagian dari proses backup terus menerus ini. Anda juga dapat mengelola pencadangan dan pemulihan Amazon AWS Backup DocumentDB dengan. .

Karakteristik operasional

Gunakan Discovery Tool for Couchbase untuk mendapatkan informasi berikut tentang kumpulan data, indeks, dan beban kerja Anda. Informasi ini akan membantu Anda mengukur cluster Amazon DocumentDB Anda.

Kumpulan data

Alat ini mengambil bucket, cakupan, dan informasi pengumpulan berikut:

1. nama bucket
2. jenis ember
3. nama lingkup

4. nama koleksi
5. ukuran total (byte)
6. total item
7. ukuran item (byte)

Indeks

Alat ini mengambil statistik indeks berikut dan semua definisi indeks untuk semua bucket. Perhatikan bahwa indeks primer dikecualikan karena Amazon DocumentDB secara otomatis membuat indeks utama untuk setiap koleksi.

1. nama bucket
2. nama lingkup
3. nama koleksi
4. nama indeks
5. ukuran indeks (byte)

Beban kerja

Alat ini mengambil K/V dan metrik kueri N1QL. K/V nilai metrik dikumpulkan pada tingkat bucket dan metrik SQL++ dikumpulkan di tingkat cluster.

Opsi baris perintah alat adalah sebagai berikut:

```
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>
```

Berikut adalah contoh perintah:

```
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 nilai metrik akan didasarkan pada sampel setiap 10 menit selama seminggu terakhir (lihat metode HTTP dan URI). Nilai metrik SQL++ akan didasarkan pada sampel setiap 1 detik selama 60 detik terakhir (lihat Label Umum). Output dari perintah akan berada di file-file berikut:

collection-stats.csv - bucket, ruang lingkup, dan informasi pengumpulan

```
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 - nama indeks dan ukuran

```
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 - dapatkan, atur, dan hapus metrik untuk semua bucket

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

n1ql-stats.csv - SQL++ pilih, hapus, dan masukkan metrik untuk cluster

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

indeks- .txt <bucket-name>— definisi indeks dari semua indeks dalam ember. Perhatikan bahwa indeks primer dikecualikan karena Amazon DocumentDB secara otomatis membuat indeks utama untuk setiap koleksi.

```
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`)
...
```

Perencanaan

Pada tahap perencanaan, Anda akan menentukan persyaratan klaster Amazon DocumentDB dan pemetaan bucket, cakupan, dan koleksi Couchbase ke database dan koleksi Amazon DocumentDB.

Persyaratan klaster Amazon DocumentDB

Gunakan data yang dikumpulkan dalam fase penemuan untuk mengukur cluster Amazon DocumentDB Anda. Lihat Ukuran instans untuk informasi selengkapnya tentang ukuran cluster Amazon DocumentDB Anda.

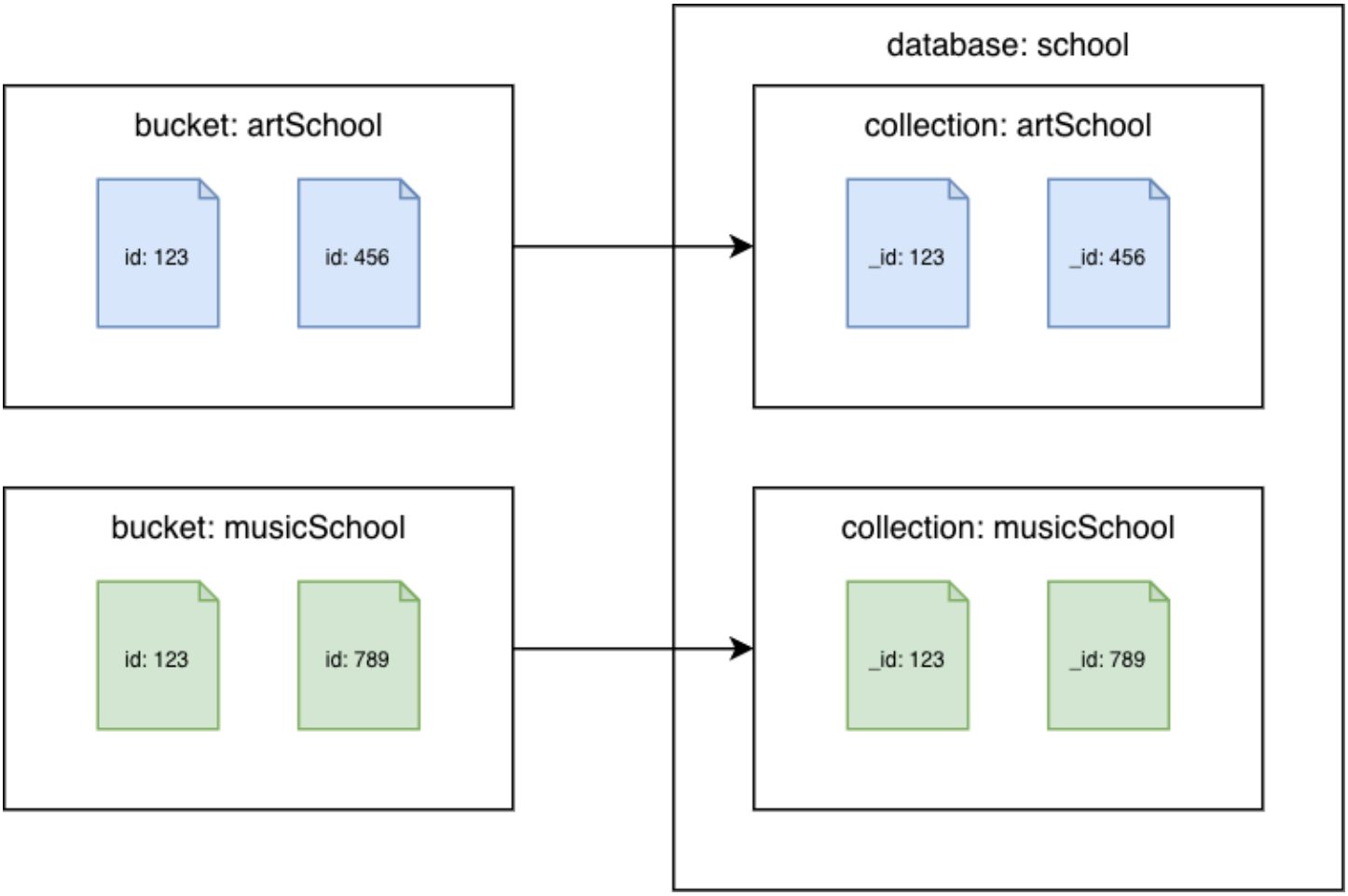
Memetakan ember, cakupan, dan koleksi ke database dan koleksi

Tentukan database dan koleksi yang akan ada di cluster Amazon DocumentDB Anda. Pertimbangkan opsi berikut tergantung pada bagaimana data diatur di cluster Couchbase Anda. Ini bukan satu-satunya pilihan, tetapi mereka memberikan titik awal untuk Anda pertimbangkan.

Couchbase Server 6.x atau sebelumnya

Bucket Couchbase ke koleksi Amazon DocumentDB

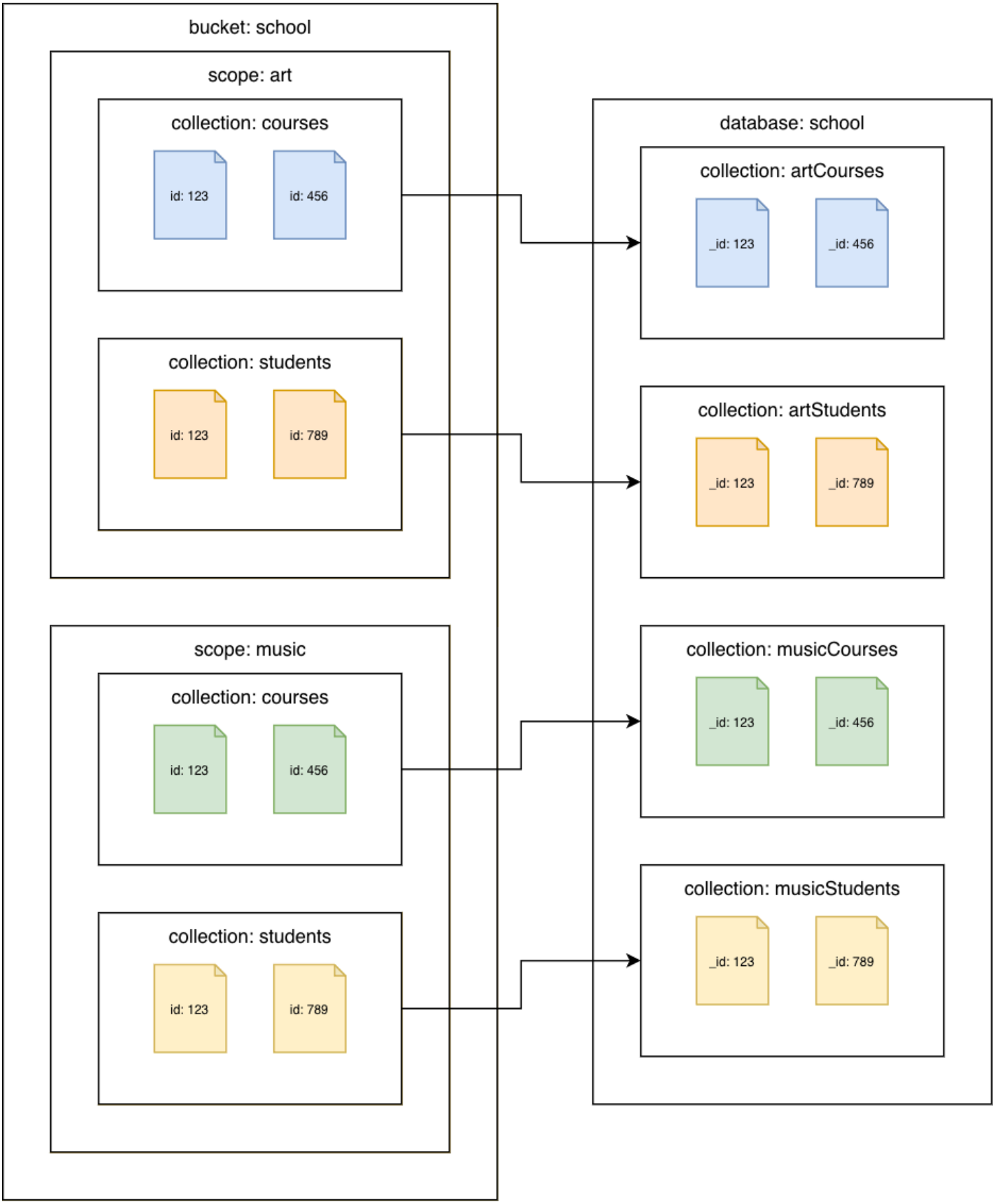
Migrasikan setiap bucket ke koleksi Amazon DocumentDB yang berbeda. Dalam skenario ini, `id` nilai dokumen Couchbase akan digunakan sebagai nilai Amazon DocumentDB. `_id`



Couchbase Server 7.0 atau yang lebih baru

Koleksi Couchbase ke koleksi Amazon DocumentDB

Migrasikan setiap koleksi ke koleksi Amazon DocumentDB yang berbeda. Dalam skenario ini, `id` nilai dokumen Couchbase akan digunakan sebagai nilai Amazon DocumentDB. `_id`


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

Migrasi

Migrasi indeks

Migrasi ke Amazon DocumentDB melibatkan transfer tidak hanya data tetapi juga indeks untuk mempertahankan kinerja kueri dan mengoptimalkan operasi database. Bagian ini menguraikan proses langkah demi langkah terperinci untuk memigrasikan indeks ke Amazon DocumentDB sambil memastikan kompatibilitas dan efisiensi.

Gunakan Amazon Q untuk mengonversi `CREATE INDEX` pernyataan SQL++ ke perintah Amazon DocumentDB. `createIndex()`

1. Unggah <bucket name>file indeks- .txt yang dibuat oleh Discovery Tool untuk Couchbase.
2. Masukkan prompt berikut:

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

Amazon Q akan menghasilkan perintah Amazon `createIndex()` DocumentDB yang setara. Perhatikan bahwa Anda mungkin perlu memperbarui nama koleksi berdasarkan cara Anda memetakan bucket, cakupan, dan koleksi Couchbase ke koleksi Amazon DocumentDB.

Contoh:

indexes-beer-sample.txt

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

Contoh keluaran Amazon Q (kutipan):

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

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

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

Untuk indeks apa pun yang tidak dapat dikonversi oleh Amazon Q, lihat Mengelola indeks Amazon DocumentDB dan Indeks serta properti indeks untuk informasi lebih lanjut.

Kode refactor untuk menggunakan API MongoDB

Klien menggunakan Couchbase SDK untuk terhubung ke Couchbase Server. Klien Amazon DocumentDB menggunakan driver MongoDB untuk terhubung ke Amazon DocumentDB. Semua bahasa yang didukung oleh Couchbase SDK juga didukung oleh driver MongoDB. Lihat Driver MongoDB untuk informasi lebih lanjut tentang driver untuk bahasa Anda.

Karena API berbeda antara Couchbase Server dan Amazon DocumentDB, Anda perlu memfaktorkan ulang kode Anda untuk menggunakan API MongoDB yang sesuai. Anda dapat menggunakan Amazon Q untuk mengonversi panggilan K/V API dan kueri SQL++ ke API MongoDB yang setara:

1. Unggah file kode sumber.
2. Masukkan prompt berikut:

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

Menggunakan contoh kode Hello Couchbase Python, Amazon Q menghasilkan berikut (kutipan):

```
from datetime import timedelta
from pymongo import MongoClient

# Connection parameters
database_name = "travel-sample"

# Connect to Amazon DocumentDB cluster
client = MongoClient('<Amazon DocumentDB connection string>')

# 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")
```

Lihat Menghubungkan secara terprogram ke Amazon DocumentDB untuk contoh menghubungkan ke Amazon DocumentDB dengan Python, PHP, Go, Java, C#/.NET, R Node.js, dan Ruby.

Pilih pendekatan migrasi

Saat memigrasikan data ke Amazon DocumentDB, ada dua opsi:

1. migrasi offline
2. migrasi online

Migrasi offline

Pertimbangkan migrasi offline saat:

- Waktu henti dapat diterima: Migrasi offline melibatkan penghentian operasi penulisan ke database sumber, mengekspor data, dan kemudian mengimpornya ke Amazon DocumentDB. Proses ini menimbulkan downtime untuk aplikasi Anda. Jika aplikasi atau beban kerja Anda dapat mentolerir periode tidak tersedianya ini, migrasi offline adalah opsi yang layak.
- Migrasi kumpulan data yang lebih kecil atau melakukan bukti konsep: Untuk kumpulan data yang lebih kecil, waktu yang diperlukan untuk proses ekspor dan impor relatif singkat, membuat migrasi offline menjadi metode yang cepat dan sederhana. Ini juga cocok untuk pengembangan, pengujian, dan lingkungan pembuktian konsep di mana waktu henti kurang kritis.
- Kesederhanaan adalah prioritas: Metode offline, menggunakan cbexport dan mongoimport, umumnya merupakan pendekatan yang paling mudah untuk memigrasikan data. Ini menghindari kompleksitas pengambilan data perubahan (CDC) yang terlibat dalam metode migrasi online.
- Tidak ada perubahan berkelanjutan yang perlu direplikasi: Jika database sumber tidak secara aktif menerima perubahan selama migrasi, atau jika perubahan tersebut tidak penting untuk ditangkap dan diterapkan ke target selama proses migrasi, maka pendekatan offline sesuai.

Topik

- Couchbase Server 6.x atau sebelumnya
- Couchbase Server 7.0 atau yang lebih baru

Couchbase Server 6.x atau sebelumnya

Bucket Couchbase ke koleksi Amazon DocumentDB

Ekspor data menggunakan cbexport json untuk membuat dump JSON dari semua data di bucket. Untuk `--format` opsi yang dapat Anda gunakan `lines` atau`list`.

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

Impor data ke koleksi Amazon DocumentDB menggunakan mongoimport dengan opsi yang sesuai untuk mengimpor baris atau daftar:

baris:

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

daftar:

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

Couchbase Server 7.0 atau yang lebih baru

Untuk melakukan migrasi offline, gunakan alat cbexport dan mongoimport:

Bucket Couchbase dengan cakupan default dan koleksi default

Ekspor data menggunakan cbexport json untuk membuat dump JSON dari semua koleksi di bucket. Untuk `--format` opsi yang dapat Anda gunakan `lines` atau`list`.

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

Impor data ke koleksi Amazon DocumentDB menggunakan mongoimport dengan opsi yang sesuai untuk mengimpor baris atau daftar:

baris:

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

daftar:

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

Koleksi Couchbase ke koleksi Amazon DocumentDB

Ekspor data menggunakan cbexport json untuk membuat dump JSON dari setiap koleksi. Gunakan `--include-data` opsi untuk mengekspor setiap koleksi. Untuk `--format` opsi yang dapat Anda

gunakan `lines` ataulist. Gunakan `--collection-field` opsi `--scope-field` dan untuk menyimpan nama ruang lingkup dan koleksi di bidang yang ditentukan di setiap dokumen 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"
```

Karena cbexport menambahkan `_collection` bidang `_scope` dan ke setiap dokumen yang diekspor, Anda dapat menghapusnya dari setiap dokumen dalam file ekspor melalui pencarian dan penggantian,`sed`, atau metode apa pun yang Anda inginkan.

Impor data untuk setiap koleksi ke koleksi Amazon DocumentDB menggunakan mongoimport dengan opsi yang sesuai untuk mengimpor baris atau daftar:

baris:

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

daftar:

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

Migrasi online

Pertimbangkan migrasi online saat Anda perlu meminimalkan waktu henti dan perubahan yang sedang berlangsung perlu direplikasi ke Amazon DocumentDB dalam waktu hampir nyata.

Lihat Cara melakukan migrasi langsung dari Couchbase ke Amazon DocumentDB untuk mempelajari cara melakukan migrasi langsung ke Amazon DocumentDB. Dokumentasi memandu Anda dalam penerapan solusi dan melakukan migrasi langsung ke bucket ke klaster Amazon DocumentDB.

Topik

- Couchbase Server 6.x atau sebelumnya
- Couchbase Server 7.0 atau yang lebih baru

Couchbase Server 6.x atau sebelumnya

Bucket Couchbase ke koleksi Amazon DocumentDB

Utilitas migrasi untuk Couchbase sudah dikonfigurasi sebelumnya untuk melakukan migrasi online bucket Couchbase ke koleksi Amazon DocumentDB. Melihat konfigurasi konektor sink, `document.id.strategy` parameter dikonfigurasi untuk menggunakan nilai kunci pesan sebagai nilai `_id` bidang (lihat Properti Strategi Id Konektor Sink):

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

Couchbase Server 7.0 atau yang lebih baru

Bucket Couchbase dengan cakupan default dan koleksi default

Utilitas migrasi untuk Couchbase sudah dikonfigurasi sebelumnya untuk melakukan migrasi online bucket Couchbase ke koleksi Amazon DocumentDB. Melihat konfigurasi konektor sink, `document.id.strategy` parameter dikonfigurasi untuk menggunakan nilai kunci pesan sebagai nilai `_id` bidang (lihat Properti Strategi Id Konektor Sink):

```
ConnectorConfiguration:
```

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

Koleksi Couchbase ke koleksi Amazon DocumentDB

Konfigurasikan konektor sumber untuk mengalirkan setiap koleksi Couchbase di setiap lingkup ke topik terpisah (lihat Opsi Konfigurasi Sumber). Contoh:

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

Konfigurasikan konektor sink untuk mengalirkan dari setiap topik ke koleksi Amazon DocumentDB terpisah (lihat Properti Konfigurasi Konektor Sink). Contoh:

```
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>'
```

Validasi

Bagian ini menyediakan proses validasi terperinci untuk memverifikasi konsistensi dan integritas data setelah bermigrasi ke Amazon DocumentDB. Langkah-langkah validasi berlaku terlepas dari metode migrasi.

Topik

- Verifikasi bahwa semua koleksi ada di target
- Verifikasi jumlah dokumen antara souce dan cluster target

- Bandingkan dokumen antara sumber dan kelompok target

Verifikasi bahwa semua koleksi ada di target

Sumber Couchbase

opsi 1: meja kerja kueri

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

opsi 2: alat cbq

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

Target Amazon DocumentDB

mongosh (lihat Connect ke cluster Amazon DocumentDB Anda):

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

Verifikasi jumlah dokumen antara souce dan cluster target

Sumber Couchbase

Couchbase Server 6.x atau sebelumnya

opsi 1: meja kerja kueri

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

opsi 2: cbq

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

Couchbase Server 7.0 atau yang lebih baru

opsi 1: meja kerja kueri

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

opsi 2: cbq

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

Target Amazon DocumentDB

mongosh (lihat Connect ke cluster Amazon DocumentDB Anda):

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

Bandingkan dokumen antara sumber dan kelompok target

Sumber Couchbase

Couchbase Server 6.x atau sebelumnya

opsi 1: meja kerja kueri

```
SELECT META().id as _id, *
```

```
FROM `<bucket>`
LIMIT 5
```

opsi 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 atau yang lebih baru

opsi 1: meja kerja kueri

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

opsi 2: cbq

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

Target Amazon DocumentDB

mongosh (lihat Connect ke cluster Amazon DocumentDB Anda):

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

Meningkatkan Amazon DocumentDB

Topik

- Tanggal dukungan versi mesin Amazon DocumentDB
- Peningkatan versi utama Amazon DocumentDB di tempat
- Peningkatan versi minor Amazon DocumentDB
- Memutakhirkan cluster Amazon DocumentDB Anda menggunakan AWS Database Migration Service
- Menggunakan rilis dukungan jangka panjang (LTS)

Tanggal dukungan versi mesin Amazon DocumentDB

Anda dapat menggunakan tanggal berikut untuk merencanakan siklus pengujian dan peningkatan Anda.

Kalender rilis untuk Amazon DocumentDB versi utama

Amazon DocumentDB saat ini mendukung versi utama berikut. Untuk meningkatkan versi mesin Anda, lihat Memutakhirkan versi mesin utama atauUpgrade menggunakan AWS DMS.

Versi mesin	Tanggal rilis	Akhir dari dukungan standar	Mulai dari Extended Support (harga tahun 1)	Mulai dari Extended Support (harga tahun 3)	Akhir dari Extended Support
Versi 3.6	9 Januari 2019	30 Maret 2026	31 Maret 2026	31 Maret 2028	30 Maret 2029
Versi 4.0	9 November 2020	N/A	N/A	N/A	N/A
Versi 5.0	1 Maret 2023	N/A	N/A	N/A	N/A
Versi 8.0	14 November 2025	N/A	N/A	N/A	N/A

Amazon DocumentDB mendukung beberapa versi engine di bawah dukungan standar. Anda dapat terus menjalankan versi melewati akhir tanggal dukungan standar dengan biaya Extended Support. Untuk informasi selengkapnya, lihat Amazon DocumentDB Extended Support dan harga Amazon DocumentDB.

Kalender rilis untuk Amazon DocumentDB versi minor

Amazon DocumentDB saat ini mendukung versi minor berikut. Jadwal rilis akan bervariasi tergantung pada fitur atau perbaikan tambahan. Versi minor dapat mencapai akhir dukungan standar sebelum versi utama yang sesuai melakukannya. Untuk memutakhirkan versi mesin Anda, lihatMemutakhirkan versi mesin minor.

> **Note**
>
> Versi minor tersedia mulai dari versi utama 5.0.

Versi mesin	Tanggal rilis
Versi 5.0.0 (LTS)	16 Februari 2026
Versi 5.0.1	8. Juni 2026

Amazon DocumentDB menetapkan versi tertentu sebagai rilis Long-Term Support (LTS). Lihat informasi yang lebih lengkap di Menggunakan rilis dukungan jangka panjang (LTS).

Peningkatan versi utama Amazon DocumentDB di tempat

Anda dapat melakukan upgrade versi utama (MVU) in-place dari cluster Amazon DocumentDB Anda sambil mempertahankan titik akhir, penyimpanan, dan tag yang sama. Aplikasi Anda terus bekerja tanpa modifikasi. Fitur ini tersedia tanpa biaya tambahan di semua wilayah di mana Amazon DocumentDB tersedia.

> **Important**
>
> Cluster Anda tidak akan tersedia selama peningkatan dan akan mengalami beberapa reboot. Jangan terhubung ke, membaca dari, atau menulis ke cluster setelah memulai upgrade.

Downtime bervariasi tergantung pada jumlah koleksi, indeks, database, dan instance. Kami merekomendasikan untuk melakukan peningkatan selama jendela pemeliharaan Anda atau jam pemanfaatan rendah.
Setelah ditingkatkan, Anda tidak dapat menurunkan versi ke versi sebelumnya. Anda dapat mengembalikan snapshot pra-upgrade ke cluster baru jika diperlukan.

Topik

- Jalur pemutakhiran yang didukung
- Prasyarat
- Praktik terbaik
- Apa yang berubah setelah memutakhirkan dari Amazon DocumentDB 5.0 ke 8.0
- Post-upgrade pertimbangan untuk cluster ditingkatkan dari 3.6 atau 4.0
- Melakukan upgrade
- Pemecahan masalah

Jalur pemutakhiran yang didukung

Versi sumber	Versi target
Amazon DocumentDB 3.6	Amazon DocumentDB 5.0
Amazon DocumentDB 4.0	Amazon DocumentDB 5.0
Amazon DocumentDB 5.0	Amazon DocumentDB 8.0

Note

In-place MVU tidak didukung untuk cluster global atau cluster elastis. Untuk memutakhirkan cluster global, hapus cluster sekunder, konversi primer ke cluster regional, lakukan MVU, lalu buat ulang cluster global dengan menambahkan cluster sekunder menggunakan nama yang sama untuk mempertahankan titik akhir Anda. Anda akan dikenakan I/O biaya saat primer yang ditingkatkan mereplikasi data ke sekunder baru. Untuk langkah mendetail, lihat Menghapus klaster dari klaster global Amazon DocumentDB.

Prasyarat

- Jenis instans - Amazon DocumentDB 4.0+ tidak mendukung instans db.r4. Ubah `db.r4.*` instance apa pun ke `db.r5.*` instance atau yang lebih baru sebelum memutakhirkan. Lihat Memodifikasi instance Amazon DocumentDB dan Kelas instance yang didukung berdasarkan wilayah.
- Patch OS - Terapkan tindakan pemeliharaan OS yang tertunda pada semua instance sebelum memutakhirkan. Lihat Pembaruan sistem operasi Amazon DocumentDB.

> **Note**
>
> Patch engine tingkat cluster yang tertunda dapat menyembunyikan patch OS instance. Terapkan tambalan mesin terlebih dahulu jika diperlukan. Lihat Melakukan pembaruan tambalan ke versi mesin cluster.

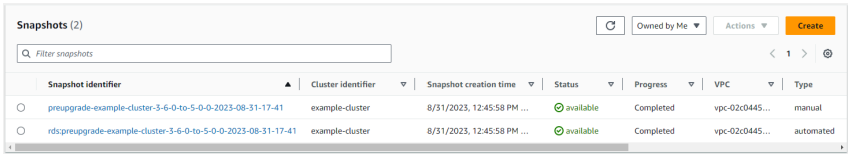
- Batas indeks pada instans burstable (instans t-family) — Jika Anda memiliki lebih dari 3.000 indeks pada instans burstable, tingkatkan primer menjadi setidaknya db.r5.large sebelum memutakhirkan. Anda dapat menurunkan skala kembali setelah pemutakhiran selesai.

Instans	Indeks maksimum untuk MVU
db.t4g.medium	3K
db.t3.medium	10K

- Grup parameter - Sebaiknya siapkan grup parameter cluster khusus untuk versi target sebelum memutakhirkan. Jika tidak ditentukan, grup parameter default untuk versi target akan digunakan (misalnya, `default.docdb5.0` atau`default.docdb8.0`).
- Cuplikan manual - Buat snapshot manual sebelum memutakhirkan. Proses pemutakhiran membuat snapshot otomatis bernama`preupgrade-<name>-<version>-<timestamp>`, tetapi kami sangat menyarankan untuk memiliki cadangan Anda sendiri. Lihat Membuat snapshot cluster manual.

> **Note**
>
> Snapshot otomatis yang dibuat oleh proses pemutakhiran tidak akan dihapus secara otomatis setelah pemutakhiran versi utama di tempat selesai. Cuplikan ini tidak akan

dikenakan biaya apa pun selama berada dalam periode retensi. Anda dapat memilih untuk menghapus snapshot ini setelah Anda memverifikasi peningkatan klaster yang berhasil.



Praktik terbaik

Sebelum upgrade - pengujian dengan klon

1. Gunakan Mengkloning volume untuk cluster Amazon DocumentDB untuk membuat tiruan dari cluster Anda. Anda tidak akan dikenakan biaya penyimpanan kecuali Anda memodifikasi data pada klon.
2. Cocokkan jumlah instance klon dengan cluster target untuk perkiraan waktu yang realistis.
3. Lakukan MVU pada klon dan uji sepenuhnya perbedaan fungsional.
4. Periksa apakah pemutakhiran sudah dijadwalkan dengan menjalankan `describe-db-clusters`dan mencari`PendingModifiedValues.EngineVersion`. Jika Anda telah memodifikasi cluster dan memilih untuk menerapkannya di jendela pemeliharaan berikutnya, jadwal tidak akan terlihat di konsol tetapi Anda dapat melihatnya di AWS CLI:

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

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

5. Jika pengujian berhasil, lanjutkan dengan upgrade pada cluster produksi Anda.

Selama upgrade

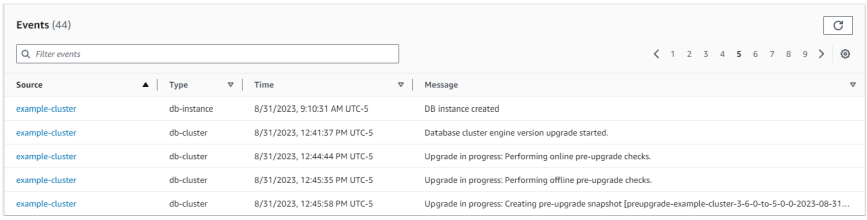
Anda dapat memantau kemajuan pemutakhiran versi utama di tempat dengan berlangganan acara pemeliharaan klaster. Saat pemutakhiran selesai, Anda akan menerima acara “Database cluster

major version has been upgraded". Ini dan peristiwa lain yang terjadi selama pemutakhiran muncul di bagian Peristiwa dan Tag pada halaman detail cluster di konsol Amazon DocumentDB. Status cluster kemudian berubah dari `upgrading` ke`available`.

Peristiwa berikut dihasilkan selama peningkatan:

1. Pemutakhiran versi utama mesin cluster basis data dimulai. Cluster tetap online.
2. Persiapan peningkatan sedang berlangsung: Memulai prapecek peningkatan online.
3. Persiapan peningkatan sedang berlangsung: Prakecek peningkatan online yang telah selesai.
4. Mengambil cluster database offline sementara instance utama menyelesaikan patch/upgrade proses.
5. Persiapan peningkatan sedang berlangsung: Memulai prapecek peningkatan offline.
6. Persiapan peningkatan sedang berlangsung: Prakecek peningkatan offline yang diselesaikan.
7. <version-to><timestamp>Upgrade sedang berlangsung: Membuat snapshot pra-upgrade [preupgrade- <cluster-name>- <version-from>-to- -].
8. Upgrade sedang berlangsung: Volume kloning.
9. Upgrade sedang berlangsung: Upgrade penulis.
10. Upgrade sedang berlangsung: Meningkatkan pembaca.
11. Versi mayor mesin klaster basis data telah ditingkatkan.

Peristiwa terlihat di konsol di bawah halaman Acara:



Dari AWS CLI, Anda dapat menjalankan `aws docdb describe-events`untuk memantau kemajuan peningkatan. Untuk menerima pemberitahuan secara otomatis, gunakan `aws docdb create-event-subscription`untuk berlangganan acara dan mengarahkannya ke Amazon SNS untuk email, pesan push, atau metode pengiriman lainnya. Untuk informasi selengkapnya, lihat Berlangganan acara Amazon DocumentDB.

```
aws docdb describe-events
  --source-identifier mydocdbcluster
```

```
  --source-type db-cluster
```

Perintah tersebut mengembalikan output serupa dengan berikut ini:

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

Setelah peningkatan

Warning

Segera setelah upgrade versi utama di tempat, klaster Amazon DocumentDB Anda mengisi kembali metadata indeks yang digunakan mesin database untuk mengoptimalkan rencana eksekusi kueri. Kinerja kueri kembali ke tingkat yang diharapkan setelah proses ini selesai. Biasanya selesai dalam beberapa menit tetapi dapat memakan waktu hingga dua jam tergantung pada jumlah indeks pada cluster Anda.
Jangan reboot, failover, atau skala instance penulis up/down Anda selama waktu ini, karena dapat mengganggu perhitungan ulang metadata indeks. Sebaiknya tunggu hingga Anda mengamati kinerja kueri yang diharapkan sebelum membuat perubahan tersebut.
Lacak kemajuan melalui peristiwa klaster berikut:

- Post-upgrade status cluster: Proses penyegaran metadata indeks dimulai
- Post-upgrade status cluster: Proses penyegaran metadata indeks selesai dalam X detik

Hubungi AWS dukungan jika proses penyegaran metadata indeks belum selesai dalam waktu tiga jam, atau jika Anda terus mengalami masalah kinerja setelah proses selesai.

1. Ambil snapshot manual dari cluster yang ditingkatkan jika Anda perlu mengembalikan ke status pasca-peningkatan. Proses snapshot otomatis akan dilanjutkan segera setelah upgrade versi utama di tempat selesai.
2. Cluster tag ditingkatkan dari 3.6. Tambahkan tag untuk membedakan kluster yang ditingkatkan dari 3.6. (lihatthe section called “Post-upgrade pertimbangan untuk cluster ditingkatkan dari 3.6 atau 4.0”).
3. Perbarui driver Anda. Untuk menggunakan fitur baru (misalnya, pemeriksaan, tampilan, atau kompresi Zstd di 8.0), tingkatkan ke versi API MongoDB yang sesuai. Untuk informasi selengkapnya, lihat Apa yang baru di Amazon DocumentDB 8.0.
4. Uji secara menyeluruh. Validasi aplikasi Anda terhadap klaster yang ditingkatkan.

Apa yang berubah setelah memutakhirkan dari Amazon DocumentDB 5.0 ke 8.0

Setelah melakukan upgrade versi utama dari Amazon DocumentDB 5.0 ke 8.0, fitur berikut diaktifkan atau diubah:

- Penyusunan. Amazon DocumentDB 8.0 mendukung pemeriksaan. Setelah upgrade, koleksi baru dan indeksnya, dan indeks baru pada koleksi yang ada, telah mengaktifkan pemeriksaan secara default.
- Indeks teks. Indeks teks baru dibuat menggunakan Text Index V2, yang menggunakan parser pencarian teks yang diperbarui untuk meningkatkan kompatibilitas MongoDB. Indeks teks yang ada tidak terpengaruh.
- Versi perencana kueri. Jika Anda tidak memiliki grup parameter kustom, grup parameter default baru dibuat untuk Amazon DocumentDB 8.0 dengan Planner Versi 3 dipilih secara otomatis. Dengan perencana kueri versi 3, tampilan juga tersedia.
- Kompresi. Amazon DocumentDB 8.0 mendukung kompresi dokumen berbasis kamus menggunakan algoritma Zstd. Setelah upgrade, koleksi baru dibuat dengan kompresi Zstd diaktifkan secara default. Koleksi yang ada dari 5.0 mempertahankan pengaturan kompresi mereka. Untuk memanfaatkan kompresi Zstd pada koleksi yang ada, Anda dapat memodifikasi pengaturan kompresi mereka. Untuk informasi lebih lanjut, lihat Dictionary-based kompresi.
- Indeks membangun kembali. Jika Anda memutakhirkan dari Amazon DocumentDB 5.0 ke Amazon DocumentDB 8.0, tidak diperlukan rebuild indeks.

> **Important**
>
> Amazon DocumentDB 8.0 membutuhkan TLS 1.2 atau lebih tinggi. TLS 1.0 dan TLS 1.1 tidak lagi didukung.

> **Note**
>
> Untuk daftar lengkap perbedaan fungsional, lihatKompatibilitas Amazon DocumentDB dengan MongoDB.

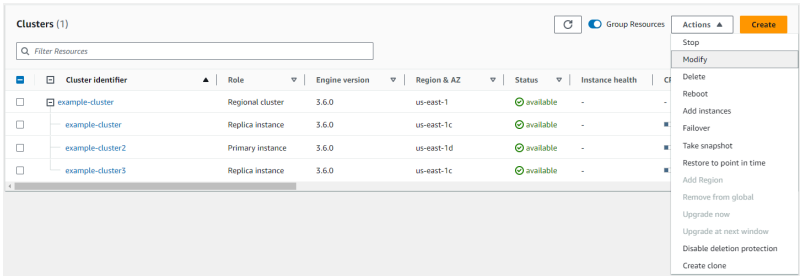
Post-upgrade pertimbangan untuk cluster ditingkatkan dari 3.6 atau 4.0

- Indeks membangun kembali. MVU mempertahankan indeks asli. Amazon DocumentDB 5.0 telah meningkatkan pemeliharaan indeks dan pengumpulan sampah, terutama untuk indeks kardinalitas rendah. Setelah memutakhirkan dari 3.6 atau 4.0, kami sarankan untuk membangun kembali indeks Anda untuk memastikan kinerja kueri yang optimal (opsional, melibatkan tambahan). I/O Lihat Pemeliharaan indeks menggunakan `reIndex`.
- Perbandingan numerik subdokumen (hanya 3.6). Cluster yang ditingkatkan dari 3.6 mewarisi perilaku 3.6 di mana tipe numerik dalam subdokumen tidak dibandingkan antar tipe. Misalnya, `{a: {b: NumberLong(1)}}` tidak sama dengan `{a: {b: 1}}` 3.6, tetapi mereka sama di 4.0 dan yang lebih baru. Perilaku ini memengaruhi klaster apa pun yang ditingkatkan dari 3.6.

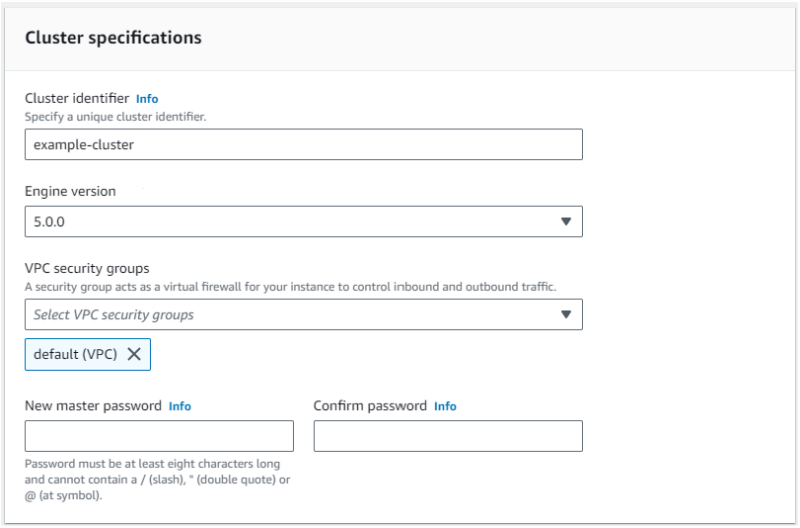
Melakukan upgrade

Using the Konsol Manajemen AWS

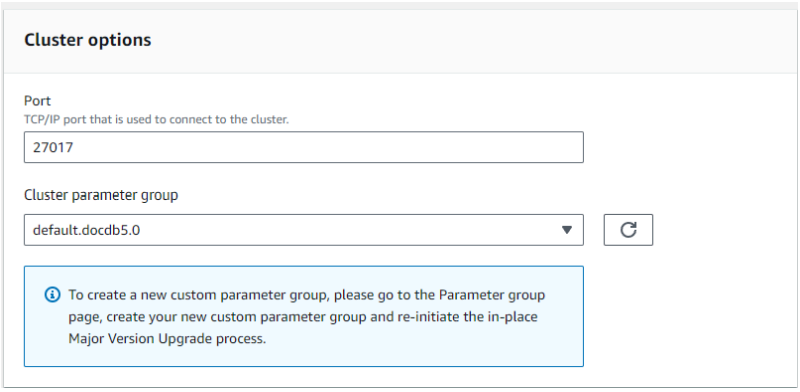
1. Masuk ke Konsol Manajemen AWSdan buka konsol Amazon DocumentDB.
2. Dalam tabel Clusters, pilih cluster sumber, pilih Actions, dan kemudian Modify.



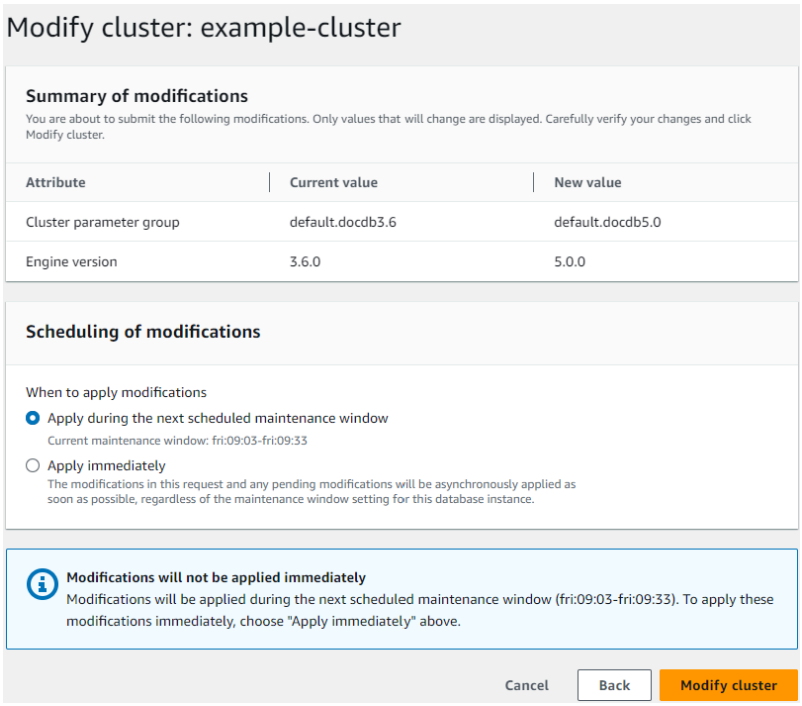
3. Dalam spesifikasi Cluster, pilih versi target (misalnya, 5.0.0 atau 8.0.0) dari dropdown versi Engine.



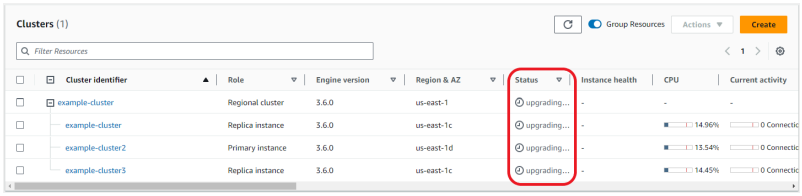
4. Dalam opsi Cluster, pilih grup parameter cluster Anda untuk versi mesin target. Anda dapat menggunakan default (misalnya, default.docdb5.0 atau default.docdb8.0) atau grup parameter khusus yang Anda buat.



5. Pilih Lanjutkan, pilih preferensi penjadwalan Anda (segera terapkan atau jendela pemeliharaan berikutnya), lalu pilih Ubah cluster.



6. Pantau status cluster di tabel cluster saat berubah menjadi upgrade:



Using the AWS CLI

Gunakan [modify-db-cluster](#)dengan `--allow-major-version-upgrade` bendera:

```
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
```

Ganti masing-masing *placeholder* dengan informasi cluster Anda.

Pemecahan masalah

- Upgrade kegagalan dan rollback. Jika upgrade gagal, secara otomatis mencoba rollback. Rollback yang berhasil menghasilkan peristiwa: “Kluster basis data berada dalam keadaan yang tidak dapat ditingkatkan.” Cluster Anda kembali ke status pra-pemutakhiran dan Anda dapat terus menggunakannya. Hubungi AWS dukungan untuk memecahkan masalah sebelum mencoba kembali.
- Post-upgrade kinerja. Degradasi kinerja sementara dan pemanfaatan CPU yang tinggi dapat terjadi saat penyegaran metadata indeks berjalan. Jika degradasi berlanjut lebih dari 3 jam, hubungi AWS dukungan.

Untuk bantuan tambahan, hubungi AWS Support.

Peningkatan versi minor Amazon DocumentDB

> **Note**
>
> Topik ini mencakup peningkatan versi minor dalam versi utama yang sama (misalnya, meningkatkan dari 5.0.0 ke 5.0.1). Untuk informasi tentang memutakhirkan ke versi utama yang berbeda, lihatPeningkatan versi utama Amazon DocumentDB di tempat.

Anda dapat memutakhirkan versi mesin minor klaster Amazon DocumentDB Anda (misalnya, dari 5.0.0 ke 5.0.1) dengan memodifikasi cluster. Upgrade versi minor dapat diterapkan segera atau selama jendela pemeliharaan berikutnya.

Sebelum melakukan upgrade versi minor

Sebelum melakukan upgrade versi minor, tindakan berikut mungkin berguna:

- Lakukan upgrade selama periode lalu lintas rendah untuk mengurangi dampak downtime singkat.
- Verifikasi bahwa versi minor target adalah target pemutakhiran yang valid untuk versi Anda saat ini dengan menggunakan `describe-db-engine-versions` perintah.
- Untuk cluster global, cluster sekunder harus ditingkatkan sebelum cluster primer. Untuk informasi selengkapnya, lihat Penambalan cluster global.

> **Important**
>
> - Upgrade adalah operasi satu arah - Anda tidak dapat menurunkan versi setelah upgrade.
> - Cluster Anda akan mengalami downtime singkat selama proses upgrade.

Melakukan upgrade versi minor

Using the Konsol Manajemen AWS

Untuk melakukan upgrade versi minor menggunakan Konsol Manajemen AWS:

1. Masuk ke Konsol Manajemen AWSdan buka konsol Amazon DocumentDB.
2. Dalam tabel Clusters, pilih cluster yang ingin Anda upgrade, pilih Actions, dan kemudian pilih Modify.
3. Pada dialog Modify cluster, di bagian Spesifikasi cluster, pilih versi minor target dari menu drop-down versi Engine.
4. Gulir ke bawah dan pilih Lanjutkan.
5. Di bagian Penjadwalan modifikasi, pilih untuk segera menerapkan perubahan atau selama jendela pemeliharaan berikutnya.

   Lalu pilih Ubah klaster.

Using the AWS CLI

Gunakan `modify-db-cluster`perintah dengan `--engine-version` parameter:

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

Pada contoh di atas, ganti masing-masing *`user input placeholder`* dengan informasi cluster Anda.

> **Note**
>
> Untuk menerapkan pemutakhiran selama jendela pemeliharaan berikutnya, bukan segera, gunakan `--no-apply-immediately` sebagai pengganti`--apply-immediately`.

Memverifikasi peningkatan versi minor

Setelah upgrade selesai, verifikasi versi mesin menggunakan: AWS CLI

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

Keluaran dari operasi ini terlihat seperti berikut ini:

```
"5.0.1"
```

Memutakhirkan cluster Amazon DocumentDB Anda menggunakan AWS Database Migration Service

> **Important**
>
> Amazon DocumentDB tidak mengikuti siklus hidup dukungan yang sama seperti MongoDB dan jadwal akhir masa pakai MongoDB tidak berlaku untuk Amazon DocumentDB.

Anda dapat memutakhirkan cluster Amazon DocumentDB Anda ke versi yang lebih tinggi dengan penggunaan downtime minimal. AWS DMS AWS DMS adalah layanan yang dikelola sepenuhnya yang memudahkan untuk bermigrasi dari versi Amazon DocumentDB yang lebih lama, database relasional, dan basis data non-relasional ke cluster Amazon DocumentDB target Anda.

Topik

- Langkah 1: Aktifkan aliran perubahan
- Langkah 2: Ubah durasi retensi aliran perubahan
- Langkah 3: Migrasikan indeks Anda

- Langkah 4: Buat AWS DMS Tipe instans replikasi
- Langkah 5: Buat AWS DMS titik akhir sumber
- Langkah 6: Buat AWS DMS titik akhir target
- Langkah 7: Buat dan jalankan tugas migrasi
- Langkah 8: Mengubah endpoint aplikasi ke cluster Amazon DocumentDB target

Langkah 1: Aktifkan aliran perubahan

Untuk melakukan migrasi downtime minimal, AWS DMS memerlukan akses ke aliran perubahan kluster. Streaming perubahan Amazon DocumentDB menyediakan urutan waktu kejadian pembaruan yang terjadi dalam koleksi dan basis data klaster Anda. Membaca dari aliran perubahan memungkinkan AWS DMS untuk melakukan pengambilan data perubahan (CDC) dan menerapkan pembaruan tambahan ke cluster Amazon DocumentDB target.

Untuk mengaktifkan streaming perubahan bagi semua koleksi pada basis data tertentu, lakukan autentikasi ke klaster Amazon DocumentDB Anda dengan menggunakan mongo shell dan jalankan perintah berikut:

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

Langkah 2: Ubah durasi retensi aliran perubahan

Selanjutnya, ubah periode retensi streaming perubahan berdasarkan berapa lama Anda ingin mempertahankan peristiwa perubahan di streaming perubahan. Misalnya, jika Anda mengharapkan migrasi AWS DMS klaster Amazon DocumentDB memakan waktu 12 jam, Anda harus menyetel retensi aliran perubahan ke nilai yang lebih besar dari 12 jam. Periode retensi default untuk klaster Amazon DocumentDB Anda adalah tiga jam. Anda dapat mengubah durasi retensi log aliran perubahan untuk klaster Amazon DocumentDB Anda menjadi antara satu jam dan tujuh hari menggunakan Konsol Manajemen AWS atau. AWS CLI Untuk detail selengkapnya, lihat Memodifikasi Durasi Retensi Log Streaming Perubahan.

Langkah 3: Migrasikan indeks Anda

Buat indeks yang sama pada cluster Amazon DocumentDB target yang Anda miliki di cluster Amazon DocumentDB sumber Anda. Meskipun AWS DMS menangani migrasi data, itu tidak memigrasikan indeks. Untuk memigrasikan indeks, gunakan Alat Indeks Amazon DocumentDB untuk mengekspor indeks dari cluster Amazon DocumentDB sumber. Anda bisa mendapatkan alat ini dengan membuat tiruan dari repo alat Amazon GitHub DocumentDB dan mengikuti instruksi di. `README.md` Anda dapat menjalankan alat dari instans Amazon EC2 atau AWS Cloud9 lingkungan yang berjalan di VPC Amazon yang sama dengan cluster Amazon DocumentDB Anda.

Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi Anda sendiri.

Kode berikut membuang indeks dari cluster Amazon DocumentDB sumber Anda:

```
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/
```

Setelah indeks Anda berhasil diekspor, pulihkan indeks tersebut di cluster Amazon DocumentDB target Anda. Untuk memulihkan indeks yang Anda ekspor di langkah sebelumnya, gunakan Alat Indeks Amazon DocumentDB. Perintah berikut mengembalikan indeks di cluster Amazon DocumentDB target Anda dari direktori yang ditentukan.

```
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
```

Untuk mengonfirmasi bahwa Anda memulihkan indeks dengan benar, sambungkan ke cluster Amazon DocumentDB target Anda dengan shell mongo dan daftarkan indeks untuk koleksi tertentu. Jalankan kode berikut:

```
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()
```

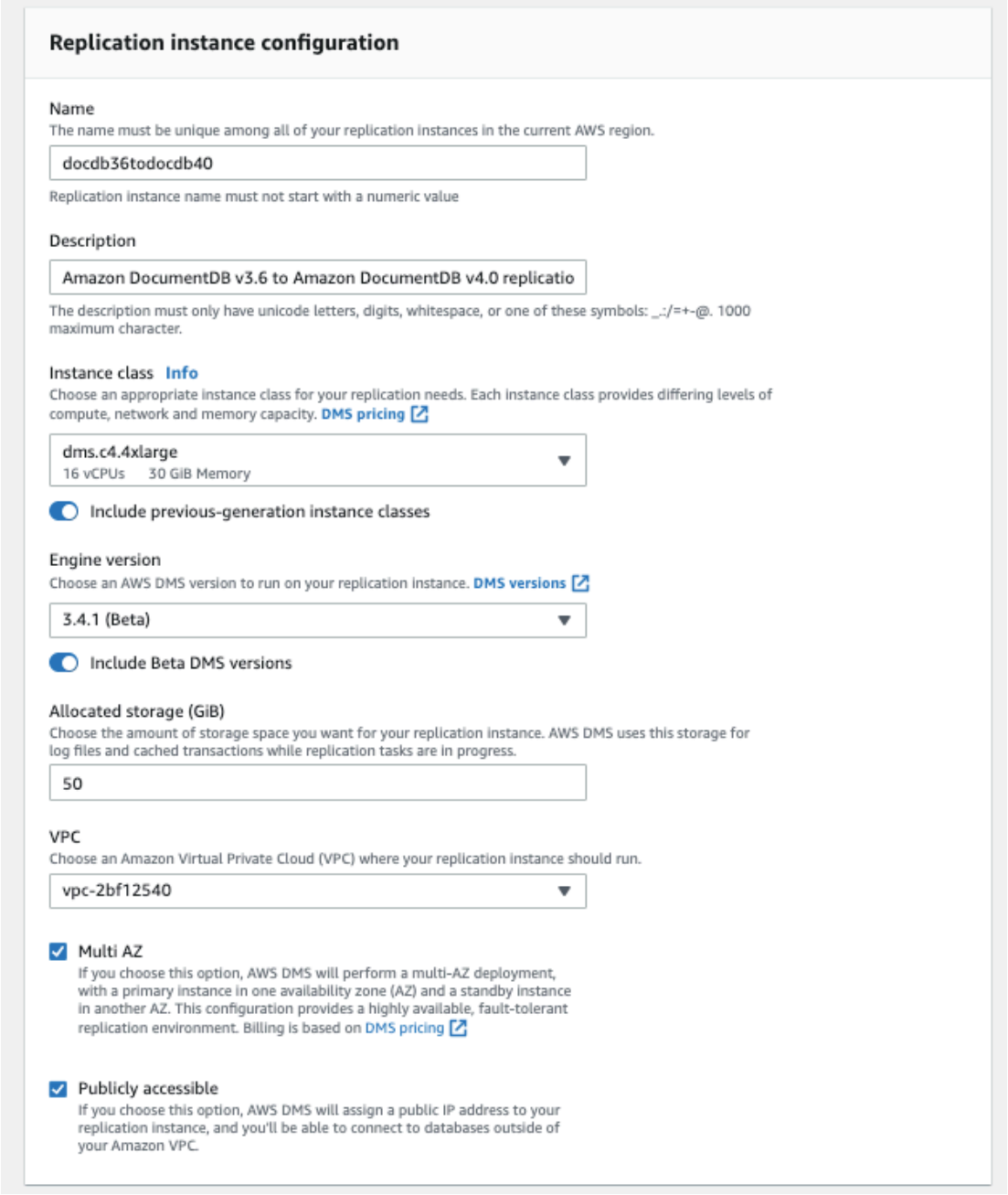
Langkah 4: Buat AWS DMS Tipe instans replikasi

Instans AWS DMS replikasi menghubungkan dan membaca data dari kluster Amazon DocumentDB sumber Anda dan menuliskannya sebagai klaster Amazon DocumentDB target Anda. Instance AWS DMS replikasi dapat melakukan operasi beban massal dan CDC. Sebagian besar pemrosesan ini terjadi di memori. Namun, operasi besar mungkin memerlukan buffering pada disk. Transaksi cache dan file log juga ditulis pada disk. Setelah data dimigrasikan, instans replikasi juga mengalirkan peristiwa perubahan apa pun untuk memastikan sinkronisasi antara sumber dan target.

Untuk membuat contoh AWS DMS replikasi:

1. Buka AWS DMS konsol.
2. Di panel navigasi, pilih Instans replikasi.
3. Pilih Membuat instans replikasi dan masukkan informasi berikut:
   - Untuk Nama, masukkan nama pilihan Anda. Misalnya, `docdb36todocdb40`.
   - Untuk Deskripsi, masukkan deskripsi pilihan Anda. Untuk item daftar, instans replikasi Amazon DocumentDB 3.6 ke Amazon DocumentDB 4.0.
   - Untuk Kelas instans, pilih ukuran berdasarkan kebutuhan Anda.
   - Untuk Versi mesin, pilih `3.4.1`.
   - Untuk Amazon VPC, pilih Amazon VPC yang menampung sumber Anda dan menargetkan cluster Amazon DocumentDB.
   - Untuk Penyimpanan yang dialokasikan (GiB), gunakan default 50 GiB. Jika Anda memiliki beban kerja throughput tulis yang tinggi, tingkatkan nilai ini agar sesuai dengan beban kerja Anda.

- Untuk Multi-AZ, pilih Ya jika Anda membutuhkan ketersediaan tinggi dan dukungan failover.
- Untuk Dapat diakses publik, aktifkan opsi ini.



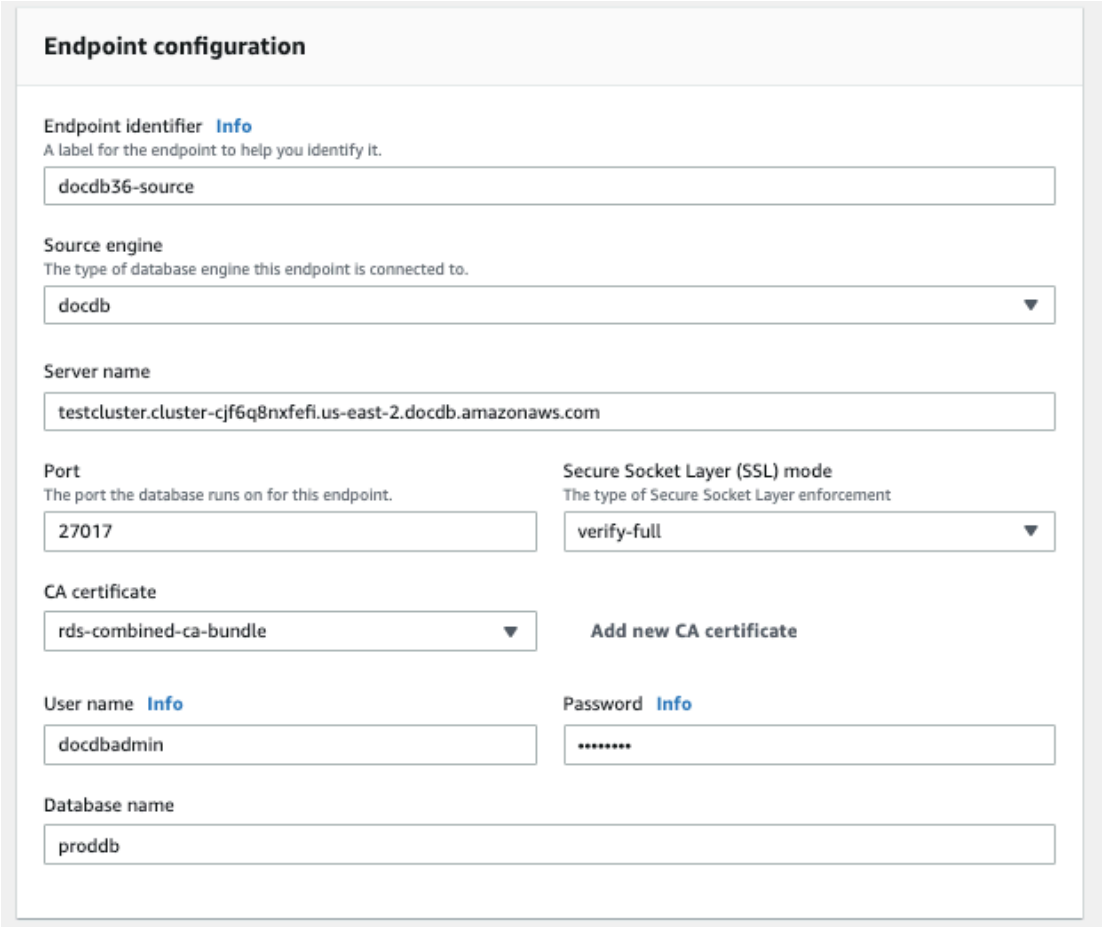
4. Pilih Buat instans replikasi.

Langkah 5: Buat AWS DMS titik akhir sumber

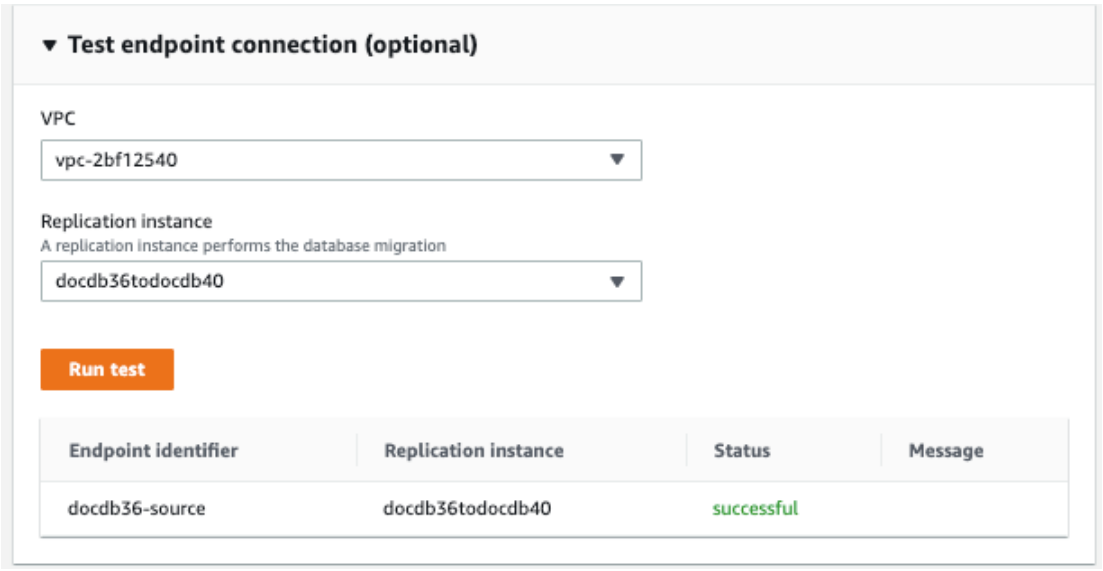
Titik akhir sumber digunakan untuk cluster Amazon DocumentDB sumber.

Untuk membuat titik akhir sumber

1. Buka AWS DMS konsol.
2. Di panel navigasi, pilih Titik Akhir.
3. Pilih `Create endpoint`, dan masukkan informasi berikut:
   - Untuk Jenis titik akhir, pilih Sumber.
   - >Untuk Pengenal Titik Akhir, masukkan nama yang mudah diingat, misalnya `docdb-source`.
   - Untuk Mesin sumber, pilih `docdb`.
   - Untuk nama Server, masukkan nama DNS dari cluster Amazon DocumentDB sumber Anda.
   - Untuk Port, masukkan nomor port cluster Amazon DocumentDB sumber Anda.
   - Untuk Mode SSL, pilih `verify-full`.
   - Untuk Sertifikat CA, pilih Menambahkan sertifikat CA baru. Unduh sertifikat CA baru untuk membuat paket koneksi TLS. Untuk Pengenal sertifikat, masukkan `rds-combined-ca-bundle` . Untuk Impor file sertifikat, pilih Pilih file dan navigasikan ke `.pem` yang Anda unduh sebelumnya. Pilih dan buka file. Pilih Impor sertifikat, lalu pilih `rds-combined-ca-bundle` dari menu pilihan menurun Pilih sertifikat
   - Untuk nama Pengguna, masukkan nama pengguna utama cluster Amazon DocumentDB sumber Anda.
   - Untuk Kata Sandi, masukkan kata sandi utama cluster Amazon DocumentDB sumber Anda.
   - Untuk Nama basis data, masukkan nama basis data yang Anda cari untuk melakukan peningkatan.



4. Uji koneksi Anda untuk memverifikasi keberhasilan pengaturannya.



5. Pilih Buat Titik Akhir.

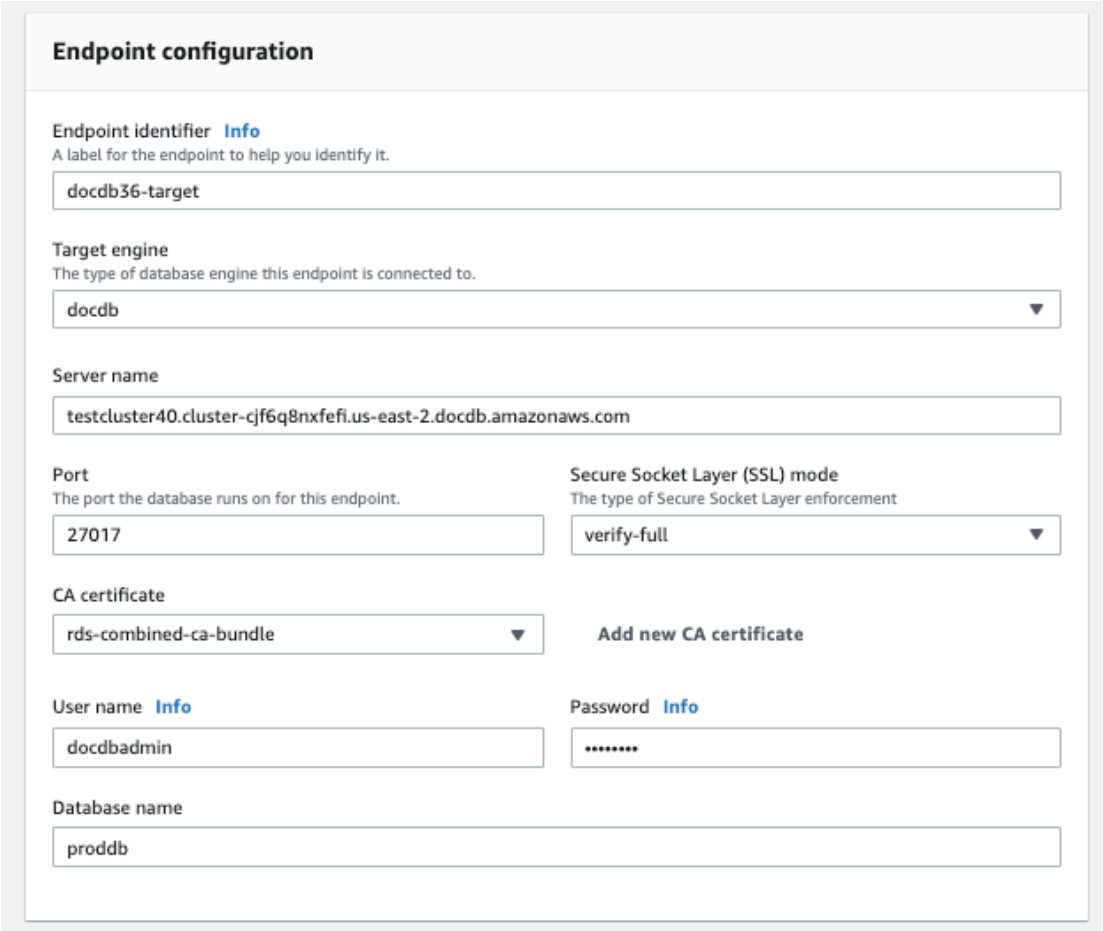
> **Note**
>
> AWS DMS hanya dapat memigrasikan satu database pada satu waktu.

Langkah 6: Buat AWS DMS titik akhir target

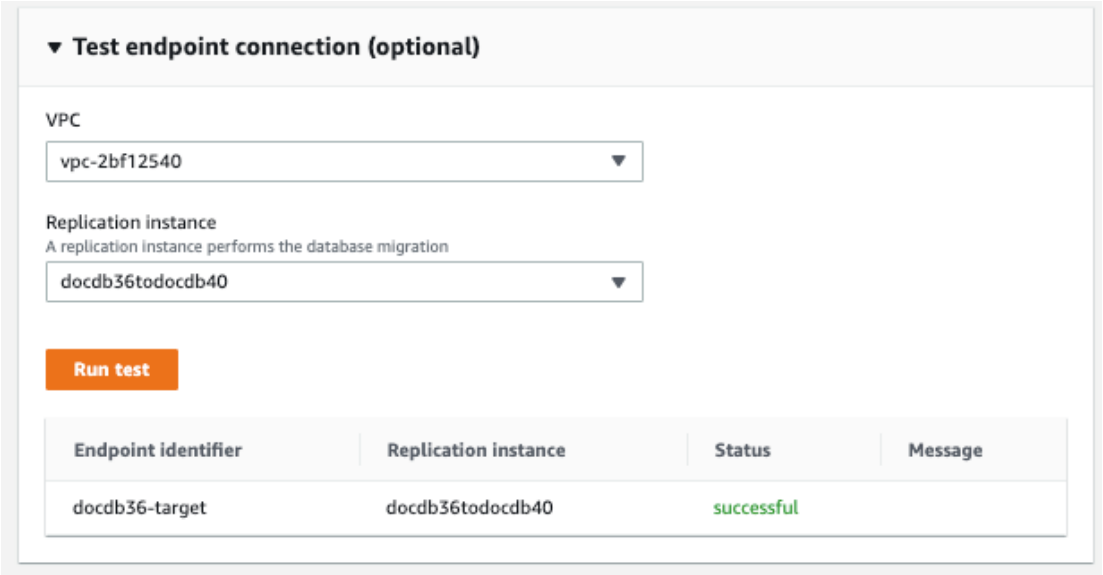
Titik akhir target adalah untuk cluster Amazon DocumentDB target Anda.

Untuk membuat titik akhir target:

1. Buka konsol AWS DMS.
2. Di panel navigasi, pilih Titik akhir.
3. Pilih Buat titik akhir dan masukkan informasi berikut:
   - Untuk Jenis titik akhir, pilih Target.
   - Untuk Pengenal Titik Akhir, masukkan nama yang mudah diingat, misalnya `docdb-target`.
   - Untuk Mesin sumber, pilih `docdb`.
   - Untuk nama Server, masukkan nama DNS kluster Amazon DocumentDB target Anda.
   - Untuk Port, masukkan nomor port cluster Amazon DocumentDB target Anda.
   - Untuk Mode SSL, pilih `verify-full`.
   - Untuk sertifikat CA, pilih sertifikat yang ada `rds-combined-ca-bundle` dari pilihan menurun Pilih sertifikat.
   - Untuk nama Pengguna, masukkan nama pengguna utama klaster Amazon DocumentDB target Anda.
   - Untuk Kata Sandi, masukkan kata sandi utama klaster Amazon DocumentDB target Anda.
   - Untuk nama Basis data, masukkan nama basis data yang sama dengan yang Anda gunakan untuk mengatur titik akhir sumber Anda.



4. Uji koneksi Anda untuk memverifikasi keberhasilan pengaturannya.



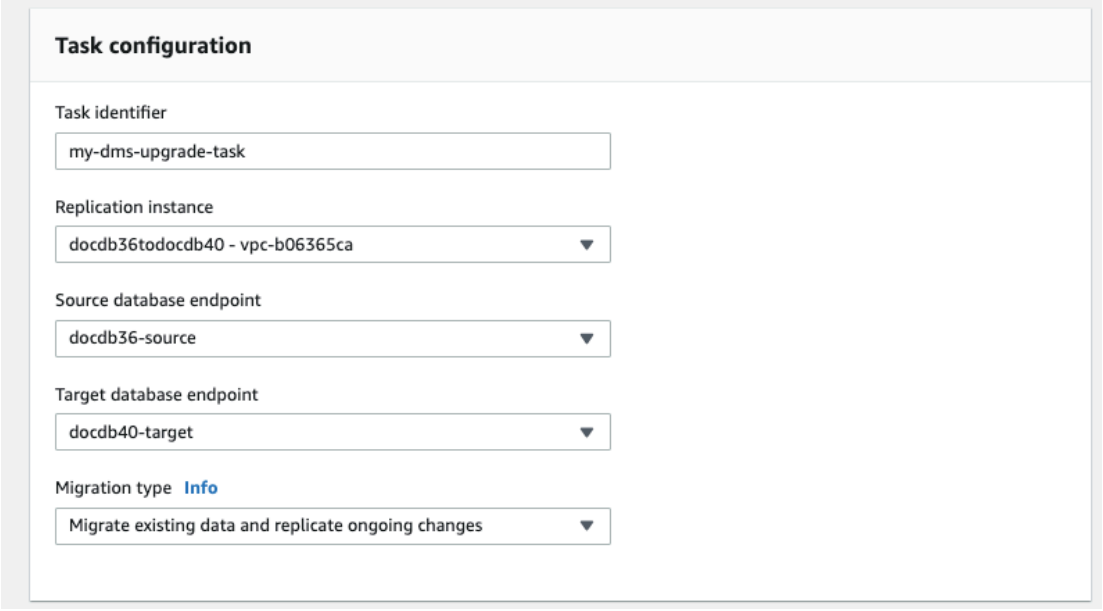
5. Pilih Buat Titik Akhir.

Langkah 7: Buat dan jalankan tugas migrasi

AWS DMS Tugas mengikat instance replikasi dengan instance sumber dan target Anda. Saat membuat tugas migrasi, Anda menentukan titik akhir sumber, titik akhir target, contoh replikasi, dan setelan migrasi yang diinginkan. AWS DMS Tugas dapat dibuat dengan tiga jenis migrasi yang berbeda - memigrasikan data yang ada, memigrasikan data yang ada, dan mereplikasi perubahan yang sedang berlangsung atau hanya mereplikasi perubahan data. Karena tujuan dari panduan ini adalah untuk memutakhirkan cluster Amazon DocumentDB dengan waktu henti minimal, langkah-langkahnya menggunakan opsi untuk memigrasikan data yang ada dan mereplikasi perubahan yang sedang berlangsung. Dengan opsi ini, AWS DMS menangkap perubahan saat memigrasi data yang ada. AWS DMS terus menangkap dan menerapkan perubahan bahkan setelah data massal dimuat. Terakhir, basis data sumber dan target akan disinkronkan, sehingga migrasi waktu henti yang minimal bisa didapatkan.

Berikut adalah langkah-langkah untuk membuat tugas migrasi untuk migrasi downtime minimal:

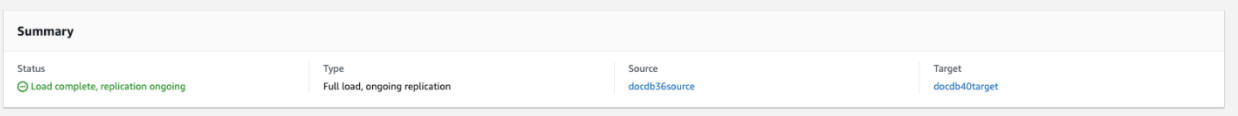
1. Buka AWS DMS konsol.
2. Di panel navigasi, pilih Tugas migrasi database.
3. Pilih Buat tugas migrasi database dan masukkan informasi berikut di bagian Konfigurasi tugas:
   - Untuk pengenal Tugas, masukkan nama yang mudah diingat, misalnya`my-dms-upgrade-task`.
   - Untuk Nama Sumber Daya Amazon Deskriptif (ARN), masukkan nama yang mudah digunakan untuk mengganti ARN DMS default.
   - Untuk contoh Replikasi, pilih contoh replikasi yang Anda buat. Langkah 4: Buat AWS DMS Tipe instans replikasi
   - Untuk titik akhir basis data Sumber, pilih titik akhir sumber yang Anda buat. Langkah 5: Buat AWS DMS titik akhir sumber
   - Untuk titik akhir database Target, pilih titik akhir target yang Anda buat. Langkah 6: Buat AWS DMS titik akhir target
   - Untuk jenis Migrasi, pilih Migrasi dan replikasi.



4. Masukkan informasi berikut di bagian Pengaturan tugas:

   - Untuk bagian mode persiapan tabel Target, pilih Jangan lakukan apa-apa. Ini akan memastikan bahwa indeks yang dibuat pada langkah 3 tidak akan dijatuhkan.
   - Untuk sub-bagian Log tugas, pilih Aktifkan CloudWatch log.
   - Untuk konfigurasi memulai tugas migrasi, pilih Otomatis saat pembuatan. Pilihan ini akan memulai tugas migrasi secara otomatis setelah Anda membuatnya.
   - Pilih Buat tugas migrasi database.

AWS DMS sekarang mulai memigrasikan data dari cluster Amazon DocumentDB sumber Anda ke cluster Amazon DocumentDB target Anda. Status tugas harus berubah dari Mulai Menjalankan. Anda dapat memantau kemajuan dengan memilih Tugas di AWS DMS konsol. Setelah beberapa minutes/hours (tergantung pada ukuran migrasi Anda), status akan berubah dari ke Load complete, replikasi sedang berlangsung. Ini berarti bahwa AWS DMS telah menyelesaikan migrasi muatan penuh dari cluster Amazon DocumentDB sumber Anda ke cluster Amazon DocumentDB target dan sekarang mereplikasi peristiwa perubahan.



Akhirnya sumber dan target Anda akan sinkron. Anda dapat memverifikasi apakah semuanya telah sinkron dengan menjalankan operasi `count()` pada koleksi Anda guna memverifikasi bahwa semua peristiwa perubahan telah dimigrasikan.

Langkah 8: Mengubah endpoint aplikasi ke cluster Amazon DocumentDB target

Setelah pemuatan penuh selesai dan proses CDC mereplikasi terus menerus, Anda siap untuk mengubah titik akhir koneksi database aplikasi Anda dari cluster Amazon DocumentDB sumber Anda ke cluster Amazon DocumentDB target Anda.

Menggunakan rilis dukungan jangka panjang (LTS)

Amazon DocumentDB menetapkan versi tertentu sebagai rilis Long-Term Support (LTS). Cluster database yang menggunakan LTS dapat tetap pada versi yang sama lebih lama dan mengalami siklus peningkatan yang lebih sedikit daripada cluster yang menggunakan rilis non-LTS. Versi LTS akan ditambal hanya jika terjadi stabilitas kritis dan patch keamanan tanpa memperkenalkan fitur baru. Patch opsional tambahan dapat dirilis seperti metrik untuk men-debug kegagalan kritis. Ini akan diterapkan secara kumulatif dengan tambalan yang diperlukan.

Rilis LTS Saat Ini

Rilis LTS saat ini untuk Amazon DocumentDB adalah:

- Versi Minor 5.0.0

Untuk menentukan versi yang digunakan cluster Anda, lihatthe section called “Cara memeriksa apakah cluster mesin versi 5.0.0 saya yang ada ada di LTS”.

Untuk detail tentang jadwal dukungan dan siklus rilis untuk versi LTS, lihat tanggal dukungan versi mesin Amazon DocumentDB. Amazon DocumentDB akan mengumumkan versi LTS berikutnya melalui saluran komunikasi standar.

> **Note**
>
> Rilis LTS untuk Amazon DocumentDB 8.0 belum ditetapkan. Jika Anda memerlukan LTS, gunakan engine versi 5.0.0 dengan Patch Version 3.0.17983 atau yang lebih baru.

Siapa yang Harus Menggunakan LTS

Rilis Amazon DocumentDB LTS dirancang untuk cluster dengan jendela pemutakhiran terbatas. LTS sangat ideal untuk lingkungan produksi di mana ketersediaan database secara langsung berdampak pada operasi bisnis.

Pertimbangkan rilis LTS jika:

- Aplikasi Anda memiliki persyaratan waktu aktif yang ketat yang membatasi peluang peningkatan
- Anda memiliki semua fitur mesin database yang diperlukan dan penyempurnaan kebutuhan aplikasi Anda

Pertimbangkan rilis standar sebagai gantinya jika:

- Aplikasi Anda dapat menangani gangguan minimal
- Anda menginginkan kemampuan dan penyempurnaan terbaru segera setelah dirilis

> **Important**
>
> - Upgrade adalah operasi satu arah - Anda tidak dapat menurunkan versi setelah memutakhirkan
> - Cluster Anda akan mengalami downtime singkat selama proses upgrade

Cara membuat cluster LTS baru

Untuk membuat cluster LTS, ikuti langkah-langkah berikut:

1. Di Management Console, buat cluster engine versi 5.0.0 baru
2. Verifikasi Anda menggunakan engine versi 5.0.0 setelah pembuatan cluster menggunakan: AWS CLI

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

Keluaran dari operasi ini terlihat seperti berikut ini:

```
"5.0.0"
```

Cara memutakhirkan dari versi mesin 3.6.0 atau 4.0.0 ke cluster LTS 5.0.0

Untuk meningkatkan ke klaster LTS, ikuti langkah-langkah berikut:

1. Di Management Console, upgrade versi utama (MVU) cluster engine versi 3.6.0 atau 4.0.0 ke 5.0.0 setelah Upgrade versi engine Amazon DocumentDB
2. Verifikasi Anda menggunakan engine versi 5.0.0 setelah memutakhirkan menggunakan: AWS CLI

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

Cara memeriksa apakah cluster mesin versi 5.0.0 saya yang ada ada di LTS

Untuk memeriksa apakah cluster engine versi 5.0.0 Anda menjalankan versi LTS:

1. Verifikasi Anda menggunakan engine versi 5.0.0 menggunakan: AWS CLI

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

2. Jika diperlukan, tingkatkan cluster engine versi 5.0.0 yang ada setelah Melakukan pembaruan tambalan ke versi mesin cluster

Amazon DocumentDB Extended Support

Amazon DocumentDB Extended Support memungkinkan Anda untuk terus menjalankan versi engine melewati akhir tanggal dukungan standar dengan biaya tambahan. Cluster Anda terdaftar ke Extended Support setelah melewati akhir tanggal dukungan standar. Jika Anda tidak menyelesaikan upgrade pada akhir tanggal dukungan standar versi engine, cluster Anda akan ditagih pada tingkat Extended Support.

Topik

- Ikhtisar Extended Support
- Biaya Extended Support
- Tanggung Jawab

Amazon DocumentDB Extended Support menyediakan pembaruan dan dukungan teknis berikut:

- Pembaruan keamanan untuk CVE kritis dan tinggi untuk instans atau klaster Anda
- Perbaikan bug dan tambalan untuk masalah kritis
- Kemampuan untuk membuka kasus dukungan dan menerima bantuan pemecahan masalah dalam perjanjian tingkat layanan Amazon DocumentDB standar

Penawaran berbayar ini memberi Anda lebih banyak waktu untuk meningkatkan ke versi mesin yang didukung. Misalnya, Amazon DocumentDB akhir tanggal dukungan standar untuk versi 3.6 adalah 30 Maret 2026, namun Anda belum siap untuk meningkatkan ke versi 5.0 atau 8.0 sebelum tanggal tersebut. Dalam hal ini, Amazon DocumentDB mendaftarkan klaster versi 3.6 Anda di Extended Support pada 31 Maret 2026 sehingga Anda dapat terus menggunakan Amazon DocumentDB versi 3.6. Mulai 1 Juli 2026, AWS menagih Anda premium Extended Support untuk klaster Amazon DocumentDB versi 3.6.

Amazon DocumentDB Extended Support tersedia hingga 3 tahun setelah tanggal dukungan standar versi engine. Setelah waktu ini, jika Anda belum memutakhirkan versi mesin Anda ke versi yang didukung, Amazon DocumentDB akan meningkatkan versi mesin Anda. Kami menyarankan Anda meningkatkan ke versi mesin yang didukung sesegera mungkin.

Amazon DocumentDB versi 3.6 akhir tanggal dukungan standar dan tanggal Dukungan Diperpanjang:

- Rilis versi 3.6 - 9 Januari 2019
- Dukungan standar akhir versi 3.6 - 30 Maret 2026
- Mulai dari versi 3.6 Extended Support tahun 1 harga - 1 Juli 2026
- Mulai dari versi 3.6 Extended Support tahun 3 harga - 31 Maret 2028
- Akhir versi 3.6 Extended Support - 30 Maret 2029

Ikhtisar Extended Support

Amazon DocumentDB memutakhirkan klaster Anda ke versi engine terakhir yang dirilis (sebelum akhir tanggal Extended Support), jika klaster Anda belum ada di versi tersebut. Periode upgrade tidak akan terjadi sampai setelah tanggal dukungan standar untuk versi mesin tersebut.

Anda dapat membuat cluster baru dengan versi mesin yang telah mencapai akhir tanggal dukungan standar, jika Anda sudah memiliki cluster pada versi mesin itu. Amazon DocumentDB mendaftarkan cluster baru ini di Amazon DocumentDB Extended Support dan menagih Anda untuk penawaran ini. Jika Anda meningkatkan ke versi engine yang masih di bawah dukungan standar Amazon DocumentDB sebelum tanggal dukungan standar versi 3.6 berakhir, Anda tidak akan dikenakan biaya premium Extended Support.

Anda dapat mengakhiri pendaftaran di Amazon DocumentDB Extended Support kapan saja. Untuk mengakhiri pendaftaran, tingkatkan setiap cluster yang terdaftar ke versi engine yang lebih baru yang masih di bawah dukungan standar Amazon DocumentDB. Akhir pendaftaran Amazon DocumentDB Extended Support akan efektif pada hari Anda menyelesaikan upgrade ke versi mesin yang lebih baru yang masih di bawah dukungan standar Amazon DocumentDB.

Untuk informasi selengkapnya tentang Amazon DocumentDB akhir tanggal dukungan standar dan Amazon DocumentDB akhir tanggal Extended Support, lihat. Tanggal dukungan versi mesin Amazon DocumentDB

Biaya Extended Support

Anda akan dikenakan biaya untuk semua cluster yang berjalan di Extended Support mulai 1 Juli 2026. Untuk akhir tanggal dukungan standar Amazon DocumentDB versi 3.6, lihat. Tanggal dukungan versi mesin Amazon DocumentDB

Biaya tambahan untuk Amazon DocumentDB Extended Support berhenti ketika Anda melakukan salah satu tindakan berikut:

- Tingkatkan ke versi mesin yang tercakup dalam dukungan standar
- Hapus cluster yang menjalankan versi 3.6 melewati akhir tanggal dukungan standar

Misalnya, Amazon DocumentDB versi 3.6 memasuki Extended Support pada 31 Maret 2026, tetapi biaya tidak dimulai hingga 1 Juli 2026. Jika Anda memutakhirkan cluster Amazon DocumentDB versi 3.6 ke Amazon DocumentDB versi 5.0 pada 31 Juli 2026, Anda hanya akan dikenakan biaya selama 30 hari Extended Support di Amazon DocumentDB versi 3.6. Untuk informasi selengkapnya, lihat harga Amazon DocumentDB.

Tanggung Jawab

Berikut ini adalah tanggung jawab Amazon DocumentDB dan tanggung jawab Anda dengan Amazon DocumentDB Extended Support.

Tanggung jawab Amazon DocumentDB

Setelah tanggal dukungan standar Amazon DocumentDB versi 3.6 berakhir, Amazon DocumentDB akan menerapkan tambalan, perbaikan bug, dan peningkatan untuk klaster versi 3.6 yang terdaftar di Extended Support.

Tanggung jawab Anda

Anda bertanggung jawab untuk meningkatkan mesin Anda ke versi mesin yang lebih baru sebelum Amazon DocumentDB versi 3.6 berakhir tanggal Extended Support. Jika Anda tidak memutakhirkan klaster Anda, setelah Amazon DocumentDB versi 3.6 berakhir tanggal Extended Support, Amazon DocumentDB akan mencoba untuk meng-upgrade klaster Anda ke versi engine terbaru yang didukung di bawah dukungan standar Amazon DocumentDB.

Keamanan di Amazon DocumentDB

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai AWS pelanggan, Anda mendapat manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. Dokumentasi ini membantu Anda memahami cara untuk menerapkan model tanggung jawab bersama saat menggunakan Amazon DocumentDB. Model tanggung jawab bersama menggambarkan hal ini sebagai keamanan dari cloud dan keamanan di cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan AWS layanan di AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Third-party Auditor secara teratur menguji dan memverifikasi efektivitas keamanan kami sebagai bagian dari program AWS kepatuhan. Untuk mempelajari tentang program kepatuhan yang berlaku untuk Amazon DocumentDB (dengan kompatibilitas MongoDB), lihat Layanan AWS dalam Cakupan melalui Program Kepatuhan.
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh AWS layanan yang Anda gunakan. Anda juga bertanggung jawab atas faktor-faktor lain termasuk sensitivitas data, kebutuhan organisasi, serta undang-undang dan peraturan yang berlaku.

Amazon DocumentDB disahkan berdasarkan Federal Risk and Authorization Management Program (FedRAMP). Ini memiliki otorisasi FedRAMP High AWS GovCloud untuk wilayah (AS) dan otorisasi FedRAMP Moderate untuk Wilayah AS. AWS East/West Untuk detail tentang AWS dan upaya kepatuhan, lihat AWS Layanan dalam Lingkup oleh Program Kepatuhan.

> **Note**
>
> Bab ini berlaku untuk cluster berbasis instance dan Elastic Cluster. Untuk informasi selengkapnya, lihat topik di bawah.

Anda juga mempelajari cara menggunakan AWS layanan lain yang membantu Anda memantau dan mengamankan sumber daya Amazon DocumentDB Anda. Topik berikut ini menunjukkan kepada Anda cara untuk mengonfigurasi Amazon DocumentDB untuk memenuhi tujuan keamanan dan kepatuhan Anda.

Topik

- Manajemen kata sandi dengan Amazon DocumentDB dan AWS Secrets Manager
- Perlindungan data di Amazon DocumentDB
- Identity and Access Management untuk Amazon DocumentDB
- Cross-service pencegahan wakil bingung
- Otentikasi menggunakan identitas IAM
- Mengelola pengguna Amazon DocumentDB
- Akses database menggunakan Role-Based Access Control
- Pencatatan dan pemantauan di Amazon DocumentDB
- Memperbarui sertifikat Amazon DocumentDB TLS Anda
- Validasi kepatuhan di Amazon DocumentDB
- Ketahanan di Amazon DocumentDB
- Keamanan infrastruktur di Amazon DocumentDB
- Amazon DocumentDB API dan titik akhir VPC antarmuka (AWS PrivateLink)
- Praktik terbaik keamanan untuk Amazon DocumentDB
- Mengaudit acara Amazon DocumentDB
- Amazon VPC dan Amazon DocumentDB

Manajemen kata sandi dengan Amazon DocumentDB dan AWS Secrets Manager

Amazon DocumentDB terintegrasi dengan Secrets Manager untuk mengelola kata sandi pengguna utama untuk cluster Anda.

Topik

- Batasan untuk integrasi Secrets Manager dengan Amazon DocumentDB
- Ikhtisar mengelola kata sandi pengguna utama dengan AWS Secrets Manager
- Menegakkan pengelolaan Amazon DocumentDB dari kata sandi pengguna utama di AWS Secrets Manager
- Mengelola kata sandi pengguna utama untuk klaster dengan Secrets Manager

Batasan untuk integrasi Secrets Manager dengan Amazon DocumentDB

Mengelola kata sandi pengguna utama dengan Secrets Manager tidak didukung untuk fitur berikut:

- Cluster yang merupakan bagian dari database global Amazon DocumentDB
- Amazon DocumentDB Replika baca lintas wilayah

Ikhtisar mengelola kata sandi pengguna utama dengan AWS Secrets Manager

Dengan AWS Secrets Manager, Anda dapat mengganti kredensi hard-code dalam kode Anda, termasuk kata sandi database, dengan panggilan API ke Secrets Manager untuk mengambil rahasia secara terprogram. Untuk informasi selengkapnya tentang Secrets Manager, lihat Panduan Pengguna AWS Secrets Manager.

Ketika Anda menyimpan rahasia database di Secrets Manager, AWS akun Anda dikenakan biaya. Untuk informasi lebih lanjut tenngenai harga, lihat harga AWS Secrets Manager.

Anda dapat menentukan bahwa Amazon DocumentDB mengelola kata sandi pengguna utama di Secrets Manager untuk klaster Amazon DocumentDB saat Anda melakukan salah satu operasi berikut:

- Buat cluster
- Memodifikasi cluster

Saat Anda menentukan bahwa Amazon DocumentDB mengelola kata sandi pengguna utama di Secrets Manager, Amazon DocumentDB menghasilkan kata sandi dan menyimpannya di Secrets Manager. Anda dapat berinteraksi langsung dengan rahasia untuk mengambil kredensil untuk pengguna utama. Anda juga dapat menentukan kunci yang dikelola pelanggan untuk mengenkripsi rahasia, atau menggunakan kunci KMS default yang disediakan oleh Secrets Manager.

Amazon DocumentDB mengelola pengaturan untuk rahasia dan memutar rahasia setiap tujuh hari secara default. Anda dapat mengubah beberapa pengaturan, seperti jadwal rotasi. Jika Anda menghapus klaster yang mengelola rahasia di Secrets Manager, rahasia dan metadata terkait juga akan dihapus.

Untuk terhubung ke cluster dengan kredensialnya secara rahasia, Anda dapat mengambil rahasia dari Secrets Manager. Untuk informasi selengkapnya, lihat Mendapatkan rahasia dari AWS Secrets

Manager dan Connect ke database SQL menggunakan JDBC dengan kredensyal dalam AWS Secrets Manager rahasia di Panduan Pengguna.AWS Secrets Manager

Menegakkan pengelolaan Amazon DocumentDB dari kata sandi pengguna utama di AWS Secrets Manager

Anda dapat menggunakan kunci kondisi IAM untuk menerapkan pengelolaan Amazon DocumentDB dari kata sandi pengguna utama. AWS Secrets Manager Kebijakan berikut tidak mengizinkan pengguna untuk membuat atau memulihkan instance atau cluster kecuali kata sandi pengguna utama dikelola oleh Amazon DocumentDB di Secrets Manager.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "rds:CreateDBCluster"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "rds:ManageMasterUserPassword": false
                }
            }
        }
    ]
}
```

Mengelola kata sandi pengguna utama untuk klaster dengan Secrets Manager

Anda dapat mengonfigurasi pengelolaan Amazon DocumentDB dari kata sandi pengguna utama di Secrets Manager saat Anda melakukan tindakan berikut:

- Membuat cluster Amazon DocumentDB

- Memodifikasi cluster Amazon DocumentDB

Anda dapat menggunakan konsol Amazon DocumentDB atau untuk melakukan tindakan AWS CLI ini.

Perlindungan data di Amazon DocumentDB

Model tanggung jawab AWS bersama model tanggung berlaku untuk perlindungan data di Amazon DocumentDB. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk mempertahankan kendali atas konten yang di-host pada infrastruktur ini. Anda juga bertanggung jawab atas tugas-tugas konfigurasi dan manajemen keamanan untuk Layanan AWS yang Anda gunakan. Untuk informasi selengkapnya tentang privasi data, lihat FAQ Privasi Data AWS . Untuk informasi tentang perlindungan data di Eropa, lihat Pusat Peraturan Umum Perlindungan Data (GDPR).

Untuk tujuan perlindungan data, kami menyarankan Anda melindungi Akun AWS kredensil dan mengatur pengguna individu dengan AWS IAM Identity Center atau AWS Identity and Access Management (IAM). Dengan cara itu, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugasnya. Kami juga menyarankan supaya Anda mengamankan data dengan cara-cara berikut:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk berkomunikasi dengan AWS sumber daya. Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail. Untuk informasi tentang penggunaan CloudTrail jejak untuk menangkap AWS aktivitas, lihat Bekerja dengan CloudTrail jejak di AWS CloudTrail Panduan Pengguna.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya Layanan AWS.
- Gunakan layanan keamanan terkelola tingkat lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-3 saat mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi selengkapnya tentang titik akhir FIPS yang tersedia di Standar Pemrosesan Informasi Federal (FIPS) 140-3.

Kami sangat merekomendasikan agar Anda tidak pernah memasukkan informasi identifikasi yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam tanda atau bidang isian bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan Amazon DocumentDB atau Layanan AWS lainnya menggunakan konsol, API AWS CLI, atau SDK. AWS Data apa pun yang Anda masukkan ke dalam tanda atau bidang isian bebas yang digunakan untuk nama dapat digunakan untuk log penagihan atau log diagnostik. Saat Anda memberikan URL ke server eksternal, kami sangat menganjurkan supaya Anda tidak menyertakan informasi kredensial di dalam URL untuk memvalidasi permintaan Anda ke server itu.

Topik

- Client-side enkripsi tingkat lapangan
- Mengenkripsi data Amazon DocumentDB saat istirahat
- Mengenkripsi data dalam perjalanan
- Manajemen kunci

Client-side enkripsi tingkat lapangan

Amazon DocumentDB client-side field level encryption (FLE) memungkinkan Anda mengenkripsi data sensitif dalam aplikasi klien Anda sebelum ditransfer ke cluster Amazon DocumentDB. Data sensitif tetap dienkripsi ketika disimpan dan diproses dalam cluster dan didekripsi pada aplikasi klien ketika diambil.

Topik

- Memulai
- Menanyakan di FLE sisi klien

Memulai

Konfigurasi awal FLE sisi klien di Amazon DocumentDB adalah proses empat langkah yang mencakup pembuatan kunci enkripsi, mengaitkan peran ke aplikasi, mengonfigurasi aplikasi, dan mendefinisikan operasi CRUD dengan opsi enkripsi.

Topik

- Langkah 1: Buat kunci enkripsi
- Langkah 2: Kaitkan peran dengan aplikasi

- Langkah 3: Konfigurasikan aplikasi
- Langkah 4: Tentukan operasi CRUD
- Contoh: file konfigurasi enkripsi tingkat bidang sisi klien

Langkah 1: Buat kunci enkripsi

Menggunakan AWS Key Management Service, membuat kunci simetris yang digunakan untuk mengenkripsi dan mendekripsi bidang data sensitif dan memberikan izin penggunaan IAM yang diperlukan. AWS KMS menyimpan Kunci Pelanggan (CK) yang digunakan untuk mengenkripsi Kunci Data (DK). Simpan Kunci Pelanggan di KMS untuk memperkuat postur keamanan Anda. Kunci Data adalah kunci sekunder yang disimpan dalam koleksi Amazon DocumentDB dan diperlukan untuk mengenkripsi bidang sensitif sebelum menyimpan dokumen di Amazon DocumentDB. Kunci Pelanggan mengenkripsi Kunci Data yang pada gilirannya mengenkripsi dan mendekripsi data Anda. Jika Anda menggunakan klaster global, Anda dapat membuat kunci multi-wilayah yang dapat digunakan oleh peran layanan yang berbeda di berbagai wilayah.

Untuk informasi selengkapnya tentang AWS Key Management Service, termasuk cara membuat kunci, lihat Panduan Pengembang Layanan Manajemen AWS Kunci.

Langkah 2: Kaitkan peran dengan aplikasi

Buat kebijakan IAM dengan AWS KMS izin yang sesuai. Kebijakan ini memungkinkan identitas IAM yang dilampirkan untuk mengenkripsi dan mendekripsi kunci KMS yang ditentukan dalam bidang sumber daya. Aplikasi Anda mengasumsikan peran IAM ini untuk diautentikasi dengan. AWS KMS

Kebijakan harus terlihat mirip dengan ini:

```
{ "Effect": "Allow",
"Action": ["kms:Decrypt", "kms:Encrypt"],
"Resource": "Customer Key ARN"
}
```

Langkah 3: Konfigurasikan aplikasi

Sekarang Anda menetapkan Kunci Pelanggan AWS KMS dan membuat peran IAM dan memberikannya izin IAM yang tepat untuk mengakses Kunci Pelanggan. Impor paket yang diperlukan.

```
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. Tentukan 'aws' sebagai jenis penyedia KMS dan masukkan kredensyal akun Anda yang diambil pada langkah sebelumnya.

   ```
   provider = "aws"
   kms_providers = {
       provider: {
           "accessKeyId": current_credentials.access_key,
           "secretAccessKey": current_credentials.secret_key
       }
   }
   ```

2. Tentukan kunci pelanggan yang digunakan untuk mengenkripsi kunci data:

   ```
   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. Konfigurasikan MongoClient objek:

   ```
   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. Hasilkan Kunci Data Anda:

```
data_key_id = client_encryption.create_data_key(provider,
    customer_key,
    key_alt_name = [key_alt_name])
```

5. Ambil Kunci Data Anda yang ada:

```
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})
```

Langkah 4: Tentukan operasi CRUD

Tentukan operasi CRUD dengan opsi enkripsi.

1. Tentukan koleksi write/read ke/hapus satu dokumen:

```
coll = client.gameinfo.users
```

2. Enkripsi Eksplisit - mengenkripsi bidang dan menyisipkan:

> **Note**
>
> Tepat salah satu dari “key_id” atau “key_alt_name” harus disediakan.

```
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"]
})
```

Contoh: file konfigurasi enkripsi tingkat bidang sisi klien

Dalam contoh berikut, ganti masing-masing *placeholder* dengan informasi Anda sendiri.

```
# 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()
```

Menanyakan di FLE sisi klien

Amazon DocumentDB hanya mendukung kueri kesetaraan dengan nilai terenkripsi FLE sisi klien.

Misalnya, untuk menanyakan dokumen di mana gamerscore terenkripsi sama dengan 500 klien menggunakan metode enkripsi eksplisit untuk mengenkripsi nilai kueri:

```
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 }
} )
```

Mengenkripsi data Amazon DocumentDB saat istirahat

> **Note**
>
> AWS KMS mengganti istilah customer master key (CMK) dengan AWS KMS keydan kunci KMS. Konsepnya tidak berubah. Untuk mencegah perubahan yang melanggar, AWS KMS adalah menjaga beberapa variasi istilah ini.

Anda mengenkripsi data at rest di klaster Amazon DocumentDB dengan menentukan opsi enkripsi penyimpanan saat Anda membuat klaster. Enkripsi penyimpanan diaktifkan di seluruh klaster dan diterapkan ke semua instans, termasuk instans primer dan replika apa pun. Hal ini juga diterapkan pada volume penyimpanan, data, indeks, log, backup otomatis, dan snapshot klaster Anda.

Amazon DocumentDB menggunakan Advanced Encryption Standard AES-256 () 256-bit untuk mengenkripsi data Anda menggunakan kunci enkripsi yang disimpan di (). AWS Key Management Service AWS KMS Saat menggunakan klaster Amazon DocumentDB dengan enkripsi saat istirahat diaktifkan, Anda tidak perlu mengubah logika aplikasi atau koneksi klien Anda. Amazon DocumentDB menangani enkripsi dan dekripsi data Anda secara transparan, dengan dampak minimal terhadap performa.

Amazon DocumentDB terintegrasi AWS KMS dengan dan menggunakan metode yang dikenal sebagai enkripsi amplop untuk melindungi data Anda. Ketika cluster Amazon DocumentDB dienkripsi dengan, AWS KMS Amazon DocumentDB AWS KMS meminta untuk menggunakan kunci KMS Anda untuk menghasilkan kunci data ciphertext untuk mengenkripsi volume penyimpanan. Kunci data ciphertext dienkripsi menggunakan kunci KMS yang Anda tentukan, dan disimpan bersama dengan data terenkripsi dan metadata penyimpanan. Saat Amazon DocumentDB perlu mengakses data terenkripsi Anda, Amazon DocumentDB AWS KMS meminta untuk mendekripsi kunci data ciphertext menggunakan kunci KMS Anda dan menyimpan kunci data teks biasa di memori untuk mengenkripsi dan mendekripsi data secara efisien dalam volume penyimpanan.

Fasilitas enkripsi penyimpanan di Amazon DocumentDB tersedia untuk semua ukuran instans yang didukung dan di semua tempat Wilayah AWS Amazon DocumentDB tersedia.

Mengaktifkan enkripsi saat istirahat untuk cluster Amazon DocumentDB

Anda dapat mengaktifkan atau menonaktifkan enkripsi saat istirahat di klaster Amazon DocumentDB saat klaster disediakan menggunakan atau Konsol Manajemen AWS (). AWS Command Line Interface AWS CLI Klaster yang Anda buat menggunakan konsol memiliki enkripsi saat istirahat yang diaktifkan secara default. Cluster yang Anda buat menggunakan enkripsi AWS CLI have at rest dinonaktifkan secara default. Oleh karena itu, Anda harus secara eksplisit mengaktifkan enkripsi saat istirahat menggunakan parameter `--storage-encrypted`. Dalam kedua kasus tersebut, setelah klaster dibuat, Anda tidak dapat mengubah opsi enkripsi saat istirahat.

Amazon DocumentDB AWS KMS menggunakan untuk mengambil dan mengelola kunci enkripsi, dan untuk menentukan kebijakan yang mengontrol bagaimana kunci ini dapat digunakan. Jika Anda tidak menentukan pengenal AWS KMS kunci, Amazon DocumentDB menggunakan kunci KMS layanan AWS terkelola default. Amazon DocumentDB membuat kunci KMS terpisah untuk masing-masing di Anda. Wilayah AWS Akun AWS Untuk informasi selengkapnya, lihatAWS Key Management Service Konsep.

Untuk memulai membuat kunci KMS Anda sendiri, lihat Memulai di Panduan AWS Key Management Service Pengembang.

> **Important**
>
> Anda harus menggunakan kunci KMS enkripsi simetris untuk mengenkripsi klaster Anda karena Amazon DocumentDB hanya mendukung kunci KMS enkripsi simetris. Jangan gunakan kunci KMS asimetris untuk mencoba mengenkripsi data di cluster Amazon DocumentDB Anda. Untuk informasi selengkapnya, lihat Kunci asimetris AWS KMS di Panduan AWS Key Management Service Pengembang.

Jika Amazon DocumentDB tidak dapat lagi mengakses kunci KMS untuk klaster — misalnya, ketika Akun AWS kunci yang memiliki ditangguhkan, kunci dinonaktifkan, kunci dijadwalkan untuk dihapus, atau kebijakan kunci atau pemberian yang diandalkan Amazon DocumentDB dihapus — klaster pertama kali beralih ke status. `inaccessible-encryption-credentials-recoverable` Saat klaster berada dalam status ini, Amazon DocumentDB menghentikan instance cluster dan Anda tidak dapat membaca dari atau menulis ke cluster, tetapi cluster masih dapat dipulihkan jika akses ke kunci KMS dipulihkan dalam 7 hari. Jika akses tidak dipulihkan dalam 7 hari, klaster akan beralih ke `inaccessible-encryption-credentials` status terminal. Dari status terminal, klaster tidak lagi tersedia dan status database saat ini tidak dapat dipulihkan — Anda hanya dapat memulihkan

dari cadangan atau melakukan pemulihan point-in-time menggunakan kunci KMS asli. Untuk Amazon DocumentDB, backup selalu diaktifkan setidaknya selama 1 hari.

> **Note**
>
> Cluster yang merupakan bagian dari cluster global berperilaku berbeda. Ketika Amazon DocumentDB mendeteksi bahwa Amazon DocumentDB tidak dapat lagi mengakses kunci KMS, semua cluster dalam klaster global bertransisi langsung ke status `inaccessible-encryption-credentials` terminal, melewatkan status yang dapat dipulihkan. Ini karena cluster yang merupakan bagian dari cluster global dapat dihentikan dan dimulai hanya ketika itu adalah satu-satunya cluster di cluster global. Untuk memulihkan, Anda harus memulihkan dari snapshot atau melakukan pemulihan point-in-time. Untuk menghapus cluster asli, Anda harus terlebih dahulu menghapus setiap cluster dari cluster global dan kemudian menghapusnya.

> **Important**
>
> Anda tidak dapat mengubah kunci KMS untuk cluster terenkripsi setelah Anda membuatnya. Pastikan untuk menentukan persyaratan kunci enkripsi Anda sebelum membuat klaster terenkripsi Anda.

Using the Konsol Manajemen AWS

Anda menentukan opsi enkripsi saat istirahat saat membuat klaster. Enkripsi saat istirahat diaktifkan secara default saat Anda membuat klaster menggunakan Konsol Manajemen AWS. Itu tidak dapat diubah setelah klaster dibuat.

Untuk menentukan opsi enkripsi saat istirahat saat membuat klaster Anda

1. Buat klaster Amazon DocumentDB seperti yang dijelaskan di bagian Memulai. Namun, pada langkah 6, jangan pilih Buat klaster.
2. Di bawah bagian Autentikasi, pilih Tampilkan pengaturan lanjutan.
3. Gulir ke bawah ke Encryption-at-restbagian.
4. Pilih opsi yang Anda inginkan untuk enkripsi saat istirahat. Opsi mana pun yang Anda pilih, Anda tidak dapat mengubahnya setelah klaster dibuat.
   - Untuk mengenkripsi data at rest di klaster ini, pilih Aktifkan enkripsi.

- Jika Anda tidak ingin mengenkripsi data at rest di klaster ini, pilih Nonaktifkan enkripsi.

5. Pilih kunci utama yang Anda inginkan. Amazon DocumentDB menggunakan AWS Key Management Service AWS KMS() untuk mengambil dan mengelola kunci enkripsi, dan untuk menentukan kebijakan yang mengontrol bagaimana kunci ini dapat digunakan. Jika Anda tidak menentukan pengenal AWS KMS kunci, Amazon DocumentDB menggunakan kunci KMS layanan AWS terkelola default. Untuk informasi selengkapnya, lihatAWS Key Management Service Konsep.

   > **Note**
   >
   > Setelah Anda membuat cluster terenkripsi, Anda tidak dapat mengubah kunci KMS untuk cluster itu. Pastikan untuk menentukan persyaratan kunci enkripsi Anda sebelum membuat klaster terenkripsi Anda.

6. Lengkapi bagian lain yang diperlukan, dan buat klaster Anda.

Using the AWS CLI

Untuk mengenkripsi cluster Amazon DocumentDB AWS CLI menggunakan, `create-db-cluster`jalankan perintah dan tentukan opsi. `--storage-encrypted` Cluster Amazon DocumentDB dibuat menggunakan AWS CLI enkripsi jangan aktifkan penyimpanan secara default.

Contoh berikut membuat klaster Amazon DocumentDB dengan enkripsi penyimpanan yang diaktifkan.

Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi cluster Anda.

Example

Untuk Linux, macOS, atau Unix:

```
aws docdb create-db-cluster \
  --db-cluster-identifier mydocdbcluster \
  --port 27017 \
  --engine docdb \
  --master-username SampleUser1 \
  --master-user-password primaryPassword \
```

```
  --storage-encrypted
```

Untuk Windows:

```
aws docdb create-db-cluster ^
  --db-cluster-identifier SampleUser1 ^
  --port 27017 ^
  --engine docdb ^
  --master-username SampleUser1 ^
  --master-user-password primaryPassword ^
  --storage-encrypted
```

Saat membuat kluster Amazon DocumentDB terenkripsi, Anda dapat menentukan pengenal kunci, seperti pada AWS KMS contoh berikut.

Example

Untuk Linux, macOS, atau 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
```

Untuk 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**
>
> Setelah Anda membuat cluster terenkripsi, Anda tidak dapat mengubah kunci KMS untuk cluster itu. Pastikan untuk menentukan persyaratan kunci enkripsi Anda sebelum membuat klaster terenkripsi Anda.

Menyelesaikan cluster Amazon DocumentDB dalam status enkripsi yang tidak dapat diakses

Cluster Amazon DocumentDB pindah ke status enkripsi yang tidak dapat diakses saat Amazon DocumentDB tidak dapat mengakses kunci KMS yang dienkripsi oleh cluster. Ada dua status seperti itu, dan jalur pemulihan tergantung pada mana cluster berada.

`status yang tidak dapat diakses-enkripsi-kredensial-dapat` dipulihkan

Saat klaster dalam `inaccessible-encryption-credentials-recoverable` status, Anda dapat mengembalikan cluster `available` dengan memulihkan akses Amazon DocumentDB ke kunci KMS dan kemudian memulai cluster. Untuk mengatasi status ini, lakukan hal berikut:

1. Konfirmasikan Akun AWS bahwa yang memiliki kunci KMS aktif. Jika akun ditangguhkan, aktifkan kembali.
2. Konfirmasikan bahwa kunci KMS diaktifkan. Untuk informasi selengkapnya, lihat Mengaktifkan dan menonaktifkan kunci di Panduan Developer AWS Key Management Service .
3. Periksa apakah kunci KMS dijadwalkan untuk dihapus. Jika ya, batalkan penghapusan kunci yang dijadwalkan. Untuk informasi selengkapnya, lihat Menjadwalkan dan membatalkan penghapusan kunci di Panduan Pengembang.AWS Key Management Service
4. Konfirmasikan bahwa kebijakan kunci KMS dan hibah apa pun yang diandalkan Amazon DocumentDB masih mengizinkan Amazon DocumentDB untuk menggunakan kunci tersebut.
5. Setelah akses ke kunci KMS dipulihkan, mulai cluster dengan menggunakan Konsol Manajemen AWS atau dengan menjalankan `start-db-cluster` AWS CLI perintah.

   Example

   Untuk Linux, macOS, atau Unix:

   ```
   aws docdb start-db-cluster \
     --db-cluster-identifier example-cluster
   ```

Untuk Windows:

```
aws docdb start-db-cluster ^
  --db-cluster-identifier example-cluster
```

> **Important**
>
> Jika akses ke kunci KMS tidak dipulihkan dalam 7 hari, klaster akan beralih ke `inaccessible-encryption-credentials` status terminal, dari mana ia tidak dapat dimulai.

status **`kredensial-enkripsi yang tidak dapat diakses`**

`inaccessible-encryption-credentials`Statusnya adalah terminal. Cluster tidak dapat dimulai, dan status menjalankan database tidak dapat dipulihkan. Untuk memulihkan data Anda, pulihkan dari snapshot atau lakukan pemulihan point-in-time ke cluster baru. Anda masih harus memiliki akses ke kunci KMS asli untuk melakukan pemulihan. Jika kunci KMS dihapus, data tidak dapat dipulihkan.

Untuk informasi selengkapnya, lihat Memulihkan dari snapshot cluster dan Memulihkan ke titik waktu.

> **Note**
>
> Cluster yang merupakan bagian dari transisi klaster global langsung ke `inaccessible-encryption-credentials` status ketika akses ke kunci KMS hilang, karena cluster yang merupakan bagian dari cluster global dapat dihentikan dan dimulai hanya ketika itu adalah satu-satunya cluster di cluster global. Untuk menghapus klaster yang berada dalam status ini, pertama-tama hapus setiap cluster dari cluster global, lalu hapus kluster satu per satu.

Jika Anda tidak dapat menghapus klaster yang berada dalam `inaccessible-encryption-credentials` status karena perlindungan penghapusan diaktifkan, matikan perlindungan penghapusan dengan menggunakan AWS CLI sebelum mencoba ulang penghapusan.

Example

Untuk Linux, macOS, atau Unix:

```
aws docdb modify-db-cluster \
  --db-cluster-identifier example-cluster \
  --no-deletion-protection
```

Untuk Windows:

```
aws docdb modify-db-cluster ^
  --db-cluster-identifier example-cluster ^
  --no-deletion-protection
```

Anda kemudian dapat menghapus cluster dengan menggunakan `delete-db-cluster` perintah.

Example

Untuk Linux, macOS, atau Unix:

```
aws docdb delete-db-cluster \
  --db-cluster-identifier example-cluster \
  --skip-final-snapshot
```

Untuk Windows:

```
aws docdb delete-db-cluster ^
  --db-cluster-identifier example-cluster ^
  --skip-final-snapshot
```

Jika cluster tidak menghapus setelah Anda menjalankan perintah sebelumnya, hubungi Support AWS .

Batasan untuk cluster terenkripsi Amazon DocumentDB

Keterbatasan berikut ada untuk klaster terenkripsi Amazon DocumentDB.

- Anda dapat mengaktifkan atau menonaktifkan enkripsi saat istirahat untuk klaster Amazon DocumentDB hanya pada saat klaster dibuat, bukan setelah klaster telah dibuat. Namun, Anda dapat membuat salinan terenkripsi dari klaster yang tidak terenkripsi dengan membuat snapshot dari klaster yang tidak terenkripsi, lalu memulihkan snapshot yang tidak terenkripsi sebagai klaster baru sambil menentukan opsi enkripsi saat istirahat.

  Untuk informasi selengkapnya, lihat topik berikut:

  - Membuat snapshot cluster manual
  - Memulihkan dari snapshot cluster
  - Menyalin snapshot cluster Amazon DocumentDB
- Klaster Amazon DocumentDB dengan enkripsi penyimpanan yang diaktifkan tidak dapat dimodifikasi untuk menonaktifkan enkripsi.
- Semua instance, backup otomatis, snapshot, dan indeks dalam cluster Amazon DocumentDB dienkripsi dengan kunci KMS yang sama.

Mengenkripsi data dalam perjalanan

Anda dapat menggunakan Keamanan Lapisan Pengangkutan (TLS) untuk mengenkripsi koneksi antara aplikasi Anda dan klaster Amazon DocumentDB. Secara default, enkripsi dalam transit diaktifkan untuk klaster Amazon DocumentDB yang baru dibuat. Itu dapat secara opsional dinonaktifkan ketika klaster dibuat, atau di lain waktu. Ketika enkripsi dalam transit diaktifkan, koneksi aman menggunakan TLS diperlukan untuk terhubung ke klaster. Untuk informasi selengkapnya menghubungkan ke Amazon DocumentDB menggunakan TLS, lihat Menghubungkan secara terprogram ke Amazon DocumentDB.

Mengelola pengaturan TLS cluster Amazon DocumentDB

Enkripsi dalam transit untuk klaster Amazon DocumentDB dikelola melalui parameter TLS dalam grup parameter klaster. Anda dapat mengelola pengaturan TLS cluster Amazon DocumentDB menggunakan atau () Konsol Manajemen AWS . AWS Command Line Interface AWS CLI Lihat bagian berikut untuk mempelajari cara memverifikasi dan memodifikasi pengaturan TLS Anda saat ini.

Using the Konsol Manajemen AWS

Ikuti langkah-langkah berikut untuk melakukan tugas manajemen untuk enkripsi TLS menggunakan konsol—seperti mengidentifikasi grup parameter, memverifikasi nilai TLS, dan membuat modifikasi yang diperlukan.

> **Note**
>
> Kecuali jika Anda menentukan secara berbeda saat membuat klaster, klaster Anda dibuat dengan grup parameter klaster default. Parameter dalam grup parameter `default` cluster tidak dapat dimodifikasi (misalnya, `tls` enabled/disabled). Jadi jika klaster

Anda menggunakan grup parameter klaster `default`, Anda perlu memodifikasi klaster untuk menggunakan grup parameter klaster non-default. Pertama, Anda mungkin perlu membuat grup parameter klaster kustom. Untuk informasi selengkapnya, lihat Membuat grup parameter cluster Amazon DocumentDB.

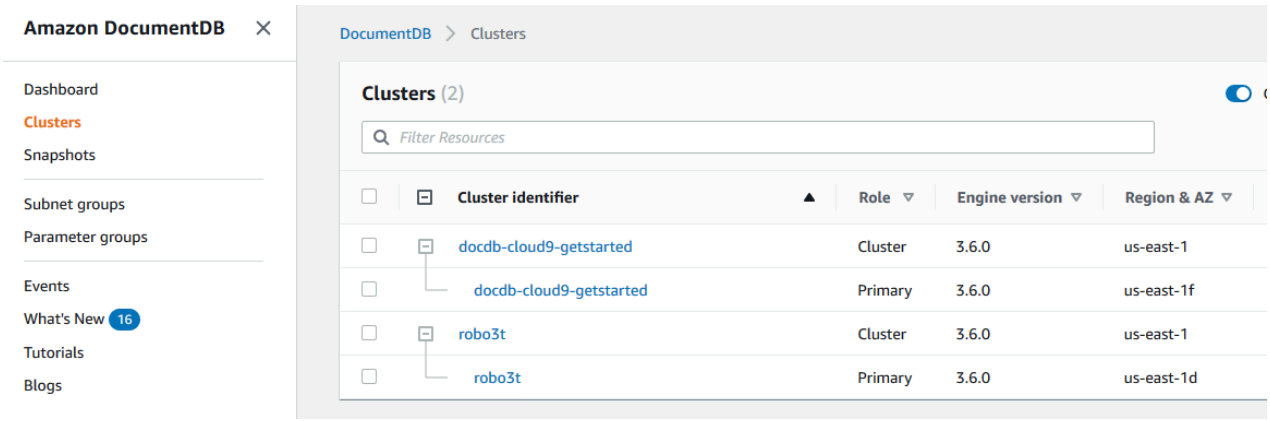
1. Tentukan grup parameter cluster yang digunakan cluster Anda.

   a. Buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb

   b. Di panel navigasi, pilih Klaster.

   > **Tip**
   >
   > Jika Anda tidak melihat panel navigasi pada sisi kiri layar Anda, pilih ikon menu (≡) di sudut kiri atas halaman.

   c. Perhatikan bahwa di kotak navigasi Clusters, kolom Cluster Identifier menunjukkan cluster dan instance. Instans terdaftar di bawah klaster. Lihat tangkapan layar berikut untuk referensi.



   d. Pilih klaster yang Anda minati.

   e. Pilih tab Konfigurasi dan gulir ke bawah ke bagian bawah detail Cluster dan temukan grup parameter Cluster. Perhatikan nama grup parameter klaster.

      Jika nama grup parameter klaster adalah `default` (misalnya, `default.docdb3.6`), Anda harus membuat grup parameter klaster kustom dan menjadikannya grup parameter klaster sebelum melanjutkan. Untuk informasi selengkapnya, lihat berikut ini:

1. Membuat grup parameter cluster Amazon DocumentDB — Jika Anda tidak memiliki grup parameter klaster kustom yang dapat Anda gunakan, buatlah.
2. Memodifikasi cluster Amazon DocumentDB — Modifikasi klaster Anda untuk menggunakan grup parameter klaster kustom.

2. Tentukan nilai parameter **`tls`** cluster saat ini.

   a. Buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb

   b. Di panel navigasi, pilih Grup parameter.

   c. Dalam daftar grup parameter klaster, pilih nama grup parameter klaster yang Anda minati.

   d. Temukan bagian Parameter klaster. Dalam daftar parameter klaster, temukan baris parameter klaster `tls`. Pada titik ini, empat kolom berikut ini penting:

      - Nama parameter klaster — Nama dari parameter klaster. Untuk mengelola TLS, Anda tertarik dengan parameter klaster `tls`.
      - Nilai — Nilai saat ini dari setiap parameter klaster.
      - Nilai yang diizinkan — Daftar nilai yang dapat diterapkan pada parameter klaster.
      - Tipe penerapan — Baik statis atau dinamis. Perubahan pada parameter klaster statis hanya dapat diterapkan saat instans di-boot ulang. Perubahan pada parameter klaster dinamis hanya dapat diterapkan baik segera atau saat instans di-boot ulang.

3. Ubah nilai parameter **`tls`** cluster.

   Jika nilai `tls` tidak sesuai dengan kebutuhan, modifikasi nilainya untuk grup parameter klaster ini. Untuk mengubah nilai parameter klaster `tls`, lanjutkan dari bagian sebelumnya dengan mengikuti langkah-langkah berikut.

   a. Pilih tombol di sebelah kiri nama parameter klaster (`tls`).

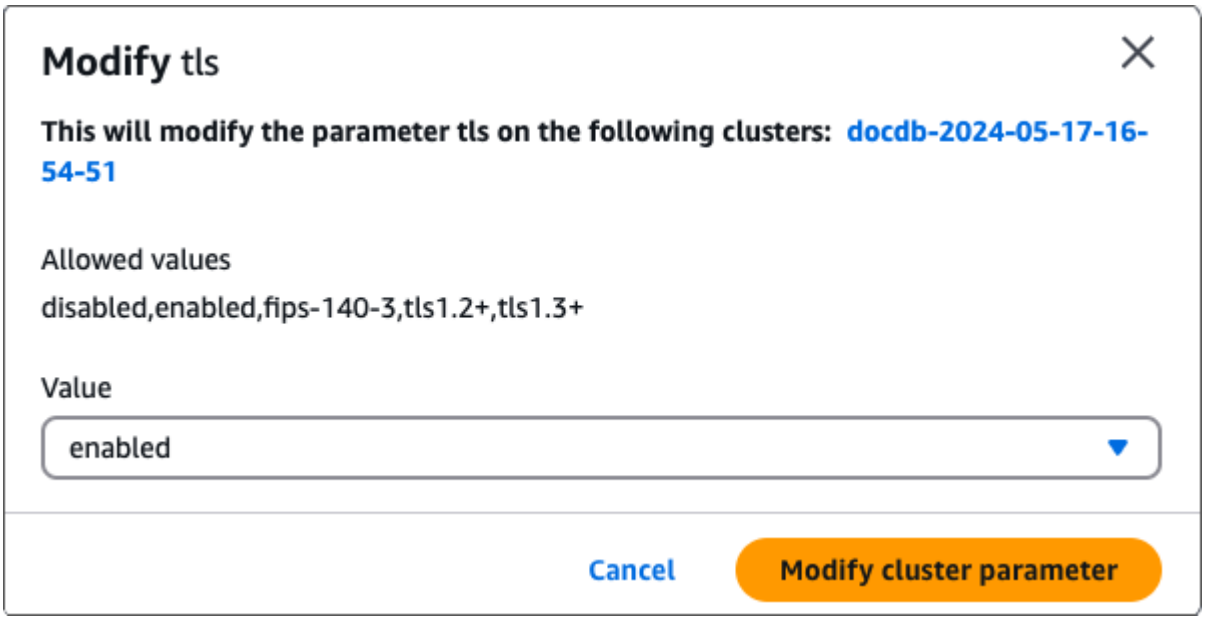
   b. Pilih Edit.

   c. Untuk mengubah nilai `tls`, di kotak dialog Modifikasi **`tls`**, pilih nilai yang Anda inginkan untuk parameter klaster dalam daftar tarik-turun.

      Nilai yang valid adalah:

      - dinonaktifkan - Menonaktifkan TLS

- diaktifkan - Mengaktifkan TLS. Untuk Amazon DocumentDB 5.0 dan yang lebih lama, ini memungkinkan TLS 1.0 hingga 1.3, tetapi kami merekomendasikan TLS 1.2 atau lebih tinggi. Hanya TLS 1.2 dan yang lebih tinggi yang didukung di Amazon DocumentDB 5.0 dimulai dengan versi minor 5.0.1, Amazon DocumentDB 8.0 dan yang lebih baru.
- fips-140-3 - Mengaktifkan TLS dengan FIPS. Cluster hanya menerima koneksi aman sesuai persyaratan publikasi Federal Information Processing Standards (FIPS) 140-3. Ini hanya didukung mulai dengan cluster Amazon DocumentDB 5.0 (versi engine 3.0.3727) di wilayah ini: ca-central-1, us-west-2, us-east-1, us-east-1, us-east-2, us-east-2, us-gov-east-1, us-gov-east-1.
- tls1.2+ - Mengaktifkan TLS versi 1.2 dan di atas. Ini hanya didukung mulai dengan Amazon DocumentDB 4.0 (versi mesin 2.0.10980) dan Amazon DocumentDB (engine versi 3.0.11051).
- tls1.3+ - Mengaktifkan TLS versi 1.3 dan di atas. Ini hanya didukung mulai dengan Amazon DocumentDB 4.0 (versi mesin 2.0.10980) dan Amazon DocumentDB (engine versi 3.0.11051).



d. Pilih Modifikasi parameter klaster. Perubahan diterapkan untuk setiap instans klaster saat di-boot ulang.

4. Nyalakan ulang instans Amazon DocumentDB.

   Boot ulang setiap instans klaster sehingga perubahan diterapkan ke semua instans di klaster.

a. Buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb

b. Di panel navigasi, pilih Instans.

c. Untuk menentukan instans yang akan di-boot ulang, temukan instans dalam daftar instans, dan pilih tombol di sebelah kiri namanya.

d. Pilih Tindakan, dan kemudian Boot ulang. Konfirmasikan bahwa Anda ingin melakukan boot ulang dengan memilih Boot ulang.

Using the AWS CLI

Ikuti langkah-langkah berikut untuk melakukan tugas manajemen untuk enkripsi TLS menggunakan AWS CLI—seperti mengidentifikasi grup parameter, memverifikasi nilai TLS, dan membuat modifikasi yang diperlukan.

> **Note**
>
> Kecuali jika Anda menentukan secara berbeda saat membuat klaster, klaster dibuat dengan grup parameter klaster default. Parameter dalam grup parameter `default` cluster tidak dapat dimodifikasi (misalnya, `tls` enabled/disabled). Jadi jika klaster Anda menggunakan grup parameter klaster `default`, Anda perlu memodifikasi klaster untuk menggunakan grup parameter klaster non-default. Pertama Anda mungkin perlu membuat grup parameter klaster kustom. Untuk informasi selengkapnya, lihat Membuat grup parameter cluster Amazon DocumentDB.

1. Tentukan grup parameter cluster yang digunakan cluster Anda.

   Jalankan `describe-db-clusters`perintah dengan opsi berikut:

   - `--db-cluster-identifier`
   - `--query`

   Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi cluster Anda.

```
aws docdb describe-db-clusters \
  --db-cluster-identifier mydocdbcluster \
```

```
  --query 'DBClusters[*].[DBClusterIdentifier,DBClusterParameterGroup]'
```

Output dari operasi ini terlihat seperti berikut (format JSON):

```
[
    [
        "mydocdbcluster",
        "myparametergroup"
    ]
]
```

Jika nama grup parameter klaster adalah `default` (misalnya, `default.docdb3.6`), Anda harus memiliki grup parameter klaster kustom dan menjadikannya grup parameter klaster sebelum melanjutkan. Untuk informasi selengkapnya, lihat topik berikut:

1. Membuat grup parameter cluster Amazon DocumentDB — Jika Anda tidak memiliki grup parameter klaster kustom yang dapat Anda gunakan, buatlah.
2. Memodifikasi cluster Amazon DocumentDB — Modifikasi klaster Anda untuk menggunakan grup parameter klaster kustom.

2. Tentukan nilai parameter **`tls`** cluster saat ini.

Untuk mendapatkan informasi lebih lanjut tentang grup parameter cluster ini, jalankan `describe-db-cluster-parameters`perintah dengan opsi berikut:

- `--db-cluster-parameter-group-name`
- `--query`

  Membatasi output hanya pada bidang yang diminati:`ParameterName`,`ParameterValue`,`AllowedValues`, dan`ApplyType`.

Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi cluster Anda.

```
aws docdb describe-db-cluster-parameters \
  --db-cluster-parameter-group-name myparametergroup \
  --query 'Parameters[*].[ParameterName,ParameterValue,AllowedValues,ApplyType]'
```

Output dari operasi ini terlihat seperti berikut (format 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. Ubah nilai parameter **`tls`** cluster.

   Jika nilai `tls` tidak sesuai dengan kebutuhan, modifikasi nilainya untuk grup parameter klaster ini. Untuk mengubah nilai parameter `tls` cluster, jalankan [modify-db-cluster-parameter-group](#)perintah dengan opsi berikut:

   - `--db-cluster-parameter-group-name` — Diperlukan. Nama grup parameter klaster yang akan dimodifikasi. Ini tidak dapat menjadi grup parameter klaster `default.*`.
   - `--parameters` — Diperlukan. Daftar parameter grup parameter klaster yang akan dimodifikasi.
     - `ParameterName` — Diperlukan. Nama parameter klaster yang akan dimodifikasi.
     - `ParameterValue` — Diperlukan. Nilai baru untuk parameter klaster ini. Harus salah satu dari `AllowedValues` parameter klaster.
       - `enabled`- Cluster menerima koneksi aman menggunakan TLS. Untuk Amazon DocumentDB 5.0 dan yang lebih lama, ini memungkinkan TLS 1.0 hingga 1.3, tetapi kami merekomendasikan TLS 1.2 atau lebih tinggi. Hanya TLS 1.2 dan yang lebih tinggi yang didukung di Amazon DocumentDB 5.0 dimulai dengan versi minor 5.0.1, Amazon DocumentDB 8.0 dan yang lebih baru.

  - `disabled` — Klaster tidak menerima koneksi aman menggunakan TLS.
  - `fips-140-3`Cluster hanya menerima koneksi aman sesuai persyaratan publikasi Federal Information Processing Standards (FIPS) 140-3. Ini hanya didukung mulai dengan cluster Amazon DocumentDB 5.0 (versi engine 3.0.3727) di wilayah ini: ca-central-1, us-west-2, us-east-1, us-east-1, us-east-2, us-east-2, us-gov-east-1, us-gov-east-1.
  - `tls1.2+`- Cluster menerima koneksi aman menggunakan TLS versi 1.2 dan di atasnya. Ini hanya didukung mulai dengan Amazon DocumentDB 4.0 (versi mesin 2.0.10980) dan Amazon DocumentDB 5.0 (versi mesin 3.0.11051).
  - `tls1.3+`- Cluster menerima koneksi aman menggunakan TLS versi 1.3 ke atas. Ini hanya didukung mulai dengan Amazon DocumentDB 4.0 (versi mesin 2.0.10980) dan Amazon DocumentDB 5.0 (versi mesin 3.0.11051).
- `ApplyMethod` — Kapan modifikasi ini diterapkan. Untuk parameter klaster statis seperti `tle`, nilai ini harus berupa `pending-reboot`.
  - `pending-reboot` — Perubahan diterapkan ke sebuah instans hanya setelah di-boot ulang. Anda harus mem-boot ulang setiap instans klaster satu per satu agar perubahan ini terjadi di semua instans klaster.

Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi cluster Anda.

Kode berikut dinonaktifkan`tls`, menerapkan perubahan ke setiap instance saat di-boot ulang.

```
aws docdb modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=tls,ParameterValue=disabled,ApplyMethod=pending-
reboot"
```

Kode berikut memungkinkan`tls`, menerapkan perubahan untuk setiap instance ketika reboot.

```
aws docdb modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=tls,ParameterValue=enabled,ApplyMethod=pending-
reboot"
```

Kode berikut memungkinkan TLS dengan`fips-140-3`, menerapkan perubahan untuk setiap instance ketika reboot.

```
aws docdb modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup2 \
  --parameters "ParameterName=tls,ParameterValue=fips-140-3,ApplyMethod=pending-
reboot"
```

Output dari operasi ini terlihat seperti berikut (format JSON):

```
{
    "DBClusterParameterGroupName": "myparametergroup"
}
```

4. Nyalakan ulang instans Amazon DocumentDB Anda.

   Boot ulang setiap instans klaster sehingga perubahan diterapkan ke semua instans di klaster. Untuk me-reboot instance Amazon DocumentDB, jalankan perintah dengan [reboot-db-instance](#)opsi berikut:

   - `--db-instance-identifier`

   Kode berikut mem-boot ulang instans `mydocdbinstance`.

   Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi cluster Anda.

   Example

   Untuk Linux, macOS, atau Unix:

   ```
   aws docdb reboot-db-instance \
     --db-instance-identifier mydocdbinstance
   ```

   Untuk Windows:

   ```
   aws docdb reboot-db-instance ^
     --db-instance-identifier mydocdbinstance
   ```

Output dari operasi ini terlihat seperti berikut (format 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"
```

```
    }
}
```

Diperlukan waktu beberapa menit agar instans di-boot ulang. Anda dapat menggunakan instans hanya ketika statusnya tersedia. Anda dapat memantau status instans menggunakan konsol atau AWS CLI. Untuk informasi selengkapnya, lihat Memantau status instans Amazon DocumentDB.

Manajemen kunci

Amazon DocumentDB AWS Key Management Service menggunakan AWS KMS() untuk mengambil dan mengelola kunci enkripsi. AWS KMS menggabungkan perangkat keras dan perangkat lunak yang aman dan sangat tersedia untuk menyediakan sistem manajemen kunci yang diskalakan untuk cloud. Dengan menggunakan AWS KMS, Anda dapat membuat kunci enkripsi dan menentukan kebijakan yang mengontrol bagaimana kunci ini dapat digunakan. AWS KMS mendukung AWS CloudTrail, sehingga Anda dapat mengaudit penggunaan kunci untuk memverifikasi bahwa kunci sedang digunakan dengan tepat.

AWS KMS Kunci Anda dapat digunakan dalam kombinasi dengan Amazon DocumentDB dan layanan yang AWS didukung seperti Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3), Amazon Relational Database Service (Amazon RDS), Amazon Elastic Block Store (Amazon EBS), dan Amazon Redshift. Untuk daftar layanan yang mendukung AWS KMS, lihat Cara AWS Layanan digunakan AWS KMS dalam Panduan AWS Key Management Service Pengembang. Untuk informasi tentang AWS KMS, lihat Apa itu AWS Key Management Service?

Identity and Access Management untuk Amazon DocumentDB

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. Administrator IAM mengontrol siapa yang dapat diautentikasi (masuk) dan diotorisasi (memiliki izin) untuk menggunakan sumber daya Amazon DocumentDB. IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

- Audiens
- Mengautentikasi dengan identitas

- Mengelola akses menggunakan kebijakan
- Bagaimana Amazon DocumentDB bekerja dengan IAM
- Identity-based contoh kebijakan untuk Amazon DocumentDB
- Memecahkan masalah identitas dan akses Amazon DocumentDB
- Mengelola izin akses ke sumber daya Amazon DocumentDB Anda
- Menggunakan kebijakan berbasis identitas (kebijakan IAM) untuk Amazon DocumentDB
- AWS kebijakan terkelola untuk Amazon DocumentDB
- Izin API Amazon DocumentDB: referensi tindakan, sumber daya, dan kondisi

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda berdasarkan peran Anda:

- Pengguna layanan - minta izin dari administrator Anda jika Anda tidak dapat mengakses fitur (lihat Memecahkan masalah identitas dan akses Amazon DocumentDB)
- Administrator layanan - tentukan akses pengguna dan mengirimkan permintaan izin (lihat Bagaimana Amazon DocumentDB bekerja dengan IAM)
- Administrator IAM - tulis kebijakan untuk mengelola akses (lihat Identity-based contoh kebijakan untuk Amazon DocumentDB)

Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensyal identitas Anda. Anda harus diautentikasi sebagai Pengguna root akun AWS, pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk sebagai identitas federasi menggunakan kredensyal dari sumber identitas seperti AWS IAM Identity Center (Pusat Identitas IAM), autentikasi masuk tunggal, atau kredensyal. Google/ Facebook Untuk informasi selengkapnya tentang cara masuk, lihat Cara masuk ke Akun AWS Anda dalam Panduan Pengguna AWS Sign-In .

Untuk akses terprogram, AWS sediakan SDK dan CLI untuk menandatangani permintaan secara kriptografis. Untuk informasi selengkapnya, lihat AWS Signature Version 4 untuk permintaan API dalam Panduan Pengguna IAM.

Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang disebut pengguna Akun AWS root yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Untuk tugas yang memerlukan kredensial pengguna root, lihat Tugas yang memerlukan kredensial pengguna root dalam Panduan Pengguna IAM.

Identitas terfederasi

Sebagai praktik terbaik, mewajibkan pengguna manusia untuk menggunakan federasi dengan penyedia identitas untuk mengakses Layanan AWS menggunakan kredensyal sementara.

Identitas federasi adalah pengguna dari direktori perusahaan Anda, penyedia identitas web, atau Directory Service yang mengakses Layanan AWS menggunakan kredensyal dari sumber identitas. Identitas terfederasi mengambil peran yang memberikan kredensial sementara.

Untuk manajemen akses terpusat, kami menyarankan AWS IAM Identity Center. Untuk informasi selengkapnya, lihat Apa itu Pusat Identitas IAM? dalam Panduan Pengguna AWS IAM Identity Center .

Pengguna dan grup IAM

Pengguna IAM adalah identitas dengan izin khusus untuk satu orang atau aplikasi. Sebaiknya gunakan kredensial sementara alih-alih pengguna IAM dengan kredensial jangka panjang. Untuk informasi selengkapnya, lihat Mewajibkan pengguna manusia untuk menggunakan federasi dengan penyedia identitas untuk mengakses AWS menggunakan kredensyal sementara di Panduan Pengguna IAM.

Grup IAM menentukan kumpulan pengguna IAM dan mempermudah pengelolaan izin untuk pengguna dalam jumlah besar. Untuk mempelajari selengkapnya, lihat Kasus penggunaan untuk pengguna IAM dalam Panduan Pengguna IAM.

Peran IAM

Peran IAM adalah identitas dengan izin khusus yang menyediakan kredensial sementara. Anda dapat mengambil peran dengan beralih dari pengguna ke peran IAM (konsol) atau dengan memanggil operasi AWS CLI atau AWS API. Untuk informasi selengkapnya, lihat Metode untuk mengambil peran dalam Panduan Pengguna IAM.

Peran IAM berguna untuk akses pengguna terfederasi, izin pengguna IAM sementara, akses lintas akun, akses lintas layanan, dan aplikasi yang berjalan di Amazon EC2. Untuk informasi selengkapnya, lihat Akses sumber daya lintas akun di IAM dalam Panduan Pengguna IAM.

Mengelola akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan menentukan izin saat dikaitkan dengan identitas atau sumber daya. AWS mengevaluasi kebijakan ini ketika kepala sekolah membuat permintaan. Sebagian besar kebijakan disimpan AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang dokumen kebijakan JSON, lihat Gambaran umum kebijakan JSON dalam Panduan Pengguna IAM.

Menggunakan kebijakan, administrator menentukan siapa yang memiliki akses ke apa dengan mendefinisikan principal mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Administrator IAM membuat kebijakan IAM dan menambahkannya ke peran, yang kemudian dapat diambil oleh pengguna. Kebijakan IAM mendefinisikan izin terlepas dari metode yang Anda gunakan untuk melakukan operasinya.

Identity-based kebijakan

Identity-based kebijakan adalah dokumen kebijakan izin JSON yang Anda lampirkan ke identitas (pengguna, grup, atau peran). Kebijakan ini mengontrol tindakan apa yang bisa dilakukan oleh identitas tersebut, terhadap sumber daya yang mana, dan dalam kondisi apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Tentukan izin IAM kustom dengan kebijakan yang dikelola pelanggan dalam Panduan Pengguna IAM.

Identity-based kebijakan dapat berupa kebijakan inline (disematkan langsung ke dalam satu identitas) atau kebijakan terkelola (kebijakan mandiri yang dilampirkan pada beberapa identitas). Untuk mempelajari cara memilih antara kebijakan terkelola dan kebijakan inline, lihat Pilih antara kebijakan terkelola dan kebijakan inline dalam Panduan Pengguna IAM.

Resource-based kebijakan

Resource-based kebijakan adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contohnya termasuk kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Anda harus menentukan principal dalam kebijakan berbasis sumber daya.

Resource-based kebijakan adalah kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan AWS terkelola dari IAM dalam kebijakan berbasis sumber daya.

Jenis-jenis kebijakan lain

AWS mendukung jenis kebijakan tambahan yang dapat menetapkan izin maksimum yang diberikan oleh jenis kebijakan yang lebih umum:

- Batasan izin – Menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas IAM. Untuk informasi selengkapnya, lihat Batasan izin untuk entitas IAM dalam Panduan Pengguna IAM.
- Kebijakan kontrol layanan (SCP) – Menentukan izin maksimum untuk organisasi atau unit organisasi di AWS Organizations. Untuk informasi selengkapnya, lihat Kebijakan kontrol layanan dalam Panduan Pengguna AWS Organizations .
- Kebijakan kontrol sumber daya (RCP) – Menetapkan izin maksimum yang tersedia untuk sumber daya di akun Anda. Untuk informasi selengkapnya, lihat Kebijakan kontrol sumber daya (RCP) dalam Panduan Pengguna AWS Organizations .
- Kebijakan sesi – Kebijakan lanjutan yang diteruskan sebagai parameter saat membuat sesi sementara untuk peran atau pengguna terfederasi. Untuk informasi selengkapnya, lihat Kebijakan sesi dalam Panduan Pengguna IAM.

Berbagai jenis kebijakan

Ketika beberapa jenis kebijakan berlaku pada suatu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan saat beberapa jenis kebijakan terlibat, lihat Logika evaluasi kebijakan di Panduan Pengguna IAM.

Bagaimana Amazon DocumentDB bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses ke Amazon DocumentDB, pelajari fitur IAM apa yang tersedia untuk digunakan dengan Amazon DocumentDB.

Fitur IAM yang dapat Anda gunakan dengan Amazon DocumentDB

Fitur IAM	Instance-based kluster	Cluster elastis
Identity-based kebijakan	Ya	Ya

Fitur IAM	Instance-based kluster	Cluster elastis
Resource-based kebijakan	Tidak	Tidak
Tindakan kebijakan	Ya	Ya
Sumber daya kebijakan	Ya	Ya
kunci-kunci persyaratan kebijakan (spesifik layanan)	Ya	Ya
ACL	Tidak	Tidak
ABAC (tanda dalam kebijakan)	Parsial	Ya
Kredensial sementara	Ya	Ya
Izin principal	Ya	Ya
Peran layanan	Ya	Ya
Service-linked peran	Tidak	Ya

Untuk mendapatkan tampilan tingkat tinggi tentang cara kerja Amazon DocumentDB AWS dan layanan lainnya dengan sebagian besar fitur IAM AWS , lihat layanan yang bekerja dengan IAM di Panduan Pengguna IAM.

Identity-based kebijakan untuk Amazon DocumentDB

Mendukung kebijakan berbasis identitas: Ya

Identity-based kebijakan adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke identitas, seperti pengguna IAM, grup pengguna, atau peran. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan berdasarkan kondisi seperti apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Tentukan izin IAM kustom dengan kebijakan terkelola pelanggan dalam Panduan Pengguna IAM.

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan secara spesifik apakah tindakan dan sumber daya diizinkan atau ditolak, serta kondisi yang menjadi dasar dikabulkan atau ditolaknya

tindakan tersebut. Untuk mempelajari semua elemen yang dapat Anda gunakan dalam kebijakan JSON, lihat Referensi elemen kebijakan JSON IAM dalam Panduan Pengguna IAM.

Identity-based contoh kebijakan untuk Amazon DocumentDB

Untuk melihat contoh kebijakan berbasis identitas Amazon DocumentDB, lihat. Identity-based contoh kebijakan untuk Amazon DocumentDB

Resource-based kebijakan dalam Amazon DocumentDB

Mendukung kebijakan berbasis sumber daya: Tidak

Resource-based kebijakan adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh principal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus menentukan principal dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

Untuk mengaktifkan akses lintas akun, Anda dapat menentukan secara spesifik seluruh akun atau entitas IAM di akun lain sebagai principal dalam kebijakan berbasis sumber daya. Untuk informasi selengkapnya, lihat Akses sumber daya lintas akun di IAM dalam Panduan Pengguna IAM.

Tindakan kebijakan untuk Amazon DocumentDB

Mendukung tindakan kebijakan: Ya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, di mana utama dapat melakukan tindakan pada sumber daya, dan dalam kondisi apa.

Elemen `Action` dari kebijakan JSON menjelaskan tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan. Sertakan tindakan dalam kebijakan untuk memberikan izin untuk melakukan operasi terkait.

> **Note**
>
> Untuk fitur pengelolaan tertentu, Amazon DocumentDB menggunakan teknologi operasional yang dibagi dengan Amazon Relational Database Service (Amazon RDS).
> Untuk melihat daftar tindakan RDS, lihat Tindakan yang ditentukan oleh Amazon Relational Database Service di Referensi Otorisasi Layanan.
> Untuk melihat tindakan kebijakan untuk klaster elastis Amazon DocumentDB, lihat Tindakan yang ditentukan oleh klaster elastis Amazon DocumentDB di Referensi Otorisasi Layanan.

Tindakan kebijakan di Amazon DocumentDB menggunakan awalan berikut sebelum tindakan:

```
aws
```

Untuk menetapkan secara spesifik beberapa tindakan dalam satu pernyataan, pisahkan tindakan tersebut dengan koma.

```
"Action": [
      "aws:action1",
      "aws:action2"
         ]
```

Untuk melihat contoh kebijakan berbasis identitas Amazon DocumentDB, lihat. Identity-based contoh kebijakan untuk Amazon DocumentDB

Sumber daya kebijakan untuk Amazon DocumentDB

Mendukung sumber daya kebijakan: Ya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, di mana utama dapat melakukan tindakan pada sumber daya, dan dalam kondisi apa.

Elemen kebijakan JSON `Resource` menentukan objek yang menjadi target penerapan tindakan. Praktik terbaiknya, tentukan sumber daya menggunakan Amazon Resource Name (ARN). Untuk tindakan yang tidak mendukung izin di tingkat sumber daya, gunakan wildcard (*) untuk menunjukkan bahwa pernyataan tersebut berlaku untuk semua sumber daya.

```
"Resource": "*"
```

> **Note**
>
> Untuk fitur pengelolaan tertentu, Amazon DocumentDB menggunakan teknologi operasional yang dibagi dengan Amazon Relational Database Service (Amazon RDS).
> Untuk melihat daftar jenis sumber daya RDS dan ARNnya, lihat Sumber daya yang ditentukan oleh Amazon Relational Database Service di Referensi Otorisasi Layanan.
> Untuk mempelajari tindakan yang dapat Anda tentukan ARN dari setiap sumber daya, lihat Tindakan yang ditentukan oleh Amazon Relational Database Service.
> Untuk melihat jenis sumber daya untuk klaster eleastic Amazon DocumentDB, lihat Jenis sumber daya yang ditentukan oleh klaster elastis Amazon DocumentDB dalam Referensi Otorisasi Layanan.

Untuk melihat contoh kebijakan berbasis identitas Amazon DocumentDB, lihat. Identity-based contoh kebijakan untuk Amazon DocumentDB

Kunci kondisi kebijakan untuk Amazon DocumentDB

Mendukung kunci kondisi kebijakan khusus layanan: Yes

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, principal dapat melakukan tindakan pada suatu sumber daya, dan dalam suatu syarat.

Elemen `Condition` menentukan ketika pernyataan dieksekusi berdasarkan kriteria yang ditetapkan. Anda dapat membuat ekspresi bersyarat yang menggunakan operator kondisi, misalnya sama dengan atau kurang dari, untuk mencocokkan kondisi dalam kebijakan dengan nilai-nilai yang diminta. Untuk melihat semua kunci kondisi AWS global, lihat kunci konteks kondisi AWS global di Panduan Pengguna IAM.

> **Note**
>
> Untuk fitur pengelolaan tertentu, Amazon DocumentDB menggunakan teknologi operasional yang dibagi dengan Amazon Relational Database Service (Amazon RDS).

Untuk melihat daftar kunci kondisi RDS, lihat Kunci kondisi untuk Amazon Relational Database Service di Referensi Otorisasi Layanan. Untuk mempelajari tindakan dan sumber daya yang dapat Anda gunakan kunci kondisi, lihat Tindakan yang ditentukan oleh Amazon Relational Database Service.
Untuk melihat kunci kondisi untuk klaster eleastic Amazon DocumentDB, lihat Kunci kondisi untuk klaster elastis Amazon DocumentDB di Referensi Otorisasi Layanan.

Untuk melihat contoh kebijakan berbasis identitas Amazon DocumentDB, lihat. Identity-based contoh kebijakan untuk Amazon DocumentDB

ACL di Amazon DocumentDB

Mendukung ACL: Tidak

Daftar kontrol akses (ACL) mengendalikan principal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL serupa dengan kebijakan berbasis sumber daya, meskipun kebijakan tersebut tidak menggunakan format dokumen kebijakan JSON.

ABAC dengan Amazon DocumentDB

Note

ABAC hanya didukung sebagian untuk cluster berbasis instance tetapi didukung untuk cluster elastis.

Attribute-based Access Control (ABAC) adalah strategi otorisasi yang mendefinisikan izin berdasarkan atribut yang disebut tag. Anda dapat melampirkan tag ke entitas dan AWS sumber daya IAM, lalu merancang kebijakan ABAC untuk mengizinkan operasi saat tag prinsipal cocok dengan tag pada sumber daya.

Untuk mengendalikan akses berdasarkan tanda, berikan informasi tentang tanda di elemen kondisi dari kebijakan menggunakan kunci kondisi `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, atau `aws:TagKeys`.

Jika sebuah layanan mendukung ketiga kunci kondisi untuk setiap jenis sumber daya, nilainya adalah Ya untuk layanan tersebut. Jika suatu layanan mendukung ketiga kunci kondisi untuk hanya beberapa jenis sumber daya, nilainya adalah Parsial.

Untuk informasi selengkapnya tentang ABAC, lihat Tentukan izin dengan otorisasi ABAC dalam Panduan Pengguna IAM. Untuk melihat tutorial yang menguraikan langkah-langkah pengaturan ABAC, lihat Menggunakan kontrol akses berbasis atribut (ABAC) dalam Panduan Pengguna IAM.

Menggunakan kredensyal sementara dengan Amazon DocumentDB

Mendukung kredensial sementara: Ya

Kredensi sementara menyediakan akses jangka pendek ke AWS sumber daya dan secara otomatis dibuat saat Anda menggunakan federasi atau beralih peran. AWS merekomendasikan agar Anda menghasilkan kredensyal sementara secara dinamis alih-alih menggunakan kunci akses jangka panjang. Untuk informasi selengkapnya, lihat Kredensial keamanan sementara di IAM dan Layanan AWS yang berfungsi dengan IAM dalam Panduan Pengguna IAM.

Cross-service izin utama untuk Amazon DocumentDB

Mendukung sesi akses terusan (FAS): Ya

Sesi akses teruskan (FAS) menggunakan izin pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat Sesi akses terusan.

Peran layanan untuk Amazon DocumentDB

Mendukung peran layanan: Ya

Peran layanan adalah peran IAM yang diambil oleh sebuah layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Buat sebuah peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.

> **Warning**
>
> Mengubah izin untuk peran layanan dapat merusak fungsionalitas Amazon DocumentDB. Edit peran layanan hanya jika Amazon DocumentDB memberikan panduan untuk melakukannya.

Service-linked peran untuk Amazon DocumentDB

> **Note**
>
> Service-linked peran tidak didukung untuk cluster berbasis instance tetapi didukung untuk cluster elastis.

Peran terkait layanan adalah jenis peran layanan yang ditautkan ke. Layanan AWS Layanan dapat mengambil peran untuk melakukan tindakan atas nama Anda. Service-linked peran muncul di Anda Akun AWS dan dimiliki oleh layanan. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.

Untuk detail tentang pembuatan atau manajemen peran terkait layanan, lihat Layanan AWS yang berfungsi dengan IAM. Temukan layanan dalam tabel yang `Yes` menyertakan kolom Service-linked peran. Pilih tautan Ya untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Identity-based contoh kebijakan untuk Amazon DocumentDB

Secara default, pengguna dan peran tidak memiliki izin untuk membuat atau memodifikasi sumber daya Amazon DocumentDB. Untuk memberikan izin kepada pengguna untuk melakukan tindakan di sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM.

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM menggunakan contoh dokumen kebijakan JSON ini, lihat Membuat kebijakan IAM (konsol) di Panduan Pengguna IAM.

Untuk detail tentang tindakan dan jenis sumber daya yang ditentukan oleh Amazon DocumentDB, termasuk format ARN untuk setiap jenis sumber daya, lihat Kunci tindakan, sumber daya, dan kondisi untuk Amazon Relational Database Service dalam Referensi Otorisasi Layanan.

> **Note**
>
> Untuk fitur pengelolaan tertentu, Amazon DocumentDB menggunakan teknologi operasional yang dibagi dengan Amazon Relational Database Service (Amazon RDS).
> Untuk tindakan kebijakan untuk klaster elastis Amazon DocumentDB, lihat Kunci tindakan, sumber daya, dan kondisi untuk Amazon DocumentDB Elastic Cluster dalam Referensi Otorisasi Layanan.

Topik

- Praktik terbaik kebijakan
- Menggunakan konsol Amazon DocumentDB
- Mengizinkan pengguna melihat izin mereka sendiri

Praktik terbaik kebijakan

Identity-based kebijakan menentukan apakah seseorang dapat membuat, mengakses, atau menghapus sumber daya Amazon DocumentDB di akun Anda. Tindakan ini membuat Akun AWS Anda dikenai biaya. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Mulailah dengan kebijakan AWS terkelola dan beralih ke izin hak istimewa paling sedikit — Untuk mulai memberikan izin kepada pengguna dan beban kerja Anda, gunakan kebijakan AWS terkelola yang memberikan izin untuk banyak kasus penggunaan umum. Mereka tersedia di Anda Akun AWS. Kami menyarankan Anda mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola AWS pelanggan yang khusus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat Kebijakan yang dikelola AWS atau Kebijakan yang dikelola AWS untuk fungsi tugas dalam Panduan Pengguna IAM.
- Menerapkan izin dengan hak akses paling rendah – Ketika Anda menetapkan izin dengan kebijakan IAM, hanya berikan izin yang diperlukan untuk melakukan tugas. Anda melakukannya dengan mendefinisikan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, yang juga dikenal sebagai izin dengan hak akses paling rendah. Untuk informasi selengkapnya tentang cara menggunakan IAM untuk mengajukan izin, lihat Kebijakan dan izin dalam IAM dalam Panduan Pengguna IAM.
- Gunakan kondisi dalam kebijakan IAM untuk membatasi akses lebih lanjut – Anda dapat menambahkan suatu kondisi ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Sebagai contoh, Anda dapat menulis kondisi kebijakan untuk menentukan bahwa semua permintaan harus dikirim menggunakan SSL. Anda juga dapat menggunakan ketentuan untuk memberikan akses ke tindakan layanan jika digunakan melalui yang spesifik Layanan AWS, seperti CloudFormation. Untuk informasi selengkapnya, lihat Elemen kebijakan JSON IAM: Kondisi dalam Panduan Pengguna IAM.
- Gunakan IAM Access Analyzer untuk memvalidasi kebijakan IAM Anda untuk memastikan izin yang aman dan fungsional – IAM Access Analyzer memvalidasi kebijakan baru dan yang sudah ada sehingga kebijakan tersebut mematuhi bahasa kebijakan IAM (JSON) dan praktik terbaik IAM. IAM Access Analyzer menyediakan lebih dari 100 pemeriksaan kebijakan dan rekomendasi yang

dapat ditindaklanjuti untuk membantu Anda membuat kebijakan yang aman dan fungsional. Untuk informasi selengkapnya, lihat Validasi kebijakan dengan IAM Access Analyzer dalam Panduan Pengguna IAM.

- Memerlukan otentikasi multi-faktor (MFA) - Jika Anda memiliki skenario yang mengharuskan pengguna IAM atau pengguna root di Anda, Akun AWS aktifkan MFA untuk keamanan tambahan. Untuk meminta MFA ketika operasi API dipanggil, tambahkan kondisi MFA pada kebijakan Anda. Untuk informasi selengkapnya, lihat Amankan akses API dengan MFA dalam Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang praktik terbaik dalam IAM, lihat Praktik terbaik keamanan di IAM dalam Panduan Pengguna IAM.

Menggunakan konsol Amazon DocumentDB

Untuk mengakses konsol Amazon DocumentDB (dengan kompatibilitas MongoDB), Anda harus memiliki set izin minimum. Izin ini harus memungkinkan Anda untuk membuat daftar dan melihat detail tentang sumber daya Amazon DocumentDB di Anda. Akun AWS Jika Anda membuat kebijakan berbasis identitas yang lebih ketat daripada izin minimum yang diperlukan, konsol tidak akan berfungsi sebagaimana mestinya untuk entitas (pengguna atau peran) dengan kebijakan tersebut.

Anda tidak perlu mengizinkan izin konsol minimum untuk pengguna yang melakukan panggilan hanya ke AWS CLI atau AWS API. Sebagai gantinya, izinkan akses hanya ke tindakan yang sesuai dengan operasi API yang coba mereka lakukan.

Untuk memastikan bahwa pengguna dan peran masih dapat menggunakan konsol Amazon DocumentDB, lampirkan juga Amazon DocumentDB atau kebijakan terkelola ke *ConsoleAccess* entitas *ReadOnly* AWS . Untuk informasi selengkapnya, lihat Menambah izin untuk pengguna dalam Panduan Pengguna IAM.

Mengizinkan pengguna melihat izin mereka sendiri

Contoh ini menunjukkan cara membuat kebijakan yang mengizinkan pengguna IAM melihat kebijakan inline dan terkelola yang dilampirkan ke identitas pengguna mereka. Kebijakan ini mencakup izin untuk menyelesaikan tindakan ini di konsol atau menggunakan API atau secara terprogram. AWS CLI AWS

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Memecahkan masalah identitas dan akses Amazon DocumentDB

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan Amazon DocumentDB dan IAM.

Topik

- Saya tidak berwenang untuk melakukan tindakan di Amazon DocumentDB
- Saya tidak berwenang untuk melakukan iam: PassRole
- Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses sumber daya Amazon DocumentDB saya

Saya tidak berwenang untuk melakukan tindakan di Amazon DocumentDB

Jika Anda menerima pesan kesalahan bahwa Anda tidak memiliki otorisasi untuk melakukan tindakan, kebijakan Anda harus diperbarui agar Anda dapat melakukan tindakan tersebut.

Contoh kesalahan berikut terjadi ketika pengguna IAM `mateojackson` mencoba menggunakan konsol untuk melihat detail tentang suatu sumber daya *`my-example-widget`* rekaan, tetapi tidak memiliki izin `aws:`*`GetWidget`* rekaan.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 aws:GetWidget on resource: my-example-widget
```

Dalam hal ini, kebijakan untuk pengguna `mateojackson` harus diperbarui untuk mengizinkan akses ke sumber daya *`my-example-widget`* dengan menggunakan tindakan `aws:`*`GetWidget`*.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya tidak berwenang untuk melakukan iam: PassRole

Jika Anda menerima kesalahan yang tidak diizinkan untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran ke Amazon DocumentDB.

Beberapa Layanan AWS memungkinkan Anda untuk meneruskan peran yang ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran terkait layanan. Untuk melakukannya, Anda harus memiliki izin untuk meneruskan peran ke layanan.

Contoh kesalahan berikut terjadi ketika pengguna IAM bernama `marymajor` mencoba menggunakan konsol untuk melakukan tindakan di Amazon DocumentDB. Namun, tindakan tersebut memerlukan layanan untuk mendapatkan izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut pada layanan.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dalam kasus ini, kebijakan Mary harus diperbarui agar dia mendapatkan izin untuk melakukan tindakan `iam:PassRole` tersebut.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses sumber daya Amazon DocumentDB saya

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau orang-orang di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa saja yang dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis sumber daya atau daftar kontrol akses (ACL), Anda dapat menggunakan kebijakan tersebut untuk memberi orang akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa referensi berikut:

- Untuk mengetahui apakah Amazon DocumentDB mendukung fitur-fitur ini, lihat. Bagaimana Amazon DocumentDB bekerja dengan IAM
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda di seluruh sumber daya Akun AWS yang Anda miliki, lihat Menyediakan akses ke pengguna IAM di pengguna lain Akun AWS yang Anda miliki di Panduan Pengguna IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda kepada pihak ketiga Akun AWS, lihat Menyediakan akses yang Akun AWS dimiliki oleh pihak ketiga dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, lihat Menyediakan akses ke pengguna terautentikasi eksternal (federasi identitas) dalam Panduan Pengguna IAM.
- Untuk mempelajari perbedaan antara menggunakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Akses sumber daya lintas akun di IAM di Panduan Pengguna IAM.

Mengelola izin akses ke sumber daya Amazon DocumentDB Anda

Setiap AWS sumber daya dimiliki oleh Akun AWS, dan izin untuk membuat atau mengakses sumber daya diatur oleh kebijakan izin. Administrator akun dapat melampirkan kebijakan izin ke identitas IAM (yaitu, pengguna, grup, dan peran), dan beberapa layanan (seperti AWS Lambda) juga mendukung melampirkan kebijakan izin ke sumber daya.

> **Note**
>
> Administrator akun (atau pengguna administrator) adalah pengguna dengan izin administrator. Untuk informasi selengkapnya, lihat Praktik Terbaik IAM dalam Panduan Pengguna IAM.

Topik

- Sumber daya dan operasi Amazon DocumentDB
- Memahami kepemilikan sumber daya
- Mengelola akses ke sumber daya
- Menentukan elemen kebijakan: tindakan, efek, sumber daya, dan prinsip
- Menentukan kondisi dalam kebijakan

Sumber daya dan operasi Amazon DocumentDB

Di Amazon DocumentDB, sumber daya primer adalah klaster. Amazon DocumentDB mendukung sumber daya lain yang dapat digunakan dengan sumber daya primer seperti instans, grup parameter, dan langganan peristiwa. Sumber daya tersebut dirujuk sebagai sub-sumber daya.

Sumber daya dan sub-sumber daya ini memiliki Amazon Resource Names (ARN) unik yang terkait dengannya, seperti yang ditunjukkan dalam tabel berikut.

Jenis Sumber Daya	Format ARN
Kluster	`arn:aws:rds:` *`region`*`:`*`account-id`* `:cluster:` *`db-cluster-name`*
Grup parameter klaster	`arn:aws:rds:` *`region`*`:`*`account-id`* `:cluster-pg:` *`cluster-parameter-group-name`*
Snapshot klaster	`arn:aws:rds:` *`region`*`:`*`account-id`* `:cluster-snapshot:` *`cluster-snapshot-name`*
Instans	`arn:aws:rds:` *`region`*`:`*`account-id`* `:db:`*`db-instance-name`*
Grup keamanan	`arn:aws:rds:` *`region`*`:`*`account-id`* `:secgrp:`*`security-group-name`*
Grup subnet	`arn:aws:rds:` *`region`*`:`*`account-id`* `:subgrp:`*`subnet-group-name`*

Amazon DocumentDB menyediakan serangkaian operasi untuk bekerja dengan sumber daya Amazon DocumentDB. Untuk daftar operasi yang tersedia, lihat Tindakan.

Memahami kepemilikan sumber daya

Pemilik sumber daya adalah Akun AWS yang menciptakan sumber daya. Artinya, pemilik sumber daya adalah entitas utama (akun root, pengguna IAM, atau peran IAM) yang mengautentikasi permintaan yang membuat sumber daya. Akun AWS Contoh berikut menggambarkan cara kerjanya:

- Jika Anda menggunakan kredensyal akun root Anda Akun AWS untuk membuat sumber daya Amazon DocumentDB, seperti instance, Akun AWS Anda adalah pemilik sumber daya Amazon DocumentDB.
- Jika Anda membuat pengguna IAM di Anda Akun AWS dan memberikan izin untuk membuat sumber daya Amazon DocumentDB kepada pengguna tersebut, pengguna dapat membuat sumber daya Amazon DocumentDB. Namun, milik Anda Akun AWS, yang menjadi milik pengguna, memiliki sumber daya Amazon DocumentDB.
- Jika Anda membuat peran IAM Akun AWS dengan izin untuk membuat sumber daya Amazon DocumentDB, siapa pun yang dapat mengambil peran tersebut dapat membuat sumber daya Amazon DocumentDB. Anda Akun AWS, yang menjadi milik peran tersebut, memiliki sumber daya Amazon DocumentDB.

Mengelola akses ke sumber daya

Kebijakan izin menjelaskan siapa yang memiliki akses ke suatu objek. Bagian berikut menjelaskan opsi yang tersedia untuk membuat kebijakan izin.

> **Note**
>
> Bagian ini membahas penggunaan IAM dalam konteks Amazon DocumentDB. Bagian ini tidak memberikan informasi yang mendetail tentang layanan IAM. Untuk dokumentasi IAM lengkap, lihat Apa yang Dimaksud dengan IAM? dalam Panduan Pengguna IAM. Untuk informasi tentang sintaksis dan deskripsi kebijakan IAM, lihat Referensi Kebijakan IAM AWS dalam Panduan Pengguna IAM.

Kebijakan yang terlampir pada identitas IAM disebut sebagai kebijakan berbasis identitas (kebijakan IAM). Kebijakan yang terlampir pada sumber daya disebut sebagai kebijakan berbasis sumber daya. Amazon DocumentDB hanya mendukung kebijakan berbasis identitas (kebijakan IAM).

Topik

- Identity-based kebijakan (kebijakan IAM)
- Resource-based kebijakan

Identity-based kebijakan (kebijakan IAM)

Anda dapat melampirkan kebijakan ke identitas IAM Anda. Misalnya, Anda dapat melakukan hal berikut:

- Lampirkan kebijakan izin ke pengguna atau grup dalam akun Anda – Administrator akun dapat menggunakan kebijakan izin yang berkaitan dengan pengguna tertentu untuk memberikan izin bagi pengguna tersebut untuk membuat sumber daya Amazon DocumentDB, seperti instans.
- Lampirkan kebijakan izin untuk peran (memberikan izin lintas akun) – Anda dapat melampirkan kebijakan izin berbasis identitas ke IAM role untuk memberikan izin lintas akun. Misalnya, administrator dapat membuat peran untuk memberikan izin lintas akun kepada orang lain Akun AWS atau AWS layanan sebagai berikut:
  1. Administrator akun A membuat peran IAM dan melampirkan kebijakan izin ke peran ini yang memberikan izin pada sumber daya di akun A.
  2. Administrator akun A melampirkan kebijakan kepercayaan ke peran yang mengidentifikasi Akun B sebagai prinsipal yang dapat mengambil peran tersebut.
  3. Administrator Akun B kemudian dapat mendelegasikan izin untuk mengambil peran kepada pengguna mana pun di Akun B. Melakukan hal ini memungkinkan pengguna di Akun B untuk membuat atau mengakses sumber daya di Akun A. Prinsip dalam kebijakan kepercayaan juga dapat menjadi prinsipal AWS layanan jika Anda ingin memberikan izin ke AWS layanan untuk mengambil peran tersebut.

  Untuk informasi selengkapnya tentang penggunaan IAM untuk mendelegasikan izin, lihat Manajemen Akses dalam Panduan Pengguna IAM.

Berikut ini adalah kebijakan contoh yang mengizinkan pengguna dengan ID `123456789012` untuk membuat instans untuk Akun AWS Anda. Instans yang baru harus menggunakan grup opsi dan grup parameter yang dimulai dengan `default`, dan harus menggunakan grup subnet `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"
            ]
        }
    ]
}
```

Untuk informasi selengkapnya tentang menggunakan kebijakan berbasis identitas dengan Amazon DocumentDB, lihat Menggunakan kebijakan berbasis identitas (kebijakan IAM) untuk Amazon DocumentDB. Untuk informasi lebih lanjut tentang pengguna, grup, peran, dan izin, lihat Identitas (Pengguna, Grup, dan Peran) dalam Panduan Pengguna IAM.

Resource-based kebijakan

Layanan lainnya, seperti Amazon Simple Storage Service (Amazon S3), juga mendukung kebijakan izin berbasis sumber daya. Misalnya, Anda dapat melampirkan kebijakan ke bucket Amazon S3 untuk mengelola izin akses ke bucket tersebut. Amazon DocumentDB tidak mendukung kebijakan berbasis sumber daya.

Menentukan elemen kebijakan: tindakan, efek, sumber daya, dan prinsip

Untuk setiap sumber daya Amazon DocumentDB (lihat Sumber daya dan operasi Amazon DocumentDB), layanan menentukan serangkaian operasi API. Untuk informasi selengkapnya, lihat Tindakan. Untuk memberikan izin bagi operasi API tersebut, Amazon DocumentDB menentukan serangkaian tindakan yang dapat Anda tentukan dalam kebijakan. Melakukan operasi API dapat memerlukan izin untuk lebih dari satu tindakan.

Berikut ini adalah elemen-elemen kebijakan dasar:

- Sumber Daya – Dalam kebijakan, Anda menggunakan Amazon Resource Name (ARN) untuk mengidentifikasi sumber daya yang diberlakukan oleh kebijakan tersebut.
- Tindakan – Anda menggunakan kata kunci tindakan untuk mengidentifikasi operasi sumber daya yang ingin Anda izinkan atau tolak. Misalnya, izin `rds:DescribeDBInstances` memungkinkan pengguna untuk melakukan operasi `DescribeDBInstances`.
- Efek – Anda menetapkan efek ketika pengguna meminta tindakan tertentu—hal ini dapat mengizinkan maupun menolak. Jika Anda tidak secara eksplisit memberikan akses ke (mengizinkan) sumber daya, akses akan ditolak secara implisit. Anda juga dapat secara eksplisit menolak akses ke sumber daya, yang mungkin Anda lakukan untuk memastikan bahwa pengguna tidak dapat mengaksesnya, meskipun kebijakan yang berbeda memberikan akses.
- Principal – Dalam kebijakan berbasis identitas (Kebijakan IAM), pengguna yang kebijakannya terlampir adalah principal yang implisit. Untuk kebijakan berbasis sumber daya, Anda menentukan pengguna, akun, layanan, atau entitas lain yang diinginkan untuk menerima izin (berlaku hanya untuk kebijakan berbasis sumber daya). Amazon DocumentDB tidak mendukung kebijakan berbasis sumber daya.

Untuk mempelajari selengkapnya tentang sintaksis dan penjelasan kebijakan IAM, lihat AWS Referensi Kebijakan IAM dalam Panduan Pengguna IAM.

Untuk tabel yang menampilkan semua Tindakan Amazon DocumentDB API dan sumber daya yang diterapkan, lihat Izin API Amazon DocumentDB: referensi tindakan, sumber daya, dan kondisi.

Menentukan kondisi dalam kebijakan

Ketika Anda memberikan izin, Anda dapat menggunakan bahasa kebijakan IAM untuk menentukan kondisi ketika kebijakan harus berlaku. Misalnya, Anda mungkin ingin kebijakan diterapkan hanya setelah tanggal tertentu. Untuk informasi selengkapnya tentang menentukan kondisi dalam bahasa kebijakan, lihat Kondisi dalam Panduan Pengguna IAM.

Untuk menyatakan kondisi, Anda menggunakan kunci kondisi standar. Amazon DocumentDB tidak memiliki kunci konteks layanan khusus yang dapat digunakan dalam kebijakan IAM. Untuk daftar kunci konteks syarat global yang tersedia untuk semua layanan, lihat Kunci yang Tersedia untuk Syarat di Panduan Pengguna IAM.

Menggunakan kebijakan berbasis identitas (kebijakan IAM) untuk Amazon DocumentDB

> **Important**
>
> Untuk fitur manajemen tertentu, Amazon DocumentDB menggunakan teknologi operasional yang dibagi dengan Amazon RDS. Konsol Amazon DocumentDB AWS CLI,, dan panggilan API dicatat sebagai panggilan yang dilakukan ke Amazon RDS API.
> Kami menyarankan Anda untuk terlebih dahulu meninjau topik pendahuluan yang menjelaskan konsep dasar dan opsi yang tersedia bagi Anda untuk mengelola akses ke sumber daya Amazon DocumentDB Anda. Untuk informasi selengkapnya, lihat Mengelola izin akses ke sumber daya Amazon DocumentDB Anda.

Topik ini menyediakan contoh kebijakan berbasis identitas di mana administrator akun dapat melampirkan kebijakan izin ke identitas IAM (yaitu, pengguna, grup, dan peran).

Berikut ini adalah contoh kebijakan 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"
            ]
        }
    ]
}
```

Kebijakan ini mencakup pernyataan tunggal yang menentukan izin berikut untuk pengguna IAM:

- Kebijakan ini memungkinkan pengguna IAM untuk membuat instance menggunakan tindakan createdBInstance (ini juga berlaku untuk operasi create-db-instance dan). AWS CLI Konsol Manajemen AWS
- Elemen `Resource` menentukan bahwa pengguna dapat melakukan tindakan pada atau dengan sumber daya. Anda menentukan sumber daya menggunakan Amazon Resource Name (ARN). ARN ini mencakup nama layanan yang dimiliki sumber daya (`rds`), Wilayah AWS (*menunjukkan Wilayah apa pun dalam contoh ini), nomor akun pengguna (`123456789012`adalah ID pengguna dalam contoh ini), dan jenis sumber daya.

  Elemen `Resource` dalam contoh menentukan batasan kebijakan berikut pada sumber daya untuk pengguna:

  - Pengidentifikasi instans untuk instans baru harus dimulai dengan `test` (sebagai contoh, `testCustomerData1`, `test-region2-data`).
  - Grup parameter klaster untuk instans baru harus dimulai dengan `default`.
  - Grup subnet untuk instans baru harus grup subnet `default`.

Kebijakan tidak menentukan elemen `Principal` karena dalam kebijakan berbasis identitas, Anda tidak menentukan prinsipal yang mendapatkan izin. Saat Anda melampirkan kebijakan kepada pengguna, pengguna tersebut menjadi prinsipal secara implisit. Saat Anda melampirkan kebijakan izin pada IAM role, penanggung jawab yang diidentifikasi dalam kebijakan kepercayaan peran mendapatkan izin.

Untuk tabel yang menampilkan semua operasi Amazon DocumentDB API dan sumber daya yang mereka terapkan, lihat Izin API Amazon DocumentDB: referensi tindakan, sumber daya, dan kondisi.

Izin diperlukan untuk menggunakan konsol Amazon DocumentDB

Agar pengguna dapat bekerja dengan konsol Amazon DocumentDB, pengguna tersebut harus memiliki rangkaian izin minimum. Izin ini memungkinkan pengguna untuk mendeskripsikan sumber daya Amazon DocumentDB Akun AWS untuk mereka dan untuk memberikan informasi terkait lainnya, termasuk keamanan Amazon EC2 dan informasi jaringan.

Jika Anda membuat kebijakan IAM yang lebih ketat daripada izin minimum yang diperlukan, konsol tidak akan berfungsi sebagaimana mestinya bagi pengguna dengan kebijakan IAM tersebut. Untuk memastikan bahwa pengguna tersebut masih dapat menggunakan konsol Amazon DocumentDB, lampirkan juga kebijakan terkelola `AmazonDocDBConsoleFullAccess` kepada pengguna, sebagaimana dijelaskan dalam AWS kebijakan terkelola untuk Amazon DocumentDB.

Anda tidak perlu mengizinkan izin konsol minimum untuk pengguna yang melakukan panggilan hanya ke API Amazon DocumentDB AWS CLI atau Amazon DocumentDB.

Contoh kebijakan yang dikelola pelanggan

Dalam bagian ini, Anda dapat menemukan contoh kebijakan pengguna yang memberikan izin untuk berbagai tindakan Amazon DocumentDB. Kebijakan ini berfungsi saat Anda menggunakan tindakan Amazon DocumentDB API AWS , SDK, atau file. AWS CLI Saat Anda menggunakan konsol, Anda perlu memberikan izin tambahan yang khusus untuk konsol. yang dibahas dalam Izin diperlukan untuk menggunakan konsol Amazon DocumentDB.

Untuk fitur manajemen tertentu, Amazon DocumentDB menggunakan teknologi operasional yang dibagi dengan Amazon Relational Database Service (Amazon RDS) dan Amazon Neptune.

> **Note**
>
> Semua contoh menggunakan Wilayah US East (N. Virginia) (`us-east-1`) dan berisi ID akun fiktif.

Contoh

- Contoh 1: Izinkan pengguna melakukan tindakan deskripsi apa pun pada sumber daya Amazon DocumentDB apa pun
- Contoh 2: Mencegah pengguna menghapus instance
- Contoh 3: Mencegah pengguna membuat cluster kecuali enkripsi penyimpanan diaktifkan

Contoh 1: Izinkan pengguna melakukan tindakan deskripsi apa pun pada sumber daya Amazon DocumentDB apa pun

Kebijakan izin berikut ini memberikan izin kepada pengguna untuk menjalankan semua tindakan yang dimulai dengan `Describe`. Tindakan tersebut menunjukkan informasi tentang sumber daya

Amazon DocumentDB, seperti instans. Karakter wildcard (*) di elemen `Resource` menunjukkan bahwa tindakan diizinkan untuk semua sumber daya Amazon DocumentDB yang dimiliki akun tersebut.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowRDSDescribe",
         "Effect":"Allow",
         "Action":"rds:Describe*",
         "Resource":"*"
      }
   ]
}
```

Contoh 2: Mencegah pengguna menghapus instance

Kebijakan izin berikut ini memberikan izin untuk mencegah pengguna menghapus instans tertentu. Sebagai contoh, Anda mungkin ingin menolak kemampuan untuk menghapus instans produksi Anda kepada setiap pengguna yang bukan administrator.

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"
      }
   ]
}
```

Contoh 3: Mencegah pengguna membuat cluster kecuali enkripsi penyimpanan diaktifkan

Kebijakan izin berikut ini menolak izin bagi pengguna untuk membuat klaster Amazon DocumentDB kecuali enkripsi penyimpanan diaktifkan.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "PreventUnencryptedDocumentDB",
         "Effect": "Deny",
         "Action": "RDS:CreateDBCluster",
         "Condition": {
         "Bool": {
         "rds:StorageEncrypted": "false"
      },
         "StringEquals": {
         "rds:DatabaseEngine": "docdb"
         }
      },
      "Resource": "*"
      }
   ]
}
```

AWS kebijakan terkelola untuk Amazon DocumentDB

Untuk menambahkan izin ke pengguna, grup, dan peran, lebih mudah menggunakan kebijakan AWS terkelola daripada menulis kebijakan sendiri. Dibutuhkan waktu dan keahlian untuk membuat kebijakan yang dikelola pelanggan IAM yang hanya memberi tim Anda izin yang mereka butuhkan. Untuk memulai dengan cepat, Anda dapat menggunakan kebijakan AWS terkelola kami. Kebijakan ini mencakup kasus penggunaan umum dan tersedia di AWS akun Anda. Untuk informasi selengkapnya tentang kebijakan AWS terkelola, lihat kebijakan AWS terkelola di Panduan Pengguna AWS Identity and Access Management.

AWS layanan memelihara dan memperbarui kebijakan AWS terkelola. Anda tidak dapat mengubah izin dalam kebijakan AWS terkelola. Layanan terkadang menambahkan izin tambahan ke kebijakan AWS terkelola untuk mendukung fitur baru. Jenis pembaruan ini akan memengaruhi semua identitas

(pengguna, grup, dan peran) di mana kebijakan tersebut dilampirkan. Layanan kemungkinan besar akan memperbarui kebijakan AWS terkelola saat fitur baru diluncurkan atau saat operasi baru tersedia. Layanan tidak menghapus izin dari kebijakan AWS terkelola, sehingga pembaruan kebijakan tidak akan merusak izin yang ada.

Selain itu, AWS mendukung kebijakan terkelola untuk fungsi pekerjaan yang mencakup beberapa layanan. Misalnya, kebijakan `ViewOnlyAccess` AWS terkelola menyediakan akses hanya-baca ke banyak AWS layanan dan sumber daya. Saat layanan meluncurkan fitur baru, AWS tambahkan izin hanya-baca untuk operasi dan sumber daya baru. Untuk daftar dan deskripsi kebijakan fungsi pekerjaan, lihat kebijakan AWS terkelola untuk fungsi pekerjaan di Panduan Pengguna AWS Identity and Access Management.

Kebijakan AWS terkelola berikut, yang dapat Anda lampirkan ke pengguna di akun Anda, khusus untuk Amazon DocumentDB:

- AmazonDocDBFullAccess— Memberikan akses penuh ke semua sumber daya Amazon DocumentDB untuk akun root. AWS
- AmazonDocDBReadOnlyAccess— Memberikan akses hanya-baca ke semua sumber daya Amazon DocumentDB untuk akun root. AWS
- AmazonDocDBConsoleFullAccess— Memberikan akses penuh untuk mengelola sumber daya cluster elastis Amazon DocumentDB dan Amazon DocumentDB menggunakan file. Konsol Manajemen AWS
- AmazonDocDBElasticReadOnlyAccess— Memberikan akses hanya-baca ke semua sumber daya cluster elastis Amazon DocumentDB untuk akun root. AWS
- AmazonDocDBElasticFullAccess— Memberikan akses penuh ke semua sumber daya cluster elastis Amazon DocumentDB untuk akun root. AWS

AmazonDocDBFullAccess

Kebijakan ini memberikan izin administratif yang memungkinkan akses penuh utama ke semua tindakan Amazon DocumentDB. Izin dalam kebijakan ini dikelompokkan sebagai berikut:

- Izin Amazon DocumentDB memungkinkan semua tindakan Amazon DocumentDB.
- Beberapa izin Amazon EC2 dalam kebijakan ini diperlukan untuk memvalidasi sumber daya yang diteruskan dalam permintaan API. Ini untuk memastikan Amazon DocumentDB berhasil menggunakan sumber daya dengan cluster. Izin Amazon EC2 lainnya dalam kebijakan ini

memungkinkan Amazon DocumentDB membuat AWS sumber daya yang diperlukan untuk memungkinkan Anda terhubung ke cluster Anda.

- Izin Amazon DocumentDB digunakan selama panggilan API untuk memvalidasi sumber daya yang diteruskan dalam permintaan. Mereka diperlukan untuk Amazon DocumentDB untuk dapat menggunakan kunci yang diteruskan dengan cluster Amazon DocumentDB.
- CloudWatch Log diperlukan untuk Amazon DocumentDB untuk dapat memastikan bahwa tujuan pengiriman log dapat dijangkau, dan valid untuk penggunaan log 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

Kebijakan ini memberikan izin hanya-baca yang memungkinkan pengguna melihat informasi di Amazon DocumentDB. Prinsipal dengan kebijakan terlampir ini tidak dapat membuat pembaruan atau menghapus sumber daya yang keluar, juga tidak dapat membuat sumber daya Amazon DocumentDB baru. Misalnya, prinsipal dengan izin ini dapat melihat daftar cluster dan konfigurasi yang terkait dengan akun mereka, tetapi tidak dapat mengubah konfigurasi atau pengaturan cluster apa pun. Izin dalam kebijakan ini dikelompokkan sebagai berikut:

- Izin Amazon DocumentDB memungkinkan Anda mencantumkan sumber daya Amazon DocumentDB, menjelaskannya, dan mendapatkan informasi tentangnya.
- Izin Amazon EC2 digunakan untuk menggambarkan VPC Amazon, subnet, grup keamanan, dan ENI yang terkait dengan kluster.
- Izin Amazon DocumentDB digunakan untuk menggambarkan kunci yang terkait dengan 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

Memberikan akses penuh untuk mengelola sumber daya Konsol Manajemen AWS Amazon DocumentDB menggunakan berikut ini:

- Izin Amazon DocumentDB untuk mengizinkan semua tindakan cluster Amazon DocumentDB dan Amazon DocumentDB.
- Beberapa izin Amazon EC2 dalam kebijakan ini diperlukan untuk memvalidasi sumber daya yang diteruskan dalam permintaan API. Ini untuk memastikan Amazon DocumentDB berhasil menggunakan sumber daya untuk menyediakan dan memelihara cluster. Izin Amazon EC2 lainnya dalam kebijakan ini memungkinkan Amazon DocumentDB membuat AWS sumber daya yang diperlukan untuk memungkinkan Anda terhubung ke cluster seperti VPcendPoint.
- AWS KMS izin digunakan selama panggilan API AWS KMS untuk memvalidasi sumber daya yang diteruskan dalam permintaan. Mereka diperlukan untuk Amazon DocumentDB untuk dapat menggunakan kunci yang diteruskan untuk mengenkripsi dan mendekripsi data saat istirahat dengan cluster elastis Amazon DocumentDB.
- CloudWatch Log diperlukan untuk Amazon DocumentDB agar dapat memastikan bahwa tujuan pengiriman log dapat dijangkau, dan valid untuk mengaudit dan membuat profil penggunaan log.
- Izin Secrets Manager diperlukan untuk memvalidasi rahasia yang diberikan dan menggunakannya untuk mengatur pengguna admin untuk cluster elastis Amazon DocumentDB.
- Izin Amazon RDS diperlukan untuk tindakan pengelolaan klaster Amazon DocumentDB. Untuk fitur manajemen tertentu, Amazon DocumentDB menggunakan teknologi operasional yang dibagi dengan Amazon RDS.
- Izin SNS memungkinkan prinsipal untuk langganan dan topik Amazon Simple Notification Service (Amazon SNS), dan untuk mempublikasikan pesan Amazon SNS.
- Izin IAM diperlukan untuk membuat peran terkait layanan yang diperlukan untuk metrik dan penerbitan log.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DocdbSids",
            "Effect": "Allow",
            "Action": [
                "docdb-elastic:CreateCluster",
                "docdb-elastic:UpdateCluster",
                "docdb-elastic:GetCluster",
                "docdb-elastic:DeleteCluster",
                "docdb-elastic:ListClusters",
                "docdb-elastic:CreateClusterSnapshot",
                "docdb-elastic:GetClusterSnapshot",
                "docdb-elastic:DeleteClusterSnapshot",
                "docdb-elastic:ListClusterSnapshots",
                "docdb-elastic:RestoreClusterFromSnapshot",
                "docdb-elastic:TagResource",
                "docdb-elastic:UntagResource",
                "docdb-elastic:ListTagsForResource",
                "docdb-elastic:CopyClusterSnapshot",
                "docdb-elastic:StartCluster",
                "docdb-elastic:StopCluster",
                "docdb-elastic:GetPendingMaintenanceAction",
                "docdb-elastic:ListPendingMaintenanceActions",
                "docdb-elastic:ApplyPendingMaintenanceAction",
                "rds:AddRoleToDBCluster",
                "rds:AddSourceIdentifierToSubscription",
                "rds:AddTagsToResource",
                "rds:ApplyPendingMaintenanceAction",
                "rds:CopyDBClusterParameterGroup",
                "rds:CopyDBClusterSnapshot",
                "rds:CopyDBParameterGroup",
                "rds:CreateDBCluster",
                "rds:CreateDBClusterParameterGroup",
                "rds:CreateDBClusterSnapshot",
                "rds:CreateDBInstance",
                "rds:CreateDBParameterGroup",
                "rds:CreateDBSubnetGroup",
                "rds:CreateEventSubscription",
                "rds:CreateGlobalCluster",
```

```
                "rds:DeleteDBCluster",
                "rds:DeleteDBClusterParameterGroup",
                "rds:DeleteDBClusterSnapshot",
                "rds:DeleteDBInstance",
                "rds:DeleteDBParameterGroup",
                "rds:DeleteDBSubnetGroup",
                "rds:DeleteEventSubscription",
                "rds:DeleteGlobalCluster",
                "rds:DescribeAccountAttributes",
                "rds:DescribeCertificates",
                "rds:DescribeDBClusterParameterGroups",
                "rds:DescribeDBClusterParameters",
                "rds:DescribeDBClusterSnapshotAttributes",
                "rds:DescribeDBClusterSnapshots",
                "rds:DescribeDBClusters",
                "rds:DescribeDBEngineVersions",
                "rds:DescribeDBInstances",
                "rds:DescribeDBLogFiles",
                "rds:DescribeDBParameterGroups",
                "rds:DescribeDBParameters",
                "rds:DescribeDBSecurityGroups",
                "rds:DescribeDBSubnetGroups",
                "rds:DescribeEngineDefaultClusterParameters",
                "rds:DescribeEngineDefaultParameters",
                "rds:DescribeEventCategories",
                "rds:DescribeEventSubscriptions",
                "rds:DescribeEvents",
                "rds:DescribeGlobalClusters",
                "rds:DescribeOptionGroups",
                "rds:DescribeOrderableDBInstanceOptions",
                "rds:DescribePendingMaintenanceActions",
                "rds:DescribeValidDBInstanceModifications",
                "rds:DownloadDBLogFilePortion",
                "rds:FailoverDBCluster",
                "rds:ListTagsForResource",
                "rds:ModifyDBCluster",
                "rds:ModifyDBClusterParameterGroup",
                "rds:ModifyDBClusterSnapshotAttribute",
                "rds:ModifyDBInstance",
                "rds:ModifyDBParameterGroup",
                "rds:ModifyDBSubnetGroup",
                "rds:ModifyEventSubscription",
                "rds:ModifyGlobalCluster",
                "rds:PromoteReadReplicaDBCluster",
```

```
                "rds:RebootDBInstance",
                "rds:RemoveFromGlobalCluster",
                "rds:RemoveRoleFromDBCluster",
                "rds:RemoveSourceIdentifierFromSubscription",
                "rds:RemoveTagsFromResource",
                "rds:ResetDBClusterParameterGroup",
                "rds:ResetDBParameterGroup",
                "rds:RestoreDBClusterFromSnapshot",
                "rds:RestoreDBClusterToPointInTime"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "DependencySids",
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "cloudwatch:GetMetricData",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics",
                "ec2:AllocateAddress",
                "ec2:AssignIpv6Addresses",
                "ec2:AssignPrivateIpAddresses",
                "ec2:AssociateAddress",
                "ec2:AssociateRouteTable",
                "ec2:AssociateSubnetCidrBlock",
                "ec2:AssociateVpcCidrBlock",
                "ec2:AttachInternetGateway",
                "ec2:AttachNetworkInterface",
                "ec2:CreateCustomerGateway",
                "ec2:CreateDefaultSubnet",
                "ec2:CreateDefaultVpc",
                "ec2:CreateInternetGateway",
                "ec2:CreateNatGateway",
                "ec2:CreateNetworkInterface",
                "ec2:CreateRoute",
                "ec2:CreateRouteTable",
                "ec2:CreateSecurityGroup",
                "ec2:CreateSubnet",
                "ec2:CreateVpc",
                "ec2:CreateVpcEndpoint",
                "ec2:DescribeAccountAttributes",
```

```
                "ec2:DescribeAddresses",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeCustomerGateways",
                "ec2:DescribeInstances",
                "ec2:DescribeNatGateways",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribePrefixLists",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroupReferences",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeVpcEndpoints",
                "ec2:DescribeVpcs",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:ModifySubnetAttribute",
                "ec2:ModifyVpcAttribute",
                "ec2:ModifyVpcEndpoint",
                "kms:DescribeKey",
                "kms:ListAliases",
                "kms:ListKeyPolicies",
                "kms:ListKeys",
                "kms:ListRetirableGrants",
                "logs:DescribeLogStreams",
                "logs:GetLogEvents",
                "sns:ListSubscriptions",
                "sns:ListTopics",
                "sns:Publish"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "DocdbSLRSid",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
AWSServiceRoleForRDS",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "rds.amazonaws.com"
                }
            }
```

```
        },
        {
            "Sid": "DocdbElasticSLRSid",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/docdb-
elastic.amazonaws.com/AWSServiceRoleForDocDB-Elastic",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "docdb-elastic.amazonaws.com"
                }
            }
        }
    ]
}
```

AmazonDocDBElasticReadOnlyAccess

Kebijakan ini memberikan izin hanya-baca yang memungkinkan pengguna melihat informasi klaster elastis di Amazon DocumentDB. Prinsipal dengan kebijakan terlampir ini tidak dapat membuat pembaruan atau menghapus sumber daya yang keluar, juga tidak dapat membuat sumber daya Amazon DocumentDB baru. Misalnya, prinsipal dengan izin ini dapat melihat daftar cluster dan konfigurasi yang terkait dengan akun mereka, tetapi tidak dapat mengubah konfigurasi atau pengaturan cluster apa pun. Izin dalam kebijakan ini dikelompokkan sebagai berikut:

- Izin cluster elastis Amazon DocumentDB memungkinkan Anda mencantumkan sumber daya cluster elastis Amazon DocumentDB, menjelaskannya, dan mendapatkan informasi tentangnya.
- CloudWatch izin digunakan untuk memverifikasi metrik layanan.

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

Kebijakan ini memberikan izin administratif yang memungkinkan akses penuh utama ke semua tindakan Amazon DocumentDB untuk klaster elastis Amazon DocumentDB.

Kebijakan ini menggunakan AWS tag (https://docs.aws.amazon.com/tag-editor/latest/userguide/tagging.html) dalam kondisi untuk cakupan akses ke sumber daya. Jika Anda menggunakan rahasia, itu harus ditandai dengan kunci tag `DocDBElasticFullAccess` dan nilai tag. Jika Anda menggunakan kunci yang dikelola pelanggan, itu harus ditandai dengan kunci tag `DocDBElasticFullAccess` dan nilai tag.

Izin dalam kebijakan ini dikelompokkan sebagai berikut:

- Izin cluster elastis Amazon DocumentDB memungkinkan semua tindakan Amazon DocumentDB.
- Beberapa izin Amazon EC2 dalam kebijakan ini diperlukan untuk memvalidasi sumber daya yang diteruskan dalam permintaan API. Ini untuk memastikan Amazon DocumentDB berhasil menggunakan sumber daya untuk menyediakan dan memelihara cluster. Izin Amazon EC2 lainnya dalam kebijakan ini memungkinkan Amazon DocumentDB membuat AWS sumber daya yang diperlukan untuk memungkinkan Anda terhubung ke cluster Anda seperti titik akhir VPC.
- AWS KMS izin diperlukan agar Amazon DocumentDB dapat menggunakan kunci yang diteruskan untuk mengenkripsi dan mendekripsi data saat istirahat dalam cluster elastis Amazon DocumentDB.

> **Note**
>
> Kunci yang dikelola pelanggan harus memiliki tag dengan kunci `DocDBElasticFullAccess` dan nilai tag.

- SecretsManager izin diperlukan untuk memvalidasi rahasia yang diberikan dan menggunakannya untuk mengatur pengguna admin untuk cluster elastis Amazon DocumentDB.

> **Note**
>
> Rahasia yang digunakan harus memiliki tag dengan kunci `DocDBElasticFullAccess` dan nilai tag.

- Izin IAM diperlukan untuk membuat peran terkait layanan yang diperlukan untuk metrik dan penerbitan log.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DocdbElasticSid",
            "Effect": "Allow",
            "Action": [
                "docdb-elastic:CreateCluster",
                "docdb-elastic:UpdateCluster",
                "docdb-elastic:GetCluster",
                "docdb-elastic:DeleteCluster",
                "docdb-elastic:ListClusters",
                "docdb-elastic:CreateClusterSnapshot",
                "docdb-elastic:GetClusterSnapshot",
                "docdb-elastic:DeleteClusterSnapshot",
                "docdb-elastic:ListClusterSnapshots",
                "docdb-elastic:RestoreClusterFromSnapshot",
                "docdb-elastic:TagResource",
                "docdb-elastic:UntagResource",
                "docdb-elastic:ListTagsForResource",
                "docdb-elastic:CopyClusterSnapshot",
```

```
                "docdb-elastic:StartCluster",
                "docdb-elastic:StopCluster",
                "docdb-elastic:GetPendingMaintenanceAction",
                "docdb-elastic:ListPendingMaintenanceActions",
                "docdb-elastic:ApplyPendingMaintenanceAction"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "EC2Sid",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVpcEndpoint",
                "ec2:DescribeVpcEndpoints",
                "ec2:DeleteVpcEndpoints",
                "ec2:ModifyVpcEndpoint",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:DescribeAvailabilityZones",
                "secretsmanager:ListSecrets"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaFirst": "docdb-elastic.amazonaws.com"
                }
            }
        },
        {
            "Sid": "KMSSid",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
            "Resource": "*",
            "Condition": {
```

```
                "StringLike": {
                    "kms:ViaService": [
                        "docdb-elastic.*.amazonaws.com"
                    ],
                    "aws:ResourceTag/DocDBElasticFullAccess": "*"
                }
            }
        },
        {
            "Sid": "KMSGrantSid",
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aws:ResourceTag/DocDBElasticFullAccess": "*",
                    "kms:ViaService": [
                        "docdb-elastic.*.amazonaws.com"
                    ]
                },
                "Bool": {
                    "kms:GrantIsForAWSResource": true
                }
            }
        },
        {
            "Sid": "SecretManagerSid",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:ListSecretVersionIds",
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:GetResourcePolicy"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "secretsmanager:ResourceTag/DocDBElasticFullAccess": "*"
                },
                "StringEquals": {
                    "aws:CalledViaFirst": "docdb-elastic.amazonaws.com"
                }
```

```
            }
        },
        {
            "Sid": "CloudwatchSid",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "cloudwatch:ListMetrics",
                "cloudwatch:GetMetricStatistics"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "SLRSid",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/docdb-
elastic.amazonaws.com/AWSServiceRoleForDocDB-Elastic",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "docdb-elastic.amazonaws.com"
                }
            }
        }
    ]
}
```

AmazonDocDB-ElasticServiceRolePolicy

Anda tidak dapat melampirkan `AmazonDocDBElasticServiceRolePolicy` ke AWS Identity and Access Management entitas Anda. Kebijakan ini dilampirkan ke peran terkait layanan yang memungkinkan Amazon DocumentDB melakukan tindakan atas nama Anda. Untuk informasi selengkapnya, lihat Service-linked peran dalam kelompok elastis.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": [
                        "AWS/DocDB-Elastic"
                    ]
                }
            }
        }
    ]
}
```

Pembaruan Amazon DocumentDB ke AWS kebijakan terkelola

Ubah	Deskripsi	Date
AmazonDocDBElasticFullAccess, AmazonDocDBConsoleFullAccess - Ubah	Kebijakan diperbarui untuk menambahkan tindakan pemeliharaan yang tertunda.	2/11/2025
AmazonDocDBElasticFullAccess, AmazonDocDBConsoleFullAccess - Ubah	Kebijakan diperbarui untuk menambahkan tindakan snapshot start/stop cluster dan menyalin cluster.	2/21/2024
AmazonDocDBElasticReadOnlyAccess, AmazonDocDBElasticFullAccess - Ubah	Kebijakan diperbarui untuk menambahkan `cloudwatch:GetMetricData` tindakan.	6/21/2023
AmazonDocDBElasticReadOnlyAccess- Kebijakan baru	Kebijakan terkelola baru untuk cluster elastis Amazon DocumentDB.	6/8/2023

Ubah	Deskripsi	Date
AmazonDocDBElastic FullAccess- Kebijakan baru	Kebijakan terkelola baru untuk cluster elastis Amazon DocumentDB.	6/5/2023
AmazonDocDB-Elasti cServiceRolePolicy – Kebijakan baru	Amazon DocumentDB menciptakan peran terkait layanan AWS ServiceRo leForDocDB-Elastic baru untuk cluster elastis Amazon DocumentDB.	11/30/2022
AmazonDocDBConsole FullAccess- Ubah	Kebijakan diperbarui untuk menambahkan izin klaster global dan elastis Amazon DocumentDB.	11/30/2022
AmazonDocDBConsole FullAccess,AmazonDoc DBFullAccess, AmazonDoc DBReadOnlyAccess - Kebijakan Baru	Peluncuran layanan.	1/19/2017

Izin API Amazon DocumentDB: referensi tindakan, sumber daya, dan kondisi

Gunakan bagian berikut ini sebagai referensi ketika Anda mengatur Menggunakan kebijakan berbasis identitas (kebijakan IAM) untuk Amazon DocumentDB dan menulis kebijakan izin yang dapat Anda lampirkan ke identitas IAM (kebijakan berbasis identitas).

Berikut ini daftar setiap operasi Amazon DocumentDB API. Termasuk dalam daftar adalah tindakan terkait yang dapat Anda berikan izin untuk melakukan tindakan, AWS sumber daya yang dapat Anda berikan izin, dan kunci kondisi yang dapat Anda sertakan untuk kontrol akses berbutir halus. Anda menentukan tindakan di bidang `Action` kebijakan, nilai sumber daya di bidang `Resource` kebijakan, dan syarat di bidang `Condition` kebijakan. Untuk informasi lebih lanjut tentang syarat, lihat Menentukan kondisi dalam kebijakan.

Anda dapat menggunakan kunci kondisi AWS-wide dalam kebijakan Amazon DocumentDB Anda untuk menyatakan kondisi. Untuk daftar lengkap tombol AWS-wide, lihat Kunci yang Tersedia di Panduan Pengguna IAM.

Anda dapat menguji kebijakan IAM dengan simulator kebijakan IAM. Secara otomatis menyediakan daftar sumber daya dan parameter yang diperlukan untuk setiap AWS tindakan, termasuk tindakan Amazon DocumentDB. Simulator kebijakan IAM menentukan izin yang diperlukan untuk setiap tindakan yang Anda tentukan. Untuk informasi tentang simulator kebijakan IAM, lihat Menguji Kebijakan IAM dengan Simulator Kebijakan IAM di Panduan Pengguna IAM.

> **Note**
>
> Untuk menentukan tindakan, gunakan awalan `rds:` yang diikuti dengan nama operasi API (sebagai contoh, `rds:CreateDBInstance`).

Berikut ini daftar operasi Amazon RDS API dan tindakan, sumber daya, dan kunci syarat yang berkaitan.

Topik

- Tindakan Amazon DocumentDB yang mendukung izin tingkat sumber daya
- Tindakan Amazon DocumentDB yang tidak mendukung izin tingkat sumber daya

Tindakan Amazon DocumentDB yang mendukung izin tingkat sumber daya

Resource-level izin memberikan kemampuan untuk menentukan sumber daya di mana pengguna diizinkan untuk melakukan tindakan. Amazon DocumentDB memiliki dukungan parsial untuk izin tingkat sumber daya. Artinya untuk tindakan Amazon DocumentDB tertentu, Anda dapat mengontrol kapan pengguna diizinkan untuk menggunakan tindakan tersebut berdasarkan syarat yang harus dipenuhi, atau sumber daya tertentu yang diizinkan untuk digunakan oleh pengguna. Sebagai contoh, Anda dapat memberikan izin kepada pengguna untuk memodifikasi hanya instans tertentu.

Berikut ini daftar operasi Amazon DocumentDB API dan tindakan, sumber daya, dan kunci syarat miliknya yang berkaitan.

> **Note**
>
> Untuk fitur manajemen tertentu, Amazon DocumentDB menggunakan teknologi operasional yang dibagi dengan Amazon RDS. Untuk tindakan dan izin Amazon DocumentDB lainnya, lihat Tindakan, sumber daya, dan kunci kondisi untuk Amazon RDS di Referensi Otorisasi Layanan.

Operasi dan Tindakan Amazon DocumentDB API	Sumber daya	Kunci Kondisi
AddTagsToResource `rds:AddTagsToResource`	Instans `arn:aws:rds:` *`region`*`:`*`account-id`*` :db:`*`db-instance-name`*	`rds:db-tag`
	Grup subnet `arn:aws:rds:` *`region`*`:`*`account-id`*` :subgrp:`*`subnet-group-name`*	`rds:subgrp-tag`
ApplyPendingMaintenanceAction `rds:ApplyPendingMaintenanceAction`	Instans `arn:aws:rds:` *`region`*`:`*`account-id`*` :db:`*`db-instance-name`*	`rds:db-tag`
CopyDBClusterSnapshot `rds:CopyDBClusterSnapshot`	Snapshot klaster `arn:aws:rds:` *`region`*`:`*`account-id`*` :cluster-snapshot:`*`cluster-snapshot-name`*	`rds:cluster-snapshot-tag`
CreateDBCluster	Kluster	`rds:cluster-tag`

Operasi dan Tindakan Amazon DocumentDB API	Sumber daya	Kunci Kondisi
rds:CreateDBCluster	arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-name*	
	Grup parameter klaster arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
	Grup subnet arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
CreateDBClusterParameterGroup rds:CreateDBClusterParameterGroup	Grup parameter klaster arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
CreateDBClusterSnapshot rds:CreateDBClusterSnapshot	Kluster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-name*	rds:cluster-tag
	Snapshot klaster arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag

Operasi dan Tindakan Amazon DocumentDB API	Sumber daya	Kunci Kondisi
CreateDBInstance rds:CreateDBInstance	Instans arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:DatabaseClass rds:db-tag
	Kluster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-name*	rds:cluster-tag
CreateDBSubnetGroup rds:CreateDBSubnetGroup	Grup subnet arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
DeleteDBInstance rds:DeleteDBInstance	Instans arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
DeleteDBSubnetGroup rds:DeleteDBSubnetGroup	Grup subnet arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
DescribeDBClusterParameterGroups rds:DescribeDBClusterParameterGroups	Grup parameter klaster arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag

Operasi dan Tindakan Amazon DocumentDB API	Sumber daya	Kunci Kondisi
DescribeDBClusterParameters rds:DescribeDBClusterParameters	Grup parameter klaster arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
DescribeDBClusters rds:DescribeDBClusters	Kluster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-instance-name*	rds:cluster-tag
DescribeDBClusterSnapshotAttributes rds:DescribeDBClusterSnapshotAttributes	Snapshot klaster arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag
DescribeDBSubnetGroups rds:DescribeDBSubnetGroups	Grup subnet arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
DescribePendingMaintenanceActions rds:DescribePendingMaintenanceActions	Instans arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:DatabaseClass rds:db-tag

Operasi dan Tindakan Amazon DocumentDB API	Sumber daya	Kunci Kondisi
FailOverdbCluster rds:FailoverDBCluster	Kluster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-instance-name*	rds:cluster-tag
ListTagsForResource rds:ListTagsForResource	Instans arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
	Grup subnet arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
ModifyDBCluster rds:ModifyDBCluster	Kluster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-name*	rds:cluster-tag
	Grup parameter klaster arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag
ModifyDBClusterParameterGroup rds:ModifyDBClusterParameterGroup	Grup parameter klaster arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag

Operasi dan Tindakan Amazon DocumentDB API	Sumber daya	Kunci Kondisi
ModifyDBClusterSnapshotAttribute rds:ModifyDBClusterSnapshotAttribute	Snapshot klaster arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag
ModifyDBInstance rds:ModifyDBInstance	Instans arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:DatabaseClass rds:db-tag
RebootdBInstance rds:RebootDBInstance	Instans arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
RemoveTagsFromResource rds:RemoveTagsFromResource	Instans arn:aws:rds: *region*:*account-id* :db:*db-instance-name*	rds:db-tag
	Grup subnet arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag
ResetDBClusterParameterGroup rds:ResetDBClusterParameterGroup	Grup parameter klaster arn:aws:rds: *region*:*account-id* :cluster-pg: *cluster-parameter-group-name*	rds:cluster-pg-tag

Operasi dan Tindakan Amazon DocumentDB API	Sumber daya	Kunci Kondisi
RestoreDBClusterFromSnapshot rds:RestoreDBClusterFromSnapshot	Kluster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-instance-name*	rds:cluster-tag
	Snapshot klaster arn:aws:rds: *region*:*account-id* :cluster-snapshot: *cluster-snapshot-name*	rds:cluster-snapshot-tag
RestoreDBClusterToPointInTime rds:RestoreDBClusterToPointInTime	Kluster arn:aws:rds: *region*:*account-id* :cluster: *db-cluster-instance-name*	rds:cluster-tag
	Grup subnet arn:aws:rds: *region*:*account-id* :subgrp:*subnet-group-name*	rds:subgrp-tag

Tindakan Amazon DocumentDB yang tidak mendukung izin tingkat sumber daya

Anda dapat menggunakan semua tindakan Amazon DocumentDB dalam kebijakan IAM untuk memberikan maupun menolak izin pengguna untuk menggunakan tindakan tersebut. Namun demikian, tidak semua tindakan Amazon DocumentDB mendukung izin tingkat sumber daya, yang memungkinkan Anda untuk menentukan sumber daya di mana tindakan dapat dilakukan. Tindakan Amazon DocumentDB API berikut ini saat ini tidak mendukung izin tingkat sumber daya. Oleh karena itu, untuk menggunakan tindakan tersebut dalam kebijakan IAM, Anda harus memberikan izin pengguna untuk menggunakan semua sumber daya untuk tindakan dengan menggunakan wildcard * untuk elemen `Resource` dalam pernyataan Anda.

- `rds:DescribeDBClusterSnapshots`
- `rds:DescribeDBInstances`

Cross-service pencegahan wakil bingung

Masalah "confused deputy" adalah masalah keamanan di mana entitas yang tidak memiliki izin untuk melakukan tindakan dapat memengaruhi entitas yang memiliki hak akses lebih tinggi untuk melakukan tindakan. Pada tahun AWS, peniruan lintas layanan dapat mengakibatkan masalah wakil yang membingungkan. Cross-service peniruan identitas dapat terjadi ketika satu layanan (layanan panggilan) memanggil layanan lain (layanan yang disebut). Layanan pemanggilan dapat dimanipulasi menggunakan izinnya untuk bertindak pada sumber daya pelanggan lain dengan cara yang seharusnya tidak dilakukannya kecuali bila memiliki izin untuk mengakses. Untuk mencegah hal ini, AWS menyediakan alat yang membantu Anda melindungi data untuk semua layanan dengan principal layanan yang telah diberi akses ke sumber daya di akun Anda.

Sebaiknya gunakan kunci konteks kondisi `aws:SourceAccount`global `aws:SourceArn`dan global dalam kebijakan sumber daya untuk membatasi izin yang diberikan Amazon DocumentDB layanan lain ke sumber daya. Gunakan `aws:SourceArn` jika Anda ingin hanya satu sumber daya yang akan dikaitkan dengan akses lintas layanan. Gunakan `aws:SourceAccount` jika Anda ingin mengizinkan sumber daya apa pun di akun tersebut dikaitkan dengan penggunaan lintas layanan.

Cara paling efektif untuk melindungi dari masalah "confused deputy" adalah dengan menggunakan kunci konteks kondisi global `aws:SourceArn` dengan ARN lengkap sumber daya. Jika Anda tidak mengetahui ARN lengkap sumber daya atau jika Anda menentukan beberapa sumber daya, gunakan kunci kondisi konteks global `aws:SourceArn` dengan karakter wildcard (*) untuk bagian ARN yang tidak diketahui. Misalnya, `arn:aws:`*`servicename`*`:*:`*`123456789012`*`:*`.

Jika nilai `aws:SourceArn` tidak berisi ID akun, seperti ARN bucket Amazon S3, Anda harus menggunakan kedua kunci konteks kondisi global tersebut untuk membatasi izin.

Nilai `aws:SourceArn` harus cluster.

Contoh berikut menunjukkan bagaimana Anda dapat menggunakan `aws:SourceArn` dan kunci konteks kondisi `aws:SourceAccount` global di Amazon DocumentDB untuk mencegah masalah wakil yang membingungkan.

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

Otentikasi menggunakan identitas IAM

Pengguna dan aplikasi Amazon DocumentDB dapat menggunakan pengguna dan peran IAM untuk mengautentikasi ke dalam cluster Amazon DocumentDB. Autentikasi Amazon DocumentDB IAM adalah metode otentikasi tanpa kata sandi. Selain itu, aplikasi klien tidak mengirim rahasia kata sandi ke cluster Amazon DocumentDB saat menggunakan IAM. roles/users Sebaliknya, koneksi klien diautentikasi dengan AWS STS menggunakan token keamanan sementara. Non-administrative pengguna dan aplikasi sekarang dapat menggunakan ARN identitas IAM yang sama saat menghubungkan ke cluster Amazon DocumentDB yang berbeda dan layanan lainnya. AWS

Anda juga dapat memilih untuk menggunakan autentikasi berbasis kata sandi dan IAM untuk mengautentikasi pengguna dan aplikasi ke cluster Amazon DocumentDB. Autentikasi IAM hanya tersedia di cluster berbasis instans Amazon DocumentDB versi 5.0. Autentikasi IAM menggunakan ARN identitas IAM tidak didukung untuk pengguna utama Amazon DocumentDB.

> **Note**
>
> Pengguna utama hanya dapat diautentikasi menggunakan otentikasi berbasis kata sandi yang ada.

Topik

- Memulai otentikasi menggunakan pengguna dan peran IAM
- Melakukan konfigurasi AWS menghitung jenis untuk mengautentikasi ke Amazon DocumentDB menggunakan AWS IAM
- Memantau permintaan otentikasi IAM
- Menggunakan otentikasi IAM
- Driver yang mendukung IAM
- FAQ otentikasi identitas IAM

Memulai otentikasi menggunakan pengguna dan peran IAM

Pengguna Amazon DocumentDB dan peran dengan identitas IAM dibuat dan dikelola dalam database. `$external`

Membuat pengguna

Connect sebagai pengguna utama, lalu buat pengguna dan peran IAM:

```
use $external;
db.createUser(
    {
        user: "arn:aws:iam::123456789123:user/iamuser",
        mechanisms: ["MONGODB-AWS"],
        roles: [ { role: "readWrite", db: "readWriteDB" } ]
    }
);
```

Atau, tambahkan pengguna Amazon DocumentDB menggunakan peran IAM:

```
use $external;
db.createUser(
    {
        user: "arn:aws:iam::123456789123:role/iamrole",
        mechanisms: ["MONGODB-AWS"],
        roles: [ { role: "readWrite", db: "readWriteDB" } ]
    }
);
```

Memodifikasi pengguna atau peran IAM

Memodifikasi pengguna IAM yang ada:

```
use $external;
db.updateUser(
  "arn:aws:iam::123456789123:user/iamuser",
  {
    roles: [ { role: "read", db: "readDB" } ]
  }
);
```

Memodifikasi peran IAM yang ada:

```
use $external;
db.updateUser(
  "arn:aws:iam::123456789123:role/iamrole",
  {
    roles: [ { role: "read", db: "readDB" } ]
  }
);
```

Untuk memberikan atau mencabut peran dari pengguna 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" } ]
);
```

Untuk memberikan atau mencabut peran dari peran 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" } ]
);
```

Menjatuhkan pengguna atau peran IAM

Untuk menjatuhkan pengguna IAM yang ada:

```
use $external;
db.dropUser("arn:aws:iam::123456789123:user/iamuser");
```

Untuk menghapus peran IAM yang ada:

```
use $external;
db.dropUser("arn:aws:iam::123456789123:role/iamrole");
```

Konfigurasikan URI koneksi untuk mengautentikasi menggunakan AWS IAM

Untuk mengautentikasi menggunakan AWS IAM, gunakan parameter URI berikut: `authSource` as `$external` dan `authMechanism` as. `MONGODB-AWS` Jika Anda menggunakan pengguna IAM, bidang nama pengguna dan kata sandi masing-masing diganti dengan Kunci Akses dan Kunci Rahasia. Jika Anda mengasumsikan peran IAM, dilampirkan ke lingkungan tempat Anda berada (misalnya, AWS Lambda fungsi, instans Amazon EC2). Anda tidak perlu secara khusus memberikan kredensi apa pun saat mengautentikasi menggunakan mekanisme. `MONGODB-AWS` Jika Anda menggunakan driver MongoDB yang mendukung `MONGODB-AWS` mekanisme otentikasi, driver juga memiliki kemampuan untuk mengambil kredensyal peran IAM dari instance komputasi (misalnya, Amazon EC2, fungsi Lambda, dan lainnya). Contoh berikut menggunakan shell mongo untuk mengautentikasi menggunakan `MONGODB-AWS` dengan meneruskan Kunci Akses dan Kunci Rahasia (dari pengguna IAM) secara manual untuk menunjukkan otentikasi terhadap Amazon DocumentDB.

Contoh berikut menggunakan kode Python untuk mengautentikasi menggunakan `MONGODB-AWS` tanpa secara eksplisit meneruskan kredensyal apa pun (menggunakan Peran IAM yang dilampirkan ke lingkungan) untuk menunjukkan otentikasi terhadap 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')
```

Contoh berikut menggunakan shell mongo untuk mengautentikasi menggunakan `MONGODB-AWS` mekanisme dengan meneruskan Kunci Akses dan Kunci Rahasia (dari pengguna IAM) secara manual untuk menunjukkan otentikasi terhadap Amazon DocumentDB.

```
$ mongo 'mongodb://<access_key>:<secret_key>@<cluster_endpoint>:<db_port>/test?
authSource=%24external&authMechanism=MONGODB-AWS'
```

Contoh berikut menggunakan shell mongo untuk mengautentikasi penggunaan `MONGODB-AWS` tanpa secara eksplisit meneruskan kredensyal apa pun (menggunakan Peran IAM yang dilampirkan ke lingkungan) untuk menunjukkan otentikasi terhadap Amazon DocumentDB.

```
$ mongo 'mongodb://<cluster_endpoint>:<db_port>/test?authSource=
%24external&authMechanism=MONGODB-AWS'
```

Melakukan konfigurasi AWS menghitung jenis untuk mengautentikasi ke Amazon DocumentDB menggunakan AWS IAM

Menggunakan Amazon EC2//AWS LambdaAWS Fargate

Amazon EC2 menggunakan variabel lingkungan berikut. Jika Anda memiliki peran IAM yang dilampirkan ke instans EC2 atau peran IAM eksekusi yang terkait dengan fungsi Lambda atau tugas Amazon ECS, maka variabel-variabel ini akan diisi secara otomatis dan driver dapat mengambil nilai-nilai ini dari lingkungan:

```
AWS_ACCESS_KEY_ID
AWS_SECRET_ACCESS_KEY
AWS_SESSION_TOKEN
```

Untuk informasi selengkapnya tentang variabel lingkungan, lihat Menggunakan variabel lingkungan Lambda di Panduan AWS Lambda Pengembang.

Menggunakan Amazon EKS

Menetapkan peran ke pod Amazon Elastic Kubernetes Service (Amazon EKS) Anda akan secara otomatis menyiapkan dua variabel lingkungan berikut:

```
AWS_WEB_IDENTITY_TOKEN_FILE - path of web identity token file
AWS_ROLE_ARN - Name of IAM role to connect with
```

Dengan bantuan variabel-variabel ini, ambil peran secara manual dari kode Anda menggunakan panggilan AWS SDK untuk`AssumeRoleWithWebIdentity`:

- Hilangkan `ProviderID` parameternya.
- Temukan nilai `WebIdentityToken` parameter dalam file yang dijelaskan dalam variabel `AWS_WEB_IDENTITY_TOKEN_FILE` lingkungan.

Untuk info selengkapnya tentang Amazon EKS, lihat Apa itu Amazon EKS di Panduan Pengguna Amazon EKS.

Memantau permintaan otentikasi IAM

Menggunakan audit Amazon DocumentDB

Buka folder log audit di Amazon CloudWatch, dan gunakan pola pencarian yang berbeda untuk mendapatkan log untuk otentikasi IAM. Misalnya, gunakan `{ $.param.mechanism = "MONGODB-AWS" }` sebagai pola pencarian untuk “Cari semua aliran log”.

Untuk info selengkapnya tentang peristiwa yang didukung dalam audit, lihatMengaudit acara Amazon DocumentDB.

Menggunakan CloudWatch metrik Amazon

`StsGetCallerIdentityCalls`: Metrik ini menunjukkan berapa banyak `GetCallerIdentity` panggilan yang dilakukan instance Amazon DocumentDB ke endpoint AWS Security Token Service AWS STS regionalized (). Lihat spesifikasi `MONGODB-AWS` otentikasi tentang mengapa instance database perlu melakukan panggilan STS`GetCallerIdentity`.

Menggunakan otentikasi IAM

Jika Anda tidak ingin mengelola nama pengguna dan kata sandi di database Anda sendiri, Anda dapat menggunakan otentikasi IAM. Autentikasi IAM hanya tersedia di cluster berbasis instans Amazon DocumentDB versi 5.0.

Autentikasi IAM memiliki ketergantungan pada layanan STS. Kami menyarankan Anda mengevaluasi apakah Anda dapat menurunkan tingkat koneksi Anda ketika Anda menggunakan otentikasi IAM untuk koneksi dan mendapatkan pengecualian pelambatan STS.

Untuk kuota IAM, lihat IAM dan AWS STS kuota di Panduan Pengguna IAM.

Driver yang mendukung IAM

Driver yang mendukung Amazon DocumentDB 5.0 dan mekanisme otentikasi harus bekerja dengan `MONGODB-AWS` implementasi autentikasi IAM di Amazon DocumentDB.

> **Important**
>
> Ada batasan yang diketahui dengan Node.js driver yang lebih lama dari versi 6.13.1, yang saat ini tidak didukung oleh otentikasi identitas IAM untuk Amazon DocumentDB. Node.js driver dan alat yang menggunakan Node.js driver (misalnya, mongosh) harus ditingkatkan untuk menggunakan Node.js driver versi 6.13.1 atau lebih tinggi.

FAQ otentikasi identitas IAM

Apakah ada sampel yang bisa saya rujuk?

Lihat halaman ini untuk contoh kasus penggunaan dan konfigurasi:

- Bagaimana pengguna manusia dapat mengautentikasi ke Amazon DocumentDB menggunakan Pengguna IAM dan Peran IAM
- Password-less otentikasi ke Amazon DocumentDB menggunakan Peran IAM

Saya mendapatkan kesalahan saat menggunakan driver Python saya: “pymongo.errors. ConfigurationError: MONGODB-AWS otentikasi membutuhkan pymongo-auth-aws”. Bagaimana saya bisa menyelesaikan ini?

Pastikan Anda menggunakan pernyataan berikut saat menginstal driver Python dengan otentikasi IAM:

```
pip install 'pymongo[aws]'
```

Ini akan menginstal AWS dependensi tambahan yang diperlukan agar otentikasi IAM berfungsi.

Apakah koneksi saya akan terputus ketika kredensyal sementara peran IAM saya kedaluwarsa?

Tidak, kredensi IAM sementara hanya digunakan untuk membuat koneksi dan otentikasi. Kemudian semua otorisasi lebih lanjut terjadi di cluster Amazon DocumentDB. Bahkan jika kredensyal IAM rotate/expire, koneksi tidak akan jatuh atau basi.

Mengelola pengguna Amazon DocumentDB

Di Amazon DocumentDB, pengguna mengautentikasi ke klaster bersama dengan kata sandi. Setiap klaster memiliki kredenal masuk utama yang dibuat selama pembuatan klaster.

> **Note**
>
> Semua pengguna baru yang dibuat sebelum 26 Maret 2020 telah diberikan peran `dbAdminAnyDatabase`, `readWriteAnyDatabase`, dan `clusterAdmin`. Sebaiknya Anda mengevaluasi kembali semua pengguna dan mengubah peran seperlunya untuk menerapkan hak istimewa paling sedikit untuk semua pengguna di klaster Anda.
> Untuk informasi selengkapnya, lihat Akses database menggunakan Role-Based Access Control.

Pengguna utama dan **`serviceadmin`**

Cluster Amazon DocumentDB yang baru dibuat memiliki dua pengguna: pengguna utama dan pengguna. `serviceadmin`

Pengguna utama adalah pengguna tunggal yang memiliki hak istimewa yang dapat melakukan tugas administratif dan membuat pengguna tambahan dengan peran. Saat Anda tersambung ke klaster Amazon DocumentDB untuk pertama kalinya, Anda harus mengautentikasi menggunakan kredensyal masuk utama. Pengguna utama menerima izin administratif ini untuk klaster Amazon DocumentDB saat klaster tersebut dibuat, dan diberi peran. `root`

Pengguna `serviceadmin` dibuat secara implisit saat klaster dibuat. Setiap cluster Amazon DocumentDB memiliki pengguna `serviceadmin` yang AWS menyediakan kemampuan untuk mengelola cluster Anda. Anda tidak dapat masuk sebagai, menjatuhkan, mengganti nama, mengubah kata sandi, atau mengubah izin untuk `serviceadmin`. Setiap upaya untuk melakukannya menghasilkan kesalahan.

> **Note**
>
> Primer dan `serviceadmin` pengguna untuk klaster Amazon DocumentDB tidak dapat dihapus dan peran `root` untuk pengguna utama tidak dapat dicabut.
> Jika Anda lupa kata sandi pengguna utama Anda, Anda dapat mengatur ulang menggunakan Konsol Manajemen AWS atau AWS CLI.

Membuat pengguna tambahan

Setelah Anda terhubung sebagai pengguna utama (atau pengguna mana pun yang memiliki perancreateUser), Anda dapat membuat pengguna baru, seperti yang ditunjukkan di bawah ini.

```
db.createUser(
    {
        user: "sample-user-1",
        pwd: "password123",
        roles:
            [{"db":"admin", "role":"dbAdminAnyDatabase" }]
    }
)
```

Untuk melihat detail pengguna, Anda dapat menggunakan perintah `show users` sebagai berikut. Anda juga dapat menghapus pengguna dengan perintah `dropUser`. Untuk informasi selengkapnya, lihat Perintah umum.

```
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"
     }
    ]
}
```

Dalam contoh di atas, pengguna baru `sample-user-1` dikaitkan dengan basis data `admin`. Hal ini selalu terjadi bagi pengguna baru. Amazon DocumentDB tidak memiliki konsep `authenticationDatabase` dan dengan demikian semua autentikasi dilakukan dalam konteks basis data `admin`.

Saat membuat pengguna, jika Anda menghilangkan bidang db saat menentukan peran, Amazon DocumentDB akan secara implisit mengaitkan peran tersebut ke basis data tempat koneksi diterbitkan. Misalnya, jika koneksi Anda dikeluarkan terhadap basis data `sample-database` dan Anda menjalankan perintah berikut, pengguna `sample-user-2` akan dibuat di basis data `admin` dan akan memiliki izin `readWrite` ke basis data `sample-database`.

```
db.createUser(
    {
        user: "sample-user-2",
        pwd: "password123",
        roles:
            ["readWrite"]
```

```
    }
)
```

Membuat pengguna dengan peran yang dicakup di semua basis data (misalnya, `readInAnyDatabase`) mengharuskan Anda berada dalam konteks basis data `admin` saat membuat pengguna atau Anda secara eksplisit menyatakan basis data untuk peran saat membuat pengguna.

Untuk mengganti konteks dari basis data Anda, Anda dapat menggunakan perintah berikut.

```
use admin
```

Untuk mempelajari selengkapnya tentang Kontrol Akses Berbasis Peran dan menerapkan hak istimewa paling rendah di antara pengguna di klaster Anda, lihat Akses database menggunakan Role-Based Access Control.

Secara otomatis memutar kata sandi untuk Amazon DocumentDB

Dengan AWS Secrets Manager, Anda dapat mengganti kredensi hardcode dalam kode Anda (termasuk kata sandi) dengan panggilan API ke Secrets Manager untuk mengambil rahasia secara terprogram. Ini membantu memastikan bahwa rahasia tidak dapat dikompromikan oleh seseorang yang memeriksa kode Anda, karena rahasianya tidak ada di sana. Selain itu, Anda dapat mengonfigurasi Secrets Manager untuk merotasi rahasia secara otomatis sesuai dengan jadwal yang Anda tentukan. Ini memungkinkan Anda mengganti rahasia jangka panjang dengan rahasia jangka pendek, yang membantu mengurangi risiko kompromi secara signifikan.

Menggunakan Secrets Manager, Anda dapat secara otomatis memutar kata sandi Amazon DocumentDB Anda (yaitu, rahasia) menggunakan fungsi AWS Lambda yang disediakan Secrets Manager.

Untuk informasi selengkapnya tentang AWS Secrets Manager dan integrasi asli dengan Amazon DocumentDB, lihat berikut ini:

- Blog: Cara memutar kredensyal Amazon DocumentDB dan Amazon Redshift di Secrets Manager AWS
- Apa itu AWS Secrets Manager?
- Merotasi rahasia AWS Secrets Manager
- Kredensyal Amazon DocumentDB di Secrets Manager

Akses database menggunakan Role-Based Access Control

Anda dapat membatasi akses ke tindakan yang dapat dilakukan pengguna pada basis data menggunakan kontrol akses berbasis peran (RBAC) di Amazon DocumentDB (dengan kompatibilitas MongoDB). RBAC bekerja dengan memberikan satu atau lebih peran kepada pengguna. Peran ini menentukan operasi yang dapat dilakukan pengguna pada sumber daya database. Amazon DocumentDB saat ini mendukung kedua peran bawaan yang dicakup pada tingkat database, `read` seperti,,, `readWrite` `readAnyDatabase``clusterAdmin`, dan peran yang ditentukan pengguna yang dapat dicakup untuk tindakan tertentu dan sumber daya terperinci seperti koleksi berdasarkan kebutuhan Anda.

Kasus penggunaan umum untuk RBAC termasuk menerapkan hak istimewa paling sedikit dengan membuat pengguna dengan akses hanya-baca ke database atau koleksi dalam cluster, dan desain aplikasi multi-penyewa yang memungkinkan satu pengguna untuk mengakses database atau koleksi tertentu dalam sebuah cluster.

> **Note**
>
> Semua pengguna baru yang dibuat sebelum 26 Maret 2020 telah diberikan peran `dbAdminAnyDatabase`, `readWriteAnyDatabase`, dan `clusterAdmin`. Disarankan bahwa Anda mengevaluasi kembali semua pengguna yang ada dan mengubah peran seperlunya untuk menerapkan hak istimewa terendah untuk klaster Anda.

Topik

- Konsep RBAC
- Memulai dengan peran bawaan RBAC
- Memulai dengan peran yang ditentukan pengguna RBAC
- Menghubungkan ke Amazon DocumentDB sebagai Pengguna
- Perintah umum
- Perbedaan fungsional
- Batas
- Akses database menggunakan Role-Based Access Control

Konsep RBAC

Berikut ini adalah istilah dan konsep penting yang berkaitan dengan kontrol akses berbasis peran. Untuk informasi lebih lanjut tentang pengguna Amazon DocumentDB, lihat Mengelola pengguna Amazon DocumentDB.

- Pengguna — Entitas individu yang dapat mengautentikasi ke basis data dan melakukan operasi.
- Kata Sandi — Rahasia yang digunakan untuk mengautentikasi pengguna.
- Peran — Mengotorisasi pengguna untuk melakukan tindakan pada satu atau lebih basis data.
- Basis Data Admin — Basis data di mana pengguna disimpan dan diberi otorisasi terhadapnya.
- Basis Data (**db**) — Namespace di dalam klaster yang berisi koleksi untuk menyimpan dokumen.

Perintah berikut ini membuat pengguna yang bernama `sample-user`.

```
db.createUser({user: "sample-user", pwd: "abc123", roles: [{role: "read", db: "sample-database"}]})
```

Dalam contoh ini:

- `user: "sample-user"` — Mengindikasikan nama pengguna.
- `pwd: "abc123"` — Mengindikasikan kata sandi pengguna.
- `role: "read", "db: "sample-database"` — Mengindikasikan bahwa pengguna `sample-user` akan memiliki izin baca di `sample-database`.



Contoh berikut ini menunjukkan keluaram setelah Anda mendapatkan `sample-user` pengguna dengan `db.getUser(sample-user)`. Dalam contoh ini, `sample-user` pengguna berada di basis data `admin` tetapi memiliki peran baca untuk `sample-database` basis data.



Saat membuat pengguna, jika Anda menghilangkan bidang db saat menentukan peran, Amazon DocumentDB akan secara implisit mengaitkan peran tersebut ke basis data tempat koneksi diterbitkan. Sebagai contoh, jika koneksi Anda dikeluarkan terhadap `sample-database` basis data dan Anda menjalankan perintah berikut ini, `sample-user` pengguna akan dibuat dalam basis data `admin` dan akan memiliki izin `readWrite` ke `sample-database` basis data.

```
db.createUser({user: "sample-user", pwd: "abc123", roles: ["readWrite"]})
```

Keluaran dari operasi ini terlihat seperti berikut ini.

```
{
   "user":"sample-user",
   "roles":[
      {
         "db":"sample-database",
         "role":"readWrite"
      }
   ]
}
```

Membuat pengguna dengan peran yang tercakup di semua basis data (sebagai contoh, `readAnyDatabase`) mengharuskan Anda untuk berada dalam konteks basis data `admin` saat membuat pengguna, maupun Anda secara eksplisit menyatakan basis data untuk peran saat membuat pengguna. Untuk mengeluarkan perintah terhadap basis data `admin`, Anda dapat menggunakan `use admin` perintah. Untuk informasi selengkapnya, lihat Perintah umum.

Memulai dengan peran bawaan RBAC

Untuk membantu Anda memulai kontrol akses berbasis peran, bagian ini memandu Anda melalui skenario contoh untuk menerapkan hak istimewa terendah dengan membuat peran untuk tiga pengguna dengan fungsi pekerjaan yang berbeda.

- `user1` adalah manajer baru yang perlu dapat melihat dan mengakses semua basis data dalam klaster.
- `user2` adalah karyawan baru yang memerlukan akses hanya ke satu basis data, `sample-database-1`, di klaster yang sama.
- `user3` adalah karyawan yang sudah ada yang perlu melihat dan mengakses basis data yang berbeda, `sample-database-2` yang tidak mereka miliki aksesnya sebelumnya, dalam klaster yang sama.

Pada suatu saat, keduanya `user1` dan `user2` meninggalkan perusahaan dan akses mereka harus dicabut.

Untuk membuat pengguna dan memberikan peran, pengguna yang Anda autentikasi ke klaster harus memiliki peran yang berkaitan yang dapat melakukan tindakan untuk `createUser` dan `grantRole`. Sebagai contoh, peran `admin` dan `userAdminAnyDatabase` keduanya dapat memberikan kemampuan semacam itu, sebagai contoh. Untuk tindakan sesuai peran, lihat Akses database menggunakan Role-Based Access Control.

> **Note**
>
> Di Amazon DocumentDB, semua pengguna dan operasi peran (sebagai contoh, `create`, `get`, `drop`, `grant`, `revoke`, dll.) secara implisit dilakukan dalam basis data `admin` baik Anda mengeluarkan perintah terhadap basis data `admin` ataupun tidak.

Pertama, untuk memahami siapa pengguna saat ini dan perannya dalam klaster, Anda dapat menjalankan perintah `show users`, sebagaimana dalam contoh berikut. Anda akan melihat dua pengguna, `serviceadmin` dan pengguna utama untuk cluster. Kedua pengguna tersebut selalu ada dan tidak dapat dihapus. Untuk informasi selengkapnya, lihat Mengelola pengguna Amazon DocumentDB.

```
show users
```

Untuk `user1`, buat peran dengan akses baca dan tulis untuk semua basis data di seluruh klaster dengan perintah berikut ini.

```
db.createUser({user: "user1", pwd: "abc123", roles: [{role: "readWriteAnyDatabase", db:
 "admin"}]})
```

Keluaran dari operasi ini terlihat seperti berikut ini.

```
{
   "user":"user1",
   "roles":[
      {
         "role":"readWriteAnyDatabase",
         "db":"admin"
      }
   ]
}
```

Untuk `user2`, buat peran dengan akses hanya-baca ke `sample-database-1` basis data dengan perintah berikut ini.

```
db.createUser({user: "user2", pwd: "abc123", roles: [{role: "read", db: "sample-
database-1"}]})
```

Keluaran dari operasi ini terlihat seperti berikut ini.

```
{
   "user":"user2",
   "roles":[
      {
         "role":"read",
         "db":"sample-database-1"
      }
   ]
}
```

Untuk menyimulasikan skenario bahwa `user3` adalah pengguna yang sudah ada, pertama buat `user3` pengguna, dan kemudian tetapkan peran baru ke `user3`.

```
db.createUser({user: "user3", pwd: "abc123", roles: [{role: "readWrite", db: "sample-
database-1"}]})
```

Keluaran dari operasi ini terlihat seperti berikut ini.

```
{
   "user":"user3",
   "roles":[
      {
         "role":"readWrite",
         "db":"sample-database-1"
      }
   ]
}
```

Sekarang setelah `user3` pengguna telah dibuat, tetapkan `user3` peran `read` ke `sample-database-2`.

```
db.grantRolesToUser("user3", [{role: "read", db: "sample-database-2"}])
```

Terakhir, `user1` dan `user2` meninggalkan perusahaan dan akses mereka ke klaster harus dicabut. Anda bisa melakukannya dengan menghapus pengguna, sebagai berikut.

```
db.dropUser("user1")
db.dropUser("user2")
```

Untuk memastikan bahwa semua pengguna memiliki peran yang tepat, Anda dapat mendaftar semua pengguna dengan perintah berikut ini.

```
show users
```

Output dari operasi ini akan terlihat seperti berikut.

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

Memulai dengan peran yang ditentukan pengguna RBAC

Untuk membantu Anda memulai dengan peran yang ditentukan pengguna, bagian ini memandu Anda melalui skenario contoh penerapan hak istimewa paling sedikit dengan membuat peran untuk tiga pengguna dengan fungsi pekerjaan yang berbeda.

Dalam contoh ini, berikut ini berlaku:

- `user1` adalah manajer baru yang perlu dapat melihat dan mengakses semua basis data dalam klaster.

- `user2`adalah karyawan baru yang hanya membutuhkan tindakan 'temukan' untuk hanya satu database,`sample-database-1`, di cluster yang sama.
- `user3`adalah karyawan yang sudah ada yang perlu melihat dan mengakses koleksi tertentu, col2 dalam database yang berbeda, `sample-database-2` yang sebelumnya tidak mereka akses, di cluster yang sama.
- Untuk `user1`, buat peran dengan akses baca dan tulis untuk semua basis data di seluruh klaster dengan perintah berikut ini.

```
db.createUser(
 {
     user: "user1", pwd: "abc123",
     roles: [{role: "readWriteAnyDatabase", db: "admin"}]
 }
 )
```

Output dari operasi ini akan terlihat seperti berikut.

```
{
   "user":"user1",
   "roles":[
      {
         "role":"readWriteAnyDatabase",
         "db":"admin"
      }
   ]
}
```

Untuk`user2`, buat peran dengan hak istimewa 'temukan' untuk semua koleksi dalam database `sample-database-1` dengan perintah berikut. Perhatikan bahwa peran ini akan memastikan bahwa setiap pengguna terkait hanya dapat menjalankan kueri find.

```
db.createRole(
{
    role: "findRole",
    privileges: [
    {
        resource: {db: "sample-database-1", collection: ""}, actions: ["find"]
    }],
    roles: []
```

```
}
)
```

Output dari operasi ini akan terlihat seperti berikut.

```
{
   "role":"findRole",
   "privileges":[
      {
         "resource":{
            "db":"sample-database-1",
            "collection":""
         },
         "actions":[
            "find"
         ]
      }
   ],
   "roles":[

   ]
}
```

Selanjutnya, buat user (`user2`) dan lampirkan peran yang baru dibuat `findRole` ke pengguna.

```
db.createUser(
{
    user: "user2",
    pwd: "abc123",
    roles: []
})

 db.grantRolesToUser("user2",["findRole"])
```

Untuk mensimulasikan skenario yang `user3` merupakan pengguna yang sudah ada, pertama buat penggunauser3, dan kemudian buat peran baru yang disebut CollectionRole yang akan kita lakukan pada langkah berikutnya. `user3`

Sekarang Anda dapat menetapkan peran baru. `user3` Peran baru ini akan memungkinkan `user3` untuk dapat menyisipkan, memperbarui, menghapus, dan menemukan akses ke satu koleksi tertentu col2 di`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: []
}
)
```

Output dari operasi ini akan terlihat seperti berikut.

```
{
   "role":"collectionRole",
   "privileges":[
      {
         "resource":{
            "db":"sample-database-2",
            "collection":"col2"
         },
         "actions":[
            "find",
            "update",
            "insert",
            "remove"
         ]
      }
   ],
   "roles":[

   ]
}
```

Sekarang pengguna `user3` telah dibuat, Anda dapat memberikan `user3` perancollectionFind.

```
db.grantRolesToUser("user3",["collectionRole"])
```

Terakhir, `user1` dan `user2` meninggalkan perusahaan dan akses mereka ke klaster harus dicabut. Anda bisa melakukannya dengan menghapus pengguna, sebagai berikut.

```
db.dropUser("user1")
db.dropUser("user2")
```

Untuk memastikan bahwa semua pengguna memiliki peran yang tepat, Anda dapat mendaftar semua pengguna dengan perintah berikut ini.

```
show users
```

Keluaran dari operasi ini terlihat seperti berikut ini.

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

Menghubungkan ke Amazon DocumentDB sebagai Pengguna

Saat menghubungkan ke klaster Amazon DocumentDB, Anda terhubung dalam konteks basis data tertentu. Secara default, jika Anda tidak menentukan basis data dalam string koneksi Anda, Anda secara otomatis terhubung ke klaster dalam konteks basis data `test`. Semua perintah tingkat koleksi seperti `insert` dan `find` dikeluarkan terhadap koleksi di basis data `test`.

Untuk melihat basis data dalam konteks Anda dari atau — dengan kata lain — mengeluarkan perintah padanya, gunakan perintah db dalam mongo shell, sebagai berikut.

Kueri:

```
db
```

Output:

```
test
```

Meskipun koneksi default mungkin dalam konteks basis data `test`, hal ini tidak selalu berarti bahwa pengguna yang berkaitan dengan koneksi diotorisasi untuk melakukan tindakan pada basis data `test`. Dalam skenario contoh sebelumnya, jika Anda mengautentikasi sebagai `user3` pengguna, yang memiliki peran `readWrite` untuk basis data `sample-database-1`, konteks default koneksi Anda adalah basis data `test`. Namun demikian, jika Anda mencoba untuk memasukkan dokumen ke koleksi pada basis data `test`, Anda akan menerima pesan kesalahan Kegagalan otorisasi. Hal ini karena pengguna tersebut tidak diotorisasi untuk melakukan perintah tersebut pada database tersebut, sebagaimana ditunjukkan di bawah ini.

Kueri:

```
db
```

Output:

```
test
```

Kueri:

```
db.col.insert({x:1})
```

Output:

```
WriteCommandError({ "ok" : 0, "code" : 13, "errmsg" : "Authorization failure" })
```

Jika Anda mengubah konteks koneksi Anda ke basis data `sample-database-1`, Anda dapat menulis ke koleksi di mana pengguna memiliki otorisasi untuk melakukannya.

Kueri:

```
use sample-database-1
```

Output:

```
switched to db sample-database-1
```

Kueri:

```
db.col.insert({x:1})
```

Output:

```
WriteResult({ "nInserted" : 1})
```

Ketika Anda mengautentikasi ke klaster dengan pengguna tertentu, Anda juga dapat menentukan basis data dalam string koneksi. Dengan melakukannya akan menghilangkan kebutuhan untuk melakukan perintah `use` setelah pengguna telah diautentikasi ke basis data `admin`.

String koneksi berikut mengautentikasi pengguna terhadap basis data `admin`, tetapi konteks koneksi akan melawan basis data `sample-database-1`.

```
mongo "mongodb://user3:abc123@sample-cluster.node.us-east-1.docdb.amazonaws.com:27017/
sample-database-2"
```

Perintah umum

Bagian ini menyediakan contoh perintah umum menggunakan kontrol akses berbasis peran di Amazon DocumentDB. Anda harus berada dalam konteks basis data `admin` untuk membuat dan memodifikasi pengguna dan peran. Anda dapat menggunakan perintah `use admin` untuk beralih ke basis data `admin`.

> **Note**
>
> Modifikasi untuk pengguna dan peran secara implisit akan terjadi di basis data `admin`. Membuat pengguna dengan peran yang tercakup di semua basis data (sebagai contoh, `readAnyDatabase`) mensyaratkan bahwa Anda dalam konteks basis data `admin` (yaitu, `use admin`) saat membuat pengguna, maupun Anda secara eksplisit menyatakan basis data untuk peran saat membuat pengguna (sebagaimana ditunjukkan dalam Contoh 2 di bagian ini).

Contoh 1: Buat pengguna dengan `read` peran untuk database`foo`.

```
db.createUser({user: "readInFooBar", pwd: "abc123", roles: [{role: "read", db:
 "foo"}]})
```

Keluaran dari operasi ini terlihat seperti berikut ini.

```
{
   "user":"readInFooBar",
   "roles":[
      {
         "role":"read",
         "db":"foo"
      }
   ]
}
```

Contoh 2: Buat pengguna dengan akses baca di semua database.

```
db.createUser({user: "readAllDBs", pwd: "abc123", roles: [{role: "readAnyDatabase", db:
 "admin"}]})
```

Keluaran dari operasi ini terlihat seperti berikut ini.

```
{
   "user":"readAllDBs",
   "roles":[
      {
         "role":"readAnyDatabase",
         "db":"admin"
      }
   ]
}
```

Contoh 3: Berikan `read` peran kepada pengguna yang sudah ada di database baru.

```
db.grantRolesToUser("readInFooBar", [{role: "read", db: "bar"}])
```

Contoh 4: Perbarui peran pengguna.

```
db.updateUser("readInFooBar", {roles: [{role: "read", db: "foo"}, {role: "read", db:
 "baz"}]})
```

Contoh 5: Mencabut akses ke database untuk pengguna.

```
db.revokeRolesFromUser("readInFooBar", [{role: "read", db: "baz"}])
```

Contoh 6: Jelaskan peran bawaan.

```
db.getRole("read", {showPrivileges:true})
```

Keluaran dari operasi ini terlihat seperti berikut ini.

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

Contoh 7: Jatuhkan pengguna dari cluster.

```
db.dropUser("readInFooBar")
```

Output dari operasi ini akan terlihat seperti berikut.

```
true
```

Contoh 8: Buat peran dengan akses baca dan tulis ke koleksi tertentu

```
db.createRole(
{
    role: "collectionRole",
    privileges: [
    {
        resource: {db: "sample-database-2", collection: "col2"}, actions: ["find",
 "update", "insert", "remove"]
    }],
    roles: []
}
)
```

Output dari operasi ini akan terlihat seperti berikut.

```
{
   "role":"collectionRole",
   "privileges":[
      {
         "resource":{
            "db":"sample-database-2",
            "collection":"col2"
         },
         "actions":[
            "find",
            "update",
            "insert",
            "remove"
         ]
      }
   ],
   "roles":[

   ]
}
```

Contoh 9: Buat pengguna dan tetapkan peran yang ditentukan pengguna

```
db.createUser(
{
    user: "user3",
    pwd: "abc123",
    roles: []
})

db.grantRolesToUser("user3",["collectionRole"])
```

Contoh 10: Berikan hak istimewa tambahan untuk peran yang ditentukan pengguna

```
db.grantPrivilegesToRole(
  "collectionRole",
  [
    {
      resource: { db: "sample-database-1", collection: "col1" },
      actions: ["find", "update", "insert", "remove"]
    }
  ]
)
```

Contoh 11: Hapus hak istimewa dari peran yang ditentukan pengguna

```
db.revokePrivilegesFromRole(
  "collectionRole",
  [
    {
      resource: { db: "sample-database-1", collection: "col2" },
      actions: ["find", "update", "insert", "remove"]
    }
  ]
)
```

Contoh 12: Perbarui peran yang ditentukan pengguna yang sudah ada

```
db.updateRole(
  "collectionRole",
  {
    privileges: [
    {
        resource: {db: "sample-database-3", collection: "sample-collection-3"},
 actions: ["find", "update", "insert", "remove"]
```

```
    }],
    roles: []
  }
)
```

Perbedaan fungsional

Di Amazon DocumentDB, definisi pengguna dan peran disimpan dalam basis data `admin` dan pengguna diautentikasi terhadap basis data `admin`. Fungsionalitas ini berbeda dari Edisi Komunitas MongoDB, tetapi konsisten dengan MongoDB Atlas.

Amazon DocumentDB juga mendukung perubahan pengaliran, yang menyediakan urutan sesuai waktu dari peristiwa perubahan yang terjadi di dalam koleksi klaster Anda. Tindakan `listChangeStreams` diterapkan pada tingkat klaster (yaitu, di semua basis data), dan tindakan `modifyChangeStreams` dapat diterapkan pada tingkat basis data dan tingkat klaster.

Batas

Tabel berikut berisi batas untuk Role-Based Access Control di Amazon DocumentDB.

Deskripsi	Kuota
Jumlah pengguna per klaster	1000
Jumlah peran yang berkaitan dengan pengguna	1000
Jumlah peran yang ditentukan pengguna	100
Jumlah sumber daya yang terkait dengan hak istimewa	100

Akses database menggunakan Role-Based Access Control

Dengan kontrol akses berbasis peran, Anda dapat membuat pengguna dan memberikan satu atau lebih peran untuk menentukan operasi apa yang dapat dilakukan pengguna dalam basis data atau klaster.

Berikut ini adalah daftar peran built-in yang saat ini didukung di Amazon DocumentDB.

> **Note**
>
> Di Amazon DocumentDB 4.0 dan 5.0, `ListCollection` perintah `ListDatabase` dan secara opsional dapat menggunakan `authorizedCollections` parameter `authorizedDatabases` dan untuk mencantumkan koleksi dan database yang diizinkan pengguna untuk mengakses dengan memerlukan `listCollections` peran dan masing-masing. `listDatabase` Juga, pengguna sekarang memiliki kemampuan untuk menghilangkan kursor mereka sendiri tanpa memerlukan peran `KillCursor`.

Database user

Nama peran	Deskripsi	Tindakan

`read`	Memberikan kepada pengguna akses baca ke basis data yang ditetapkan.	`changeStreams`
		`collStats`
		`dbStats`
		`find`
		`killCursors`
		`listIndexes`
		`listCollections`

`readWrite`	Memberikan kepada pengguna akses baca dan tulis ke basis data yang ditetapkan.	Semua tindakan dari izin `read`.
		`createCollection`
		`dropCollection`
		`createIndex`
		`dropIndex`

Nama peran	Deskripsi	Tindakan

		`insert`
		`killCursors`
		`listIndexes`
		`listCollections`
		`remove`
		`update`

Cluster user

Nama peran	Deskripsi	Tindakan

membaca AnyDatabase	Memberikan kepada pengguna akses baca ke semua basis data dalam klaster.	Semua tindakan dari izin `read`.
		`listChangeStreams`
		`listDatabases`

`readWrite` AnyDatabase	Memberikan kepada pengguna akses baca dan tulis ke semua basis data dalam klaster.	Semua tindakan dari izin `readWrite` .
		`listChangeStreams`
		`listDatabases`

`userAdmin` AnyDatabase	Memberikan kepada pengguna kemampuan untuk menetapkan dan memodifikasi peran atau hak istimewa yang dimiliki setiap pengguna untuk basis data yang ditetapkan.	`changeCustomData` `changePassword`

Nama peran	Deskripsi	Tindakan

		`createUser`
		`dropRole`
		`dropUser`
		`grantRole`
		`listDatabases`
		`revokeRole`
		`viewRole`
		`viewUser`

`dbAdminAnyDatabase`	Memberikan kepada pengguna kemampuan untuk melakukan peran administrasi basis data pada setiap basis data yang ditetapkan.	Semua tindakan dari izin `dbAdmin`.
		`dropCollection`
		`listDatabases`
		`listChangeStreams`
		`modifyChangeStreams`

Superuser

Nama peran	Deskripsi	Tindakan
`root`	Memberikan kepada pengguna akses ke sumber daya dan operasi dari semua gabungan peran berikut ini: `readWriteAnyDatabase` , `dbAdminAnyDatabase`	Semua tindakan dari `readWriteAnyDatabase` , `dbAdminAnyDatabase`

Nama peran	Deskripsi	Tindakan
	, `userAdminAnyDatabase` , `clusterAdmin` , `restore`, dan `backup`.	, `userAdminAnyDatabase` , `clusterAdmin` , `restore`, dan `backup`.

Database administrator

Nama peran	Deskripsi	Tindakan

`dbAdmin`	Memberikan kepada pengguna kemampuan untuk melakukan tugas administratif pada basis data yang ditetapkan.	`bypassDocumentValidation`
		`collMod`
		`collStats`
		`createCollection`
		`createIndex`
		`dropCollection`
		`dropDatabase`
		`dropIndex`
		`dbStats`
		`find`
		`killCursors`
		`listIndexes`

Nama peran	Deskripsi	Tindakan

		`listCollections`
		
		`modifyChangeStreams`

`dbOwner`	Memberikan kepada pengguna kemampuan untuk melakukan tugas administratif pada basis data yang ditetapkan dengan menggabungkan peran `dbAdmin` dan `readWrite` .	Semua tindakan dari `dbAdmin` dan `readWrite` .

Cluster administrator

Nama peran	Deskripsi	Tindakan

`clusterAdmin`	Memberikan kepada pengguna akses manajemen klaster terbesar dengan menggabungkan peran `clusterManager` , `clusterMonitor` , dan `hostManager` .	Semua tindakan dari `clusterManager` , `clusterMonitor` , dan `hostManager` .
		
		`listChangeStreams`
		
		`dropDatabase`
		
		`modifyChangeStreams`

`clusterManager`	Memberikan kepada pengguna kemampuan untuk mengambil tindakan manajemen dan pemantauan pada klaster yang ditetapkan.	`listChangeStreams`
		
		`listSessions`
		
		`modifyChangeStreams`

Nama peran	Deskripsi	Tindakan
		`replSetGetConfig`

`clusterMonitor`	Memberikan kepada pengguna kemampuan untuk memiliki akses hanya-baca ke alat pemantauan.	`collStats`
		`dbStats`
		`find`
		`getParameter`
		`hostInfo`
		`indexStats`
		`killCursors`
		`listChangeStreams`
		`listCollections`
		`listDatabases`
		`listIndexes`
		`listSessions`
		`replSetGetConfig`
		`serverStatus`
		`top`

Nama peran	Deskripsi	Tindakan

`hostManager`	Memberikan kepada pengguna kemampuan untuk memantau dan mengelola server.	`auditConfigure`
		`killCursors`
		`killAnyCursor`
		`killAnySession`
		`killop`

Backup administrator

Nama peran	Deskripsi	Tindakan

`backup`	Memberikan kepada pengguna akses yang diperlukan untuk mencadangkan data.	`getParameter`
		`insert`
		`find`
		`listChangeStreams`
		`listCollections`
		`listDatabases`
		`listIndexes`
		`update`

`restore`	Memberikan kepada pengguna akses yang diperlukan untuk memulihkan data.	`bypassDocumentValidation` `changeCustomData`

Nama peran	Deskripsi	Tindakan
		changePassword collMod createCollection createIndex createUser dropCollection dropRole dropUser getParameter grantRole find insert listCollections modifyChangeStreams revokeRole remove viewRole viewUser update

Pencatatan dan pemantauan di Amazon DocumentDB

Amazon DocumentDB (dengan kompatibilitas MongoDB) menyediakan berbagai metrik CloudWatch Amazon yang dapat Anda pantau untuk menentukan kesehatan dan kinerja cluster dan instans Amazon DocumentDB Anda. Anda dapat melihat metrik Amazon DocumentDB menggunakan berbagai alat, termasuk konsol Amazon DocumentDB, konsol Amazon, AWS CLI dan API. CloudWatch CloudWatch Untuk informasi selengkapnya tentang pemantauan, lihat Memantau Amazon DocumentDB.

Selain CloudWatch metrik Amazon, Anda dapat menggunakan profiler untuk mencatat waktu eksekusi dan detail operasi yang dilakukan di klaster Anda. Profiler berguna untuk memantau operasi paling lambat di klaster Anda untuk membantu Anda meningkatkan performa kueri individual dan performa klaster secara keseluruhan. Saat diaktifkan, operasi akan dicatat ke Amazon CloudWatch Logs dan Anda dapat menggunakan CloudWatch Insight untuk menganalisis, memantau, dan mengarsipkan data profil Amazon DocumentDB Anda. Untuk informasi selengkapnya, lihat Membuat profil operasi Amazon DocumentDB.

Amazon DocumentDB juga terintegrasi AWS CloudTrail dengan, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau layanan AWS di Amazon DocumentDB (dengan kompatibilitas MongoDB). CloudTrail menangkap semua panggilan AWS CLI API untuk Amazon DocumentDB sebagai peristiwa, termasuk panggilan dari Amazon DocumentDB dan dari panggilan kode ke Konsol Manajemen AWS Amazon DocumentDB SDK. Untuk informasi selengkapnya, lihat Mencatat panggilan API Amazon DocumentDB dengan AWS CloudTrail.

Dengan Amazon DocumentDB, Anda dapat mengaudit peristiwa yang dilakukan di klaster Anda. Contoh log acara termasuk upaya autentikasi yang berhasil dan gagal, membuang koleksi dalam basis data, atau membuat indeks. Secara default, pengauditan dinonaktifkan di Amazon DocumentDB dan mengharuskan Anda untuk ikut serta dalam fitur ini. Untuk informasi selengkapnya, lihat Mengaudit acara Amazon DocumentDB.

Memperbarui sertifikat Amazon DocumentDB TLS Anda

Topik

- Memperbarui aplikasi Anda dan klaster Amazon DocumentDB
- Rotasi sertifikat server otomatis
- Pemecahan masalah

- Pertanyaan yang Sering Diajukan

Sertifikat otoritas sertifikat (CA) untuk klaster Amazon DocumentDB diperbarui pada Agustus 2024. Jika Anda menggunakan klaster Amazon DocumentDB dengan Keamanan Lapisan Pengangkutan (TLS) diaktifkan (pengaturan default) dan Anda belum merotasi aplikasi klien dan sertifikat server, langkah-langkah berikut diperlukan untuk mengurangi masalah konektivitas antara aplikasi Anda dan klaster Amazon DocumentDB Anda.

- Langkah 1: Unduh sertifikat CA baru dan perbarui aplikasi Anda
- Langkah 2: Perbarui sertifikat server

Sertifikat CA dan server diperbarui sebagai bagian dari praktik terbaik pemeliharaan dan keamanan standar untuk Amazon DocumentDB. Aplikasi klien harus menambahkan sertifikat CA baru ke penyimpanan kepercayaan mereka, dan instans Amazon DocumentDB yang ada harus diperbarui untuk menggunakan sertifikat CA baru sebelum tanggal kedaluwarsa ini.

Memperbarui aplikasi Anda dan klaster Amazon DocumentDB

Ikuti langkah-langkah di bagian ini untuk memperbarui paket sertifikat CA aplikasi Anda (Langkah 1) dan sertifikat server klaster Anda (Langkah 2). Sebelum Anda menerapkan perubahan ke lingkungan produksi, kami sangat merekomendasikan untuk menguji langkah-langkah ini di lingkungan pengembangan atau penahapan.

> **Note**
>
> Anda harus menyelesaikan Langkah 1 dan 2 Wilayah AWS di masing-masing tempat Anda memiliki cluster Amazon DocumentDB.

Langkah 1: Unduh sertifikat CA baru dan perbarui aplikasi Anda

Unduh sertifikat CA baru dan perbarui aplikasi Anda untuk menggunakan sertifikat CA baru guna membuat koneksi TLS ke Amazon DocumentDB. Unduh paket sertifikat CA baru dari https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem. Operasi ini mengunduh file bernama `global-bundle.pem`.

> **Note**
>
> Jika Anda mengakses keystore yang berisi sertifikat CA lama (`rds-ca-2019-root.pem`) dan sertifikat CA baru (`rds-ca-rsa2048-g1`,,`rds-ca-ecc384-g1`)`rds-ca-rsa4096-g1`, verifikasi bahwa keystore memilih. `global-bundle`

```
wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
```

Selanjutnya, perbarui aplikasi Anda untuk menggunakan paket sertifikat baru. Bundel CA baru berisi sertifikat CA lama (rds-ca-2019) dan sertifikat CA baru (rds-ca-rsa2048-g1, rds-ca-rsa4096-g1, rds-ca-ecc384-g1). Dengan memiliki kedua sertifikat CA di paket CA baru, Anda dapat memperbarui aplikasi dan klaster dalam dua langkah.

Untuk aplikasi Java, Anda harus membuat toko kepercayaan baru dengan sertifikat CA baru. Untuk petunjuk, lihat tab Java di Menghubungkan dengan TLS diaktifkan topik.

Untuk memverifikasi bahwa aplikasi Anda menggunakan paket sertifikat CA terbaru, lihat Bagaimana saya bisa yakin bahwa saya menggunakan paket CA terbaru?. Jika Anda sudah menggunakan paket sertifikat CA terbaru di aplikasi Anda, Anda dapat melompat ke langkah 2.

Untuk contoh penggunaan paket CA dengan aplikasi Anda, lihat Mengenkripsi data dalam perjalanan dan Menghubungkan dengan TLS diaktifkan.

> **Note**
>
> Saat ini, MongoDB Go Driver 1.2.1 hanya menerima satu sertifikat server CA di `sslcertificateauthorityfile`. Lihat Menghubungkan dengan TLS diaktifkan untuk menghubungkan ke Amazon DocumentDB menggunakan Go saat TLS diaktifkan.

Langkah 2: Perbarui sertifikat server

Setelah aplikasi telah diperbarui untuk menggunakan paket CA baru, langkah berikutnya adalah untuk memperbarui sertifikat server dengan memodifikasi setiap instans dalam sebuah klaster Amazon DocumentDB. Untuk mengubah instans untuk menggunakan sertifikat server baru, lihat petunjuk berikut.

Amazon DocumentDB menyediakan CA berikut untuk menandatangani sertifikat server DB untuk instans DB:

- rds-ca-ecc384-g1 —Menggunakan otoritas sertifikat dengan algoritma kunci pribadi ECC 384 dan algoritma penandatanganan SHA384. CA ini mendukung rotasi sertifikat server otomatis. Ini hanya didukung di Amazon DocumentDB 4.0 dan 5.0.
- rds-ca-rsa2048-g1 —Menggunakan otoritas sertifikat dengan algoritma kunci pribadi RSA 2048 dan algoritma penandatanganan SHA256 di sebagian besar wilayah. AWS CA ini mendukung rotasi sertifikat server otomatis.
- rds-ca-rsa4096-g1 —Menggunakan otoritas sertifikat dengan algoritma kunci pribadi RSA 4096 dan algoritma penandatanganan SHA384. CA ini mendukung rotasi sertifikat server otomatis.

> **Note**
>
> Jika Anda menggunakan AWS CLI, Anda dapat melihat validitas otoritas sertifikat yang tercantum di bagian sebelumnya dengan menggunakan deskripsi-sertifikat.

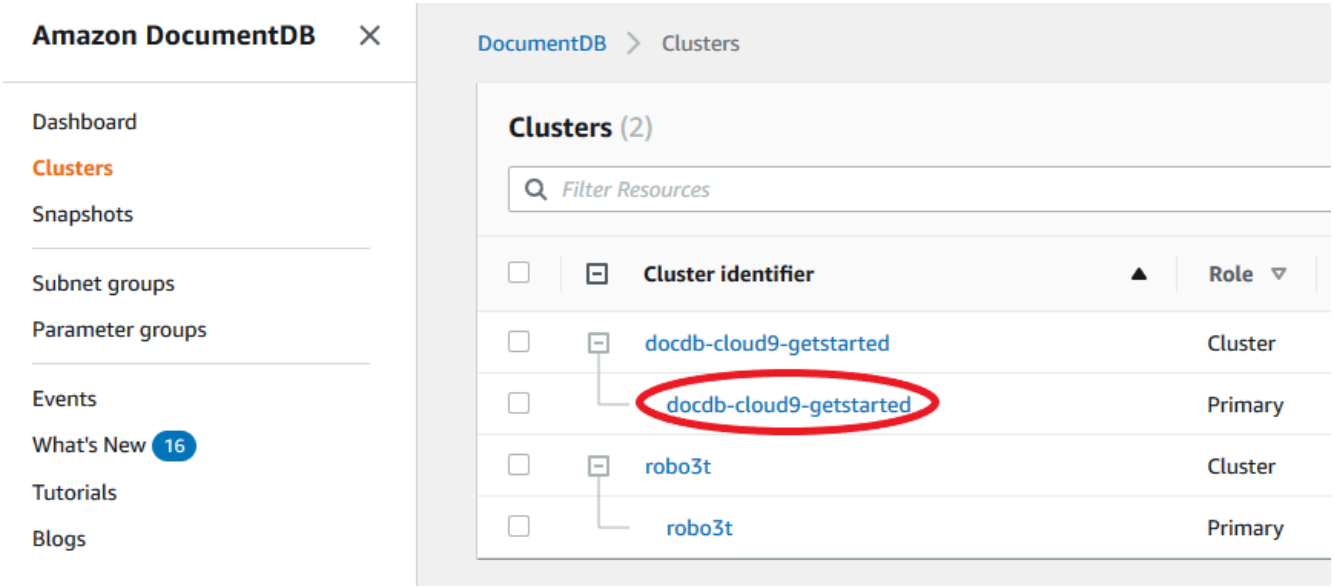
Sertifikat CA ini termasuk dalam paket sertifikat regional dan global. Bila Anda menggunakan rds-ca-rsa2048-g1, rds-ca-rsa4096-g1, atau rds-ca-ecc384-g1 CA dengan database, Amazon DocumentDB mengelola sertifikat server DB pada database. Amazon DocumentDB memutar sertifikat server DB secara otomatis sebelum kedaluwarsa.

> **Note**
>
> Amazon DocumentDB tidak memerlukan reboot untuk rotasi sertifikat jika klaster Anda berjalan pada versi patch engine berikut:
>
> - Amazon DocumentDB 3.6:1.0.208662 atau lebih tinggi
> - Amazon DocumentDB 4.0:2.0.10179 atau lebih tinggi
> - Amazon DocumentDB 5.0:3.0.4780 atau lebih tinggi
>
> Anda dapat menentukan versi patch mesin Amazon DocumentDB saat ini dengan menjalankan perintah berikut:. `db.runCommand({getEngineVersion: 1})` Sebelum memperbarui sertifikat server, pastikan Anda telah menyelesaikannyaLangkah 1.

Using the Konsol Manajemen AWS

Selesaikan langkah-langkah berikut untuk mengidentifikasi dan memutar sertifikat server lama untuk instans Amazon DocumentDB Anda yang ada menggunakan Konsol Manajemen AWS.

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Dalam daftar Wilayah di sudut kanan atas layar, pilih tempat klaster Anda Wilayah AWS berada.
3. Di panel navigasi di sisi kiri konsol, pilih Klaster.
4. Anda mungkin perlu mengidentifikasi instans mana yang masih di sertifikat server lama (`rds-ca-2019`). Anda dapat melakukan ini di kolom otoritas Sertifikat yang terletak di paling kanan tabel Clusters.
5. Dalam tabel Clusters, Anda akan melihat kolom Cluster identifier di paling kiri. Instance Anda tercantum di bawah cluster, mirip dengan tangkapan layar berikut.



6. Centang kotak di sebelah kiri instans yang Anda minati.
7. Pilih Tindakan, dan kemudian pilih Modifikasi.
8. Di bawah Otoritas sertifikat, pilih sertifikat server baru (`rds-ca-rsa2048-g1`) untuk instans ini.
9. Anda dapat melihat ringkasan perubahan pada halaman berikutnya. Perhatikan bahwa ada peringatan tambahan untuk mengingatkan Anda untuk memastikan bahwa aplikasi Anda menggunakan paket sertifikat CA terbaru sebelum memodifikasi instans untuk menghindari menyebabkan gangguan dalam konektivitas.

10. Anda dapat memilih untuk menerapkan modifikasi selama jendela pemeliharaan berikutnya atau menerapkan dengan segera. Jika maksud Anda adalah untuk mengubah sertifikat server dengan segera, gunakan opsi Terapkan Segera.
11. Pilih Modifikasi instans untuk menyelesaikan pembaruan.

Using the AWS CLI

Selesaikan langkah-langkah berikut untuk mengidentifikasi dan memutar sertifikat server lama untuk instans Amazon DocumentDB Anda yang ada menggunakan AWS CLI.

1. Untuk memodifikasi instans dengan segera, jalankan perintah berikut untuk setiap instans dalam klaster.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier>
 --ca-certificate-identifier rds-ca-rsa2048-g1 --apply-immediately
```

2. Untuk memodifikasi instans dalam klaster Anda agar menggunakan sertifikat CA baru selama jendela pemeliharaan klaster Anda berikutnya, jalankan perintah berikut untuk setiap instans dalam klaster.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier>
 --ca-certificate-identifier rds-ca-rsa2048-g1 --no-apply-immediately
```

Rotasi sertifikat server otomatis

Amazon DocumentDB mendukung rotasi sertifikat server otomatis. Sertifikat server adalah sertifikat daun yang dikeluarkan untuk setiap instance cluster. Berbeda dengan sertifikat CA root, sertifikat server memiliki validitas pendek (12 bulan) dan Amazon DocumentDB secara otomatis menangani rotasi mereka tanpa tindakan apa pun dari Anda. Amazon DocumentDB menggunakan CA root yang sama untuk rotasi otomatis ini, jadi Anda tidak perlu mengunduh bundel CA baru.

> **Important**
>
> Saat menghubungkan ke klaster Amazon DocumentDB Anda, sebaiknya Anda mempercayai bundel CA root sebagai lawan mempercayai secara langsung setiap sertifikat server. Ini akan mencegah kesalahan koneksi setelah sertifikat server diputar. Lihat Menghubungkan dengan TLS diaktifkan.

Amazon DocumentDB mencoba memutar sertifikat server Anda di jendela pemeliharaan pilihan Anda pada waktu paruh sertifikat server. Sertifikat server baru berlaku selama 12 bulan.

Gunakan `describe-db-engine-versions`perintah dan periksa `SupportsCertificateRotationWithoutRestart` bendera untuk mengidentifikasi apakah versi mesin mendukung memutar sertifikat tanpa restart.

> **Note**
>
> Amazon DocumentDB mendukung rotasi sertifikat server tanpa restart jika klaster Anda berjalan pada versi patch engine berikut:
>
> - Amazon DocumentDB 3.6:1.0.208662 atau lebih tinggi
> - Amazon DocumentDB 4.0:2.0.10179 atau lebih tinggi
> - Amazon DocumentDB 5.0:3.0.4780 atau lebih tinggi
>
> Anda dapat menentukan versi patch mesin Amazon DocumentDB saat ini dengan menjalankan perintah ini:. `db.runCommand({getEngineVersion: 1})`
> Jika Anda menggunakan versi patch mesin yang lebih lama, Amazon DocumentDB akan memutar sertifikat server dan menjadwalkan acara restart database pada jendela pemeliharaan pilihan Anda.

Pemecahan masalah

Jika Anda mengalami masalah saat menyambungkan ke klaster sebagai bagian dari rotasi sertifikat, kami menyarankan hal berikut:

- Verifikasi bahwa klien Anda menggunakan bundel sertifikat terbaru. Lihat Bagaimana saya bisa yakin bahwa saya menggunakan paket CA terbaru?.
- Verifikasi bahwa instans Anda menggunakan sertifikat terbaru. Lihat Bagaimana cara mengetahui instans Amazon DocumentDB mana yang menggunakan sertifikat server? old/new.
- Verifikasi bahwa sertifikat CA terbaru sedang digunakan oleh aplikasi Anda. Beberapa driver, seperti Java dan Go, memerlukan kode tambahan untuk mengimpor beberapa sertifikat dari paket sertifikat ke penyimpanan kepercayaan. Untuk informasi selengkapnya tentang menghubungkan ke Amazon DocumentDB dengan TLS, lihat Menghubungkan secara terprogram ke Amazon DocumentDB.

- Hubungi dukungan. Jika Anda memiliki pertanyaan atau masalah, hubungi Dukungan.

Pertanyaan yang Sering Diajukan

Berikut ini adalah jawaban atas beberapa pertanyaan umum tentang sertifikat TLS.

Bagaimana jika saya memiliki pertanyaan atau masalah?

Jika Anda memiliki pertanyaan atau masalah, hubungi Dukungan.

Bagaimana saya tahu apakah saya menggunakan TLS untuk terhubung ke klaster Amazon DocumentDB saya?

Anda dapat menentukan apakah klaster Anda menggunakan TLS dengan memeriksa parameter `tls` untuk grup parameter klaster dari klaster Anda. Jika parameter `tls` diatur ke `enabled`, Anda menggunakan sertifikat TLS untuk terhubung ke klaster Anda. Untuk informasi selengkapnya, lihat Mengelola grup parameter cluster Amazon DocumentDB.

Mengapa Anda memperbarui sertifikat CA dan server?

Amazon DocumentDB CA dan sertifikat server sedang diperbarui sebagai bagian dari pemeliharaan standar dan praktik terbaik keamanan untuk Amazon DocumentDB.

Apa yang terjadi jika saya tidak mengambil tindakan apa pun pada tanggal kedaluwarsa?

Jika Anda menggunakan TLS untuk terhubung ke cluster Amazon DocumentDB Anda dengan sertifikat CA kedaluwarsa, aplikasi Anda yang terhubung melalui TLS tidak akan lagi dapat berkomunikasi dengan cluster Amazon DocumentDB.

Amazon DocumentDB tidak akan memutar sertifikat database Anda secara otomatis sebelum kedaluwarsa. Anda harus memperbarui aplikasi dan cluster untuk menggunakan sertifikat CA baru sebelum atau setelah tanggal kedaluwarsa.

Bagaimana cara mengetahui instans Amazon DocumentDB mana yang menggunakan sertifikat server? old/new

Untuk mengidentifikasi instans Amazon DocumentDB yang masih menggunakan sertifikat server lama, Anda dapat menggunakan Amazon DocumentDB atau file. Konsol Manajemen AWS AWS CLI

Menggunakan Konsol Manajemen AWS

Untuk mengidentifikasi instans di klaster Anda yang menggunakan sertifikat yang lebih lama

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Dalam daftar Wilayah di sudut kanan atas layar, pilih tempat instance Anda Wilayah AWS berada.
3. Di panel navigasi di sisi kiri konsol, pilih Klaster.
4. Kolom otoritas Sertifikat (dekat paling kanan tabel) menunjukkan instance mana yang masih ada di sertifikat server lama (`rds-ca-2019`) dan sertifikat server baru (`rds-ca-rsa2048-g1`).

Menggunakan AWS CLI

Untuk mengidentifikasi instans di klaster Anda yang menggunakan sertifikat server yang lebih lama, gunakan perintah `describe-db-clusters` dengan berikut ini.

```
aws docdb describe-db-instances \
    --filters Name=engine,Values=docdb \
    --query 'DBInstances[*].
{CertificateVersion:CACertificateIdentifier,InstanceID:DBInstanceIdentifier}'
```

Bagaimana cara memodifikasi instans individu di klaster Amazon DocumentDB saya untuk memperbarui sertifikat server?

Kami merekomendasikan Anda memperbarui sertifikat server untuk semua instans dalam klaster tertentu pada waktu yang sama. Untuk memodifikasi instans di klaster Anda, Anda dapat menggunakan konsol atau AWS CLI.

> **Note**
>
> Sebelum memperbarui sertifikat server, pastikan Anda telah menyelesaikan Langkah 1.

Menggunakan Konsol Manajemen AWS

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Dalam daftar Wilayah di sudut kanan atas layar, pilih tempat klaster Anda Wilayah AWS berada.
3. Di panel navigasi di sisi kiri konsol, pilih Klaster.
4. Kolom otoritas Sertifikat (dekat paling kanan tabel) menunjukkan instance mana yang masih ada di sertifikat server lama (`rds-ca-2019`).
5. Dalam tabel Clusters, di bawah Cluster identifier, pilih instance untuk memodifikasi.
6. Pilih Tindakan, dan kemudian pilih Modifikasi.
7. Di bawah Otoritas sertifikat, pilih sertifikat server baru (`rds-ca-rsa2048-g1`) untuk instans ini.
8. Anda dapat melihat ringkasan perubahan pada halaman berikutnya. Perhatikan bahwa ada peringatan tambahan untuk mengingatkan Anda untuk memastikan bahwa aplikasi Anda menggunakan paket sertifikat CA terbaru sebelum memodifikasi instans untuk menghindari menyebabkan gangguan dalam konektivitas.
9. Anda dapat memilih untuk menerapkan modifikasi selama jendela pemeliharaan berikutnya atau menerapkan dengan segera.
10. Pilih Modifikasi instans untuk menyelesaikan pembaruan.

Menggunakan AWS CLI

Selesaikan langkah-langkah berikut untuk mengidentifikasi dan memutar sertifikat server lama untuk instans Amazon DocumentDB Anda yang ada menggunakan AWS CLI.

1. Untuk memodifikasi instans dengan segera, jalankan perintah berikut untuk setiap instans dalam klaster.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier> --
ca-certificate-identifier rds-ca-rsa2048-g1 --apply-immediately
```

2. Untuk memodifikasi instans dalam klaster Anda agar menggunakan sertifikat CA baru selama jendela pemeliharaan klaster Anda berikutnya, jalankan perintah berikut untuk setiap instans dalam klaster.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier> --
ca-certificate-identifier rds-ca-rsa2048-g1 --no-apply-immediately
```

Apa yang terjadi jika saya menambahkan instans baru ke klaster yang ada?

Semua instans baru yang dibuat menggunakan sertifikat server lama dan memerlukan koneksi TLS menggunakan sertifikat CA lama. Instans Amazon DocumentDB baru yang dibuat setelah 25 Januari 2024 akan default menggunakan sertifikat baru rds-ca-rsa2048-g1.

Apa yang terjadi jika ada penggantian instans atau failover di klaster saya?

Jika ada pengganti instans di klaster Anda, instans baru yang dibuat akan terus menggunakan sertifikat server yang sama dengan yang digunakan instans sebelumnya. Kami merekomendasikan Anda memperbarui sertifikat server untuk semua instans pada waktu yang sama. Jika terjadi failover di klaster, sertifikat server pada primer baru digunakan.

Jika saya tidak menggunakan TLS untuk terhubung ke klaster saya, apakah saya masih perlu memperbarui setiap instans saya?

Kami sangat menyarankan untuk mengaktifkan TLS. Jika Anda tidak mengaktifkan TLS, kami tetap menyarankan untuk memutar sertifikat pada instans Amazon DocumentDB Anda jika Anda berencana menggunakan TLS untuk terhubung ke cluster Anda di masa mendatang. Jika Anda tidak pernah berencana menggunakan TLS untuk terhubung ke cluster Amazon DocumentDB Anda, tidak diperlukan tindakan.

Jika saya tidak menggunakan TLS untuk terhubung ke klaster saya tetapi saya berencana untuk melakukannya di masa mendatang, apa yang harus saya lakukan?

Jika Anda membuat klaster sebelum Januari 2024, ikuti Langkah 1 dan Langkah 2 di bagian sebelumnya untuk memastikan bahwa aplikasi Anda menggunakan bundel CA yang diperbarui, dan bahwa setiap instans Amazon DocumentDB menggunakan sertifikat server terbaru. Jika Anda membuat cluster setelah 25 Januari 2024, cluster Anda sudah memiliki sertifikat server terbaru (rds-ca-rsa2048-g1). Untuk memverifikasi bahwa aplikasi Anda menggunakan paket CA terbaru, lihat Jika saya tidak menggunakan TLS untuk terhubung ke klaster saya, apakah saya masih perlu memperbarui setiap instans saya?

Bisakah tenggat waktu diperpanjang melampaui Agustus 2024?

Jika aplikasi Anda terhubung melalui TLS, batas waktu tidak dapat diperpanjang.

Bagaimana saya bisa yakin bahwa saya menggunakan paket CA terbaru?

Untuk memverifikasi bahwa Anda memiliki bundel terbaru, gunakan perintah berikut. Untuk menjalankan perintah ini, Anda harus menginstal java dan alat java harus berada dalam variabel PATH shell Anda. Untuk informasi selengkapnya, lihat Menggunakan Java

macOS dan Amazon Linux

```
keytool -printcert -v -file global-bundle.pem
```

Windows

```
keytool -printcert -v -file global-bundle.p7b
```

Mengapa saya melihat “RDS” dalam nama paket CA?

Untuk fitur pengelolaan tertentu, seperti manajemen sertifikat, Amazon DocumentDB menggunakan teknologi operasional yang dibagikan dengan Amazon Relational Database Service (Amazon RDS).

Kapan sertifikat baru akan kedaluwarsa?

Sertifikat server baru akan kedaluwarsa (umumnya) sebagai berikut:

- rds-ca-rsa2048-g1 —Kedaluwarsa 2061
- rds-ca-rsa4096-g1 —Kedaluwarsa 2121
- rds-ca-ecc384-g1 —Kedaluwarsa 2121

Kesalahan apa yang akan saya lihat jika saya tidak mengambil tindakan sebelum sertifikat berakhir?

Pesan kesalahan akan bervariasi tergantung pada driver Anda. Secara umum, Anda akan melihat kesalahan validasi sertifikat yang berisi string “sertifikat telah kedaluwarsa”.

Jika saya menerapkan sertifikat server baru, dapatkah saya mengembalikannya ke sertifikat server lama?

Jika Anda perlu mengembalikan instans ke sertifikat server lama, kami merekomendasikan Anda melakukannya untuk semua instans di klaster. Anda dapat mengembalikan sertifikat server untuk setiap instance dalam klaster dengan menggunakan Konsol Manajemen AWS atau. AWS CLI

Menggunakan Konsol Manajemen AWS

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Dalam daftar Wilayah di sudut kanan atas layar, pilih tempat klaster Anda Wilayah AWS berada.
3. Di panel navigasi di sisi kiri konsol, pilih Klaster.
4. Dalam tabel Clusters, di bawah Cluster identifier, pilih instance untuk memodifikasi. Pilih Tindakan, dan kemudian pilih Modifikasi.
5. Di bawah Otoritas sertifikat, Anda dapat memilih sertifikat server lama (`rds-ca-2019`).
6. Pilih Lanjutkan dan lihat ringkasan dari modifikasi Anda.
7. Di halaman yang dihasilkan ini, Anda dapat memilih untuk menjadwalkan modifikasi Anda untuk diterapkan di jendela pemeliharaan berikutnya atau segera menerapkan modifikasi Anda. Tentukan pilihan Anda, dan pilih Modifikasi instans.

> **Note**
>
> Jika Anda memilih untuk menerapkan modifikasi Anda dengan segera, perubahan apa pun pada antrean modifikasi yang tertunda juga akan diterapkan. Jika salah satu dari modifikasi yang tertunda memerlukan waktu henti, memilih opsi ini dapat menyebabkan waktu henti yang tidak terduga.

Menggunakan AWS CLI

```
aws docdb modify-db-instance --db-instance-identifier <db_instance_name> ca-
certificate-identifier rds-ca-2019 <--apply-immediately | --no-apply-immediately>
```

Jika Anda memilih `--no-apply-immediately`, perubahan akan diterapkan selama jendela pemeliharaan klaster berikutnya.

Jika saya memulihkan dari snapshot atau pemulihan titik waktu, apakah itu akan memiliki sertifikat server baru?

Jika Anda memulihkan snapshot atau melakukan pemulihan point-in-time setelah Agustus 2024, cluster baru yang dibuat akan menggunakan sertifikat CA baru.

Bagaimana jika saya mengalami masalah saat terhubung langsung ke cluster Amazon DocumentDB saya dari Mac OS mana pun?

Mac OS telah memperbarui persyaratan untuk sertifikat tepercaya. Sertifikat tepercaya sekarang harus berlaku selama 397 hari atau kurang (lihathttps://support.apple.com/en-us/HT211025).

> **Note**
>
> Pembatasan ini diamati pada versi Mac OS yang lebih baru.

Sertifikat instans Amazon DocumentDB berlaku selama lebih dari empat tahun, lebih lama dari maksimum Mac OS. Untuk terhubung langsung ke cluster Amazon DocumentDB dari komputer yang menjalankan Mac OS, Anda harus mengizinkan sertifikat yang tidak valid saat membuat koneksi TLS. Dalam hal ini, sertifikat yang tidak valid berarti bahwa masa berlaku lebih dari 397 hari. Anda harus memahami risikonya sebelum mengizinkan sertifikat yang tidak valid saat menghubungkan ke klaster Amazon DocumentDB Anda.

Untuk terhubung ke cluster Amazon DocumentDB dari Mac OS menggunakan, gunakan AWS CLI parameter. `tlsAllowInvalidCertificates`

```
mongo --tls --host <hostname> --username <username> --password <password> --port 27017
 --tlsAllowInvalidCertificates
```

Validasi kepatuhan di Amazon DocumentDB

Keamanan dan kepatuhan Amazon DocumentDB dinilai oleh auditor pihak ketiga sebagai bagian dari AWS beberapa program kepatuhan, termasuk yang berikut:

- Kontrol Sistem dan Organisasi (SOC) 1, 2, dan 3. Untuk informasi selengkapnya, lihat SOC.
- Program Manajemen Risiko dan Otorisasi Federal (FedRAMP). Untuk informasi selengkapnya, lihat AWS Layanan dalam Lingkup menurut Program Kepatuhan.
- Standar Keamanan Data Industri Kartu Pembayaran (PCI DSS). Untuk informasi lain, lihat PCI DSS.
- ISO 9001, 27001, 27017, dan 27018. Untuk informasi selengkapnya, lihat Bersertifikat ISO.
- Undang-Undang Akuntabilitas dan Portabilitas Asuransi Kesehatan Perjanjian Rekan Bisnis (HIPAA BAA). Untuk informasi lebih lanjut, lihat Kepatuhan HIPAA

AWS menyediakan daftar AWS layanan yang sering diperbarui dalam lingkup program kepatuhan khusus di AWS Layanan dalam Lingkup oleh Program Kepatuhan.

Third-party Laporan audit tersedia untuk Anda unduh menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Mengunduh Laporan di AWS Artifak.

Untuk informasi selengkapnya tentang program AWS kepatuhan, lihat Program AWS Kepatuhan.

Tanggung jawab kepatuhan Anda saat menggunakan Amazon DocumentDB ditentukan oleh sensitivitas data Anda, tujuan kepatuhan organisasi Anda, serta undang-undang dan peraturan yang berlaku. Jika penggunaan Amazon DocumentDB Anda tunduk pada kepatuhan terhadap standar seperti HIPAA atau PCI AWS , sediakan sumber daya untuk membantu:

- AWS Sumber Daya Kepatuhan — Kumpulan buku kerja dan panduan yang mungkin berlaku untuk industri dan lokasi Anda.
- Panduan Memulai Cepat Keamanan dan Kepatuhan — Panduan penerapan yang membahas pertimbangan arsitektur dan memberikan langkah-langkah untuk menerapkan lingkungan dasar yang berfokus pada keamanan dan kepatuhan. AWS
- AWS Config — Layanan yang menilai seberapa baik konfigurasi sumber daya Anda mematuhi praktik internal, pedoman industri, dan peraturan.
- AWS Security Hub — Pandangan komprehensif tentang status keamanan Anda di dalamnya AWS yang membantu Anda memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik.
- Arsitektur untuk HIPAA Security and Compliance Whitepaper — Whitepaper yang menjelaskan bagaimana perusahaan dapat menggunakan untuk membuat aplikasi. AWS HIPAA-compliant

Ketahanan di Amazon DocumentDB

Infrastruktur AWS global dibangun di sekitar Wilayah AWS dan Availability Zones. Wilayah AWS menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang dan mengoperasikan aplikasi dan basis data yang secara otomatis melakukan failover di antara Zona Ketersediaan tanpa gangguan. Zona Ketersediaan memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur biasa yang terdiri dari satu atau beberapa pusat data.

Sebuah klaster Amazon DocumentDB hanya dapat dibuat dalam Amazon VPC yang memiliki setidaknya dua subnet di setidaknya dua Availability Zone. Dengan mendistribusikan instans

klaster di setidaknya dua Availability Zone, Amazon DocumentDB membantu memastikan bahwa ada instans yang tersedia di klaster Anda apabila terjadi kegagalan Availability Zone yang tidak dimungkinkan. Volume klaster untuk klaster Amazon DocumentDB Anda selalu mencakup tiga Availability Zone untuk menyediakan penyimpanan tahan lama dengan kemungkinan kehilangan data yang lebih kecil.

Untuk informasi selengkapnya tentang Wilayah AWS dan Availability Zone, lihat Infrastruktur AWS Global.

Selain infrastruktur AWS global, Amazon DocumentDB menawarkan beberapa fitur untuk membantu mendukung ketahanan data dan kebutuhan pencadangan Anda.

Fault-tolerant dan penyimpanan penyembuhan diri

Setiap porsi 10 GB dari volume penyimpanan Anda direplikasi dengan enam cara, di tiga Availability Zone. Amazon DocumentDB menggunakan penyimpanan toleran kesalahan yang secara transparan menangani hilangnya hingga dua salinan data tanpa memengaruhi ketersediaan tulis basis data, dan hingga tiga salinan tanpa memengaruhi ketersediaan baca. Penyimpanan Amazon DocumentDB juga memulihkan diri; blok data dan disk terus dipindai untuk mencari kesalahan dan diganti secara otomatis.

Backup dan pemulihan manual

Amazon DocumentDB menyediakan kemampuan untuk membuat backup penuh klaster Anda untuk retensi dan pemulihan jangka panjang. Untuk informasi selengkapnya, lihat Mencadangkan dan memulihkan di Amazon DocumentDB.

Point-in-time pemulihan

Point-in-time pemulihan membantu melindungi cluster Amazon DocumentDB Anda dari operasi menulis atau menghapus yang tidak disengaja. Dengan pemulihan titik waktu, Anda tidak perlu khawatir tentang membuat, memelihara, atau menjadwalkan pencadangan sesuai permintaan. Untuk informasi selengkapnya, lihat Memulihkan ke titik waktu.

Keamanan infrastruktur di Amazon DocumentDB

Sebagai layanan terkelola, Amazon DocumentDB dilindungi AWS oleh keamanan jaringan global. Untuk informasi tentang layanan AWS keamanan dan cara AWS melindungi infrastruktur, lihat Keamanan AWS Cloud. Untuk mendesain AWS lingkungan Anda menggunakan praktik terbaik

untuk keamanan infrastruktur, lihat Perlindungan Infrastruktur dalam Kerangka Kerja yang AWS Diarsiteksikan dengan Baik Pilar Keamanan.

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses Amazon DocumentDB melalui jaringan. Klien harus mendukung hal-hal berikut:

- Keamanan Lapisan Pengangkutan (TLS). Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Cipher suite dengan perfect forward secrecy (PFS) seperti DHE (Ephemeral) atau ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Diffie-Hellman Sebagian besar sistem modern seperti Java 7 dan versi lebih baru mendukung mode-mode ini.

Anda dapat memanggil operasi API ini dari lokasi jaringan mana pun. Anda dapat menggunakan kebijakan Amazon DocumentDB untuk mengontrol akses dari titik akhir Amazon Virtual Private Cloud (Amazon VPC) tertentu atau VPC tertentu. Secara efektif, ini mengisolasi akses jaringan ke sumber daya Amazon DocumentDB tertentu hanya dari VPC tertentu dalam jaringan. AWS

> **Note**
>
> Amazon DocumentDB tidak mendukung kebijakan akses berbasis sumber daya.

Amazon DocumentDB API dan titik akhir VPC antarmuka (AWS PrivateLink)

> **Note**
>
> Cluster elastis Amazon DocumentDB tidak mendukung titik akhir VPC. AWS PrivateLink

Anda dapat membuat koneksi pribadi antara VPC dan titik akhir API Amazon DocumentDB dengan membuat titik akhir VPC antarmuka. Endpoint antarmuka didukung oleh AWS PrivateLink.

Meskipun cluster berbasis instans Amazon DocumentDB tidak memerlukan koneksi titik akhir VPC antarmuka AWS PrivateLink , memungkinkan Anda mengakses operasi Amazon DocumentDB API secara pribadi tanpa gateway internet, perangkat NAT, koneksi VPN, atau koneksi. Direct Connect Instans Amazon DocumentDB di VPC Anda tidak memerlukan alamat IP publik untuk berkomunikasi

dengan titik akhir API Amazon DocumentDB untuk meluncurkan, memodifikasi, atau menghentikan instance database dan cluster database. Instans Amazon DocumentDB Anda juga tidak memerlukan alamat IP publik untuk menggunakan salah satu operasi Amazon DocumentDB API yang tersedia. Lalu lintas antara VPC Anda dan Amazon DocumentDB tidak meninggalkan jaringan Amazon.

Setiap titik akhir antarmuka direpresentasikan oleh satu atau beberapa antarmuka jaringan elastis di subnet Anda. Untuk informasi selengkapnya, lihat Antarmuka jaringan elastis di Panduan Pengguna Amazon EC2.

Untuk informasi selengkapnya tentang titik akhir VPC, lihat Mengakses titik akhir VPC antarmuka Layanan AWS menggunakan antarmuka di Panduan Pengguna Amazon Virtual Private Cloud ().AWS PrivateLink Untuk informasi selengkapnya tentang operasi Amazon DocumentDB, lihat Referensi API Amazon DocumentDB.

Topik

- Pertimbangan untuk titik akhir VPC
- Ketersediaan wilayah
- Membuat titik akhir VPC antarmuka untuk Amazon DocumentDB API
- Membuat kebijakan titik akhir VPC untuk Amazon DocumentDB API

Pertimbangan untuk titik akhir VPC

Sebelum menyiapkan titik akhir VPC antarmuka untuk titik akhir Amazon DocumentDB API, pastikan Anda meninjau prasyarat titik akhir antarmuka di Panduan Pengguna Amazon Virtual Private Cloud ().AWS PrivateLink

Semua operasi Amazon DocumentDB API yang relevan untuk mengelola sumber daya Amazon DocumentDB tersedia dari VPC Anda menggunakan. AWS PrivateLink

Kebijakan titik akhir VPC didukung untuk titik akhir Amazon DocumentDB API. Secara default, akses penuh ke operasi Amazon DocumentDB API diizinkan melalui titik akhir. Untuk informasi selengkapnya, lihat Mengontrol akses ke titik akhir VPC menggunakan kebijakan titik akhir di Panduan Pengguna Amazon Virtual Private Cloud ()AWS PrivateLink.

Ketersediaan wilayah

Amazon DocumentDB API saat ini mendukung titik akhir VPC sebagai berikut: Wilayah AWS

- AS Timur (Ohio)
- AS Timur (Virginia Utara)
- AS Barat (Oregon)
- Afrika (Cape Town)
- Asia Pasifik (Hong Kong)
- Asia Pasifik (Mumbai)
- Asia Pasifik (Hyderabad)
- Asia Pasifik (Osaka)
- Asia Pasifik (Seoul)
- Asia Pasifik (Singapura)
- Asia Pasifik (Sydney)
- Asia Pasifik (Tokyo)
- Kanada (Pusat)
- Tiongkok (Beijing)
- Tiongkok (Ningxia)
- Eropa (Frankfurt)
- Eropa (Irlandia)
- Eropa (London)
- Eropa (Paris)
- Eropa (Spanyol)
- Europe (Milan)
- Timur Tengah (UAE)
- Amerika Selatan (Sao Paulo)
- AWS GovCloud (US-East)
- AWS GovCloud (US-West)

Membuat titik akhir VPC antarmuka untuk Amazon DocumentDB API

Anda dapat membuat titik akhir VPC untuk Amazon DocumentDB API menggunakan Konsol VPC Amazon atau (). AWS Command Line Interface AWS CLI Untuk informasi selengkapnya,

lihat Mengakses titik akhir VPC antarmuka Layanan AWS menggunakan antarmuka di Panduan Pengguna Amazon Virtual Private Cloud (AWS PrivateLink).

Buat titik akhir VPC untuk Amazon DocumentDB API menggunakan nama layanan. `com.amazonaws.region.rds`

Tidak termasuk Wilayah AWS di Tiongkok, jika Anda mengaktifkan DNS pribadi untuk titik akhir, Anda dapat membuat permintaan API ke Amazon DocumentDB dengan titik akhir VPC menggunakan nama DNS default untuk, misalnya rds.us-east-1.amazonaws.com. Wilayah AWS Untuk China (Beijing) dan Tiongkok (Ningxia Wilayah AWS), Anda dapat membuat permintaan API dengan titik akhir VPC masing-masing menggunakan rds-api.cn-north-1.amazonaws.com .cn dan rds-api.cn-northwest-1.amazonaws.com .cn.

Untuk informasi selengkapnya, lihat Mengakses titik akhir VPC antarmuka Layanan AWS menggunakan antarmuka di Panduan Pengguna Amazon Virtual Private Cloud (AWS PrivateLink).

Membuat kebijakan titik akhir VPC untuk Amazon DocumentDB API

Anda dapat melampirkan kebijakan titik akhir ke titik akhir VPC yang mengontrol akses ke Amazon DocumentDB API. Kebijakan titik akhir menentukan informasi berikut:

- Prinsipal yang dapat melakukan tindakan.
- Tindakan yang dapat dilakukan.
- Sumber daya yang menjadi target tindakan.

Untuk informasi selengkapnya, lihat Mengontrol akses ke titik akhir VPC menggunakan kebijakan titik akhir di Panduan Pengguna Amazon Virtual Private Cloud ()AWS PrivateLink.

Contoh: Kebijakan titik akhir VPC untuk tindakan API Amazon DocumentDB

Berikut ini adalah contoh kebijakan endpoint untuk Amazon DocumentDB API. Saat dilampirkan ke titik akhir, kebijakan ini memberikan akses ke tindakan Amazon DocumentDB API yang terdaftar untuk semua prinsipal di semua sumber daya.

```
{
"Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
```

```
            "docdb:CreateDBInstance",
            "docdb:ModifyDBInstance",
            "docdb:CreateDBSnapshot"
         ],
         "Resource":"*"
      }
   ]
}
```

Contoh: Kebijakan titik akhir VPC yang menolak semua akses dari akun tertentu AWS

Kebijakan titik akhir VPC berikut menyangkal AWS akun 123456789012 semua akses ke sumber daya menggunakan titik akhir. Kebijakan ini mengizinkan semua tindakan dari akun lainnya.

```
{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Resource": "*",
      "Principal": "*"
    },
    {
      "Action": "*",
      "Effect": "Deny",
      "Resource": "*",
      "Principal": { "AWS": [ "123456789012" ] }
     }
   ]
}
```

Praktik terbaik keamanan untuk Amazon DocumentDB

Untuk praktik terbaik keamanan, Anda harus menggunakan akun AWS Identity and Access Management (IAM) untuk mengontrol akses ke operasi Amazon DocumentDB API, terutama operasi yang membuat, memodifikasi, atau menghapus sumber daya Amazon DocumentDB. Sumber daya tersebut termasuk klaster, grup keamanan, dan grup parameter. Anda juga harus menggunakan IAM untuk mengontrol tindakan yang melakukan tindakan administrasi umum seperti mencadangkan pemulihan klaster. Saat membuat IAM role, terapkan prinsip hak istimewa terendah.

- Terapkan hak istimewa terendah dengan kontrol akses berbasis peran.

- Tetapkan akun IAM individu untuk setiap orang yang mengelola sumber daya Amazon DocumentDB. Jangan gunakan pengguna Akun AWS root untuk mengelola sumber daya Amazon DocumentDB. Buat pengguna IAM untuk semua orang, termasuk Anda sendiri.
- Berikan setiap pengguna set izin minimum yang diperlukan untuk melakukan tugas mereka.
- Gunakan grup IAM untuk mengelola izin secara efektif bagi beberapa pengguna. Untuk informasi lebih lanjut tentang IAM, lihat Panduan Pengguna IAM. Untuk informasi tentang praktik terbaik IAM, lihat Praktik Terbaik IAM.
- Secara rutin putar kredensial IAM Anda.
- Konfigurasikan AWS Secrets Manager untuk secara otomatis memutar rahasia untuk Amazon DocumentDB. Untuk informasi selengkapnya, lihat Memutar AWS Rahasia Secrets Manager Anda dan Rotating Secrets untuk Amazon DocumentDB di Panduan Pengguna AWS Secrets Manager.
- Gunakan Keamanan Lapisan Pengangkutan (TLS) dan enkripsi yang tidak bergerak untuk mengenkripsi data Anda.

Mengaudit acara Amazon DocumentDB

Dengan Amazon DocumentDB (dengan kompatibilitas MongoDB), Anda dapat mengaudit peristiwa yang dilakukan di klaster Anda. Contoh log acara termasuk upaya autentikasi yang berhasil dan gagal, membuang koleksi dalam basis data, atau membuat indeks. Secara default, pengauditan dinonaktifkan di Amazon DocumentDB dan mengharuskan Anda untuk menggunakan fitur ini.

Saat audit diaktifkan, Amazon DocumentDB mencatat Data Definition Language (DDL), Data Manipulation Language (DHTML), otentikasi, otorisasi, dan peristiwa manajemen pengguna ke Amazon Logs. CloudWatch Saat audit diaktifkan, Amazon DocumentDB mengekspor catatan audit klaster Anda (dokumen JSON) ke Amazon Logs. CloudWatch Anda dapat menggunakan CloudWatch Log Amazon untuk menganalisis, memantau, dan mengarsipkan peristiwa audit Amazon DocumentDB Anda.

Meskipun Amazon DocumentDB tidak membebankan biaya tambahan untuk mengaktifkan audit, Anda dikenakan tarif standar untuk penggunaan Log. CloudWatch Untuk informasi tentang harga CloudWatch Log, lihat CloudWatch harga Amazon.

Fitur audit Amazon DocumentDB jelas berbeda dari penggunaan sumber daya layanan yang dipantau. AWS CloudTrail CloudTrail merekam operasi yang dilakukan dengan AWS Command Line Interface (AWS CLI) atau Konsol Manajemen AWS pada sumber daya seperti cluster, instance, grup parameter, dan snapshot. Audit sumber daya aktif CloudTrail secara default dan tidak dapat

dinonaktifkan. Fitur audit Amazon DocumentDB adalah fitur keikutsertaan. Ini mencatat operasi yang terjadi dalam klaster Anda pada objek, seperti basis data, koleksi, indeks, dan pengguna.

Topik

- Acara yang didukung
- Mengaktifkan audit
- Menonaktifkan audit
- Mengakses acara audit Anda
- Memfilter acara audit DML.

Acara yang didukung

Audit Amazon DocumentDB mendukung kategori acara berikut:

- Data Definition Language (DDL) - termasuk operasi manajemen database, koneksi, manajemen pengguna, dan otorisasi.
- Data Manipulation Language read events (pembacaan DHTML) - termasuk `find()` dan berbagai operator agregasi, operator aritmatika, operator boolean, dan operator kueri baca lainnya.
- Manipulasi Data Bahasa menulis peristiwa (DHTML menulis) - termasuk `insert(), update(), delete()`, dan operator `bulkWrite()`

Jenis peristiwanya adalah sebagai berikut.

Jenis Acara	Kategori	Deskripsi
`authCheck`	Otorisasi	Kode hasil 0: Sukses
		Kode hasil 13: Upaya tidak sah untuk melakukan operasi.
`authenticate`	Koneksi	Upaya autentikasi yang berhasil atau gagal pada koneksi baru.

Jenis Acara	Kategori	Deskripsi
`auditConfigure`	DDL	Konfigurasi filter audit.
`createDatabase`	DDL	Pembuatan basis data baru.
`createCollection`	DDL	Pembuatan koleksi baru dalam basis data.
`createIndex`	DDL	Pembuatan indeks baru dalam koleksi.
`dropCollection`	DDL	Menghapus koleksi dalam basis data.
`dropDatabase`	DDL	Menghapus basis data.
`dropIndex`	DDL	Menghapus indeks dalam koleksi.
`modifyChangeStreams`	DDL	Ubah aliran telah dibuat.
`renameCollection`	DDL	Mengganti nama koleksi dalam database.
`createRole`	Manajemen Peran	Menciptakan peran.
`dropAllRolesFromDatabase`	Manajemen Peran	Menjatuhkan semua peran dalam database.
`dropRole`	Manajemen Peran	Menjatuhkan peran.

Jenis Acara	Kategori	Deskripsi
`grantPrivilegesToRole`	Manajemen Peran	Memberikan hak istimewa untuk suatu peran.
`grantRolesToRole`	Manajemen Peran	Memberikan peran ke peran yang ditentukan pengguna.
`revokePrivilegesFromRole`	Manajemen Peran	Membatalkan hak istimewa dari suatu peran.
`revokeRolesFromRole`	Manajemen Peran	Mencabut peran dari peran yang ditentukan pengguna.
`updateRole`	Manajemen Peran	Memperbarui peran.
`createUser`	Manajemen pengguna	Pembuatan pengguna baru.
`dropAllUsersFromDatabase`	Manajemen pengguna	Menghapus semua pengguna dalam basis data.
`dropUser`	Manajemen pengguna	Menghapus pengguna yang ada.
`grantRolesToUser`	Manajemen pengguna	Memberikan peran kepada pengguna.
`revokeRolesFromUser`	Manajemen pengguna	Mencabut peran dari pengguna.
`updateUser`	UserManagement	Memperbarui pengguna yang sudah ada.

Jenis Acara	Kategori	Deskripsi
`insert`	DMLTulis	Menyisipkan dokumen atau dokumen ke dalam koleksi.
`delete`	DMLTulis	Menghapus dokumen atau dokumen dari koleksi.
`update`	DMLTulis	Memodifikasi dokumen atau dokumen yang ada dalam koleksi.
`bulkWrite`	DMLTulis	Melakukan beberapa operasi tulis dengan kontrol untuk urutan eksekusi.
`setAuditConfig`	DMLTulis	Tetapkan filter baru untuk audit DML.
`count`	DMLdibaca	Mengembalikan jumlah dokumen yang akan cocok dengan query find () untuk koleksi atau tampilan.
`countDocuments`	DMLdibaca	Mengembalikan jumlah dokumen yang cocok dengan query untuk koleksi atau tampilan.

Jenis Acara	Kategori	Deskripsi
`find`	DMLdibaca	Memilih dokumen dalam koleksi atau tampilan dan mengembalikan kursor ke dokumen yang dipilih.
`getAuditConfig`	DMLdibaca	Ambil filter saat ini untuk audit DML.
`findAndModify`	DMLmembaca dan menulis DML	Memodifikasi dan mengembalikan satu dokumen.
`findOneAndDelete`	DMLmembaca dan menulis DML	Menghapus satu dokumen berdasarkan filter dan kriteria pengurutan, mengembalikan dokumen yang dihapus.
`findOneAndReplace`	DMLmembaca dan menulis DML	Menggantikan satu dokumen berdasarkan filter yang ditentukan.
`findOneAndUpdate`	DMLmembaca dan menulis DML	Memperbarui satu dokumen berdasarkan filter dan kriteria pengurutan.
`aggregate`	DMLmembaca dan menulis DML	Mendukung API dalam pipeline agregasi.

Jenis Acara	Kategori	Deskripsi
`distinct`	DMLdibaca	Menemukan nilai yang berbeda untuk bidang tertentu di satu koleksi atau tampilan dan mengembalikan hasilnya dalam array.

> **Note**
>
> Nilai dalam bidang parameter dokumen peristiwa DMLmemiliki batas ukuran 1KB. Amazon DocumentDB memotong nilainya jika melebihi 1KB.

> **Note**
>
> Acara penghapusan TTL tidak diaudit saat ini.

Mengaktifkan audit

Mengaktifkan audit pada klaster adalah proses dua langkah. Pastikan bahwa kedua langkah selesai, atau log audit tidak akan dikirim ke CloudWatch Log.

Langkah 1. Aktifkan parameter klaster audit_logs

Untuk mengaktifkan audit, Anda perlu memodifikasi `audit_logs` parameter dalam grup parameter. `audit_logs`adalah daftar peristiwa yang dibatasi koma untuk dicatat. Peristiwa harus ditentukan dalam huruf kecil dan seharusnya tidak ada spasi putih di antara elemen daftar.

Anda dapat mengatur nilai berikut untuk grup parameter:

Nilai	Deskripsi	
`ddl`	Menyetel ini akan memungkinkan audit	

Nilai	Deskripsi	
	untuk acara DDL seperti createDatabase, DropDatabase, createCollection, DropCollection, createIndex, DropIndex, AuthCheck, otentikasi, createUser, Dropuser, memberikan, mencabut, updateUser, dan drop RolesToUser RolesFromUser AllUsersFromDatabase	
`dml_read`	Menyetel ini akan memungkinkan audit untuk acara baca DMLseperti find, sort count, different, group, projecta, unwind, geoIntersects, GeoIntersects, GeoWithin dan operator kueri baca MongoDB lainnya.	
`dml_write`	Menyetel ini akan memungkinkan audit untuk acara penulisan DMLseperti insert (), update (), delete (), dan bulkWrite ()	

Nilai	Deskripsi	
`all`	Pengaturan ini akan memungkinkan audit untuk peristiwa database Anda, seperti kueri baca, kueri tulis, tindakan database, dan tindakan administrator.	
none	Pengaturan ini akan menonaktifkan audit	

Nilai	Deskripsi	
`enabled`(warisan)	Ini adalah pengaturan parameter lama yang setara dengan 'ddl'. Menyetel ini akan memungkinkan audit untuk acara DDL seperti createDatabase, DropDatabase, createCollection, DropCollection, createIndex, DropIndex, AuthCheck, authenticate, createUser, Dropuser, grant, revoke, updateUser, dan drop. RolesToUser RolesFromUser AllUsersFromDatabase Kami tidak menyarankan menggunakan pengaturan ini karena ini adalah pengaturan lama.	

Nilai	Deskripsi
`disabled` (warisan)	Ini adalah pengaturan parameter lama yang setara dengan 'tidak ada'. Kami tidak menyarankan menggunakan pengaturan ini karena ini adalah pengaturan lama.

> **Note**
>
> Nilai default untuk parameter cluster audit_logs adalah `none` (legacy ""`disabled`).

Anda juga dapat menggunakan nilai yang disebutkan di atas dalam kombinasi.

Nilai	Deskripsi
`ddl, dml_read`	Menyetel ini akan memungkinkan audit untuk acara DDL dan acara baca DML.
`ddl, dml_write`	Menyetel ini akan memungkinkan audit untuk acara DDL dan penulisan DML.
`dml_read, dml_write`	Menyetel ini akan memungkinkan audit untuk semua peristiwa DML.

> **Note**
>
> Anda tidak dapat memodifikasi grup parameter default.

Untuk informasi selengkapnya, lihat berikut ini:

- Membuat grup parameter cluster Amazon DocumentDB

  Setelah membuat grup parameter khusus, ubah dengan mengganti nilai parameter `audit_logs` menjadi `all`.
- Memodifikasi grup parameter cluster Amazon DocumentDB

Langkah 2. Aktifkan ekspor CloudWatch Log Amazon

Ketika nilai parameter `audit_logs` cluster adalah`enabled`,,, atau `ddl` `dml_read``dml_write`, Anda juga harus mengaktifkan Amazon DocumentDB untuk mengekspor log ke Amazon. CloudWatch Jika Anda menghilangkan salah satu dari langkah-langkah ini, log audit tidak akan dikirim ke CloudWatch.

Saat membuat klaster, melakukan pemulihan point-in-time-, atau memulihkan snapshot, Anda dapat mengaktifkan CloudWatch Log dengan mengikuti langkah-langkah berikut.

Using the Konsol Manajemen AWS

Untuk mengaktifkan Amazon DocumentDB mengekspor log CloudWatch untuk menggunakan konsol, lihat topik berikut:

- Saat membuat klaster — Di Membuat cluster dan instance primer menggunakan Konsol Manajemen AWS, lihat Membuat Klaster: Konfigurasi Tambahan (langkah 5, Ekspor log)
- Saat memodifikasi cluster yang ada — Memodifikasi cluster Amazon DocumentDB
- Saat melakukan pemulihan snapshot cluster — Memulihkan dari snapshot cluster
- Saat melakukan pemulihan point-in-time — Memulihkan ke titik waktu

Using the AWS CLI

Untuk mengaktifkan log audit saat membuat klaster baru

Kode berikut membuat cluster `sample-cluster` dan memungkinkan log CloudWatch audit.

Example

Untuk Linux, macOS, atau 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
```

Untuk 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
```

Untuk mengaktifkan log audit ketika memodifikasi klaster yang ada

Kode berikut memodifikasi cluster `sample-cluster` dan memungkinkan log CloudWatch audit.

Example

Untuk Linux, macOS, atau Unix:

```
aws docdb modify-db-cluster \
   --db-cluster-identifier sample-cluster \
   --cloudwatch-logs-export-configuration '{"EnableLogTypes":["audit"]}'
```

Untuk Windows:

```
aws docdb modify-db-cluster ^
   --db-cluster-identifier sample-cluster ^
```

```
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":["audit"]}'
```

Ouput dari operasi ini terlihat seperti berikut ini (format 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"
```

```
    }
}
```

Menonaktifkan audit

Anda dapat menonaktifkan audit dengan menonaktifkan ekspor CloudWatch Log dan menonaktifkan parameter. `audit_logs`

Menonaktifkan ekspor Log CloudWatch

Anda dapat menonaktifkan mengekspor log audit menggunakan file Konsol Manajemen AWS atau file. AWS CLI

Using the Konsol Manajemen AWS

Prosedur berikut menggunakan Konsol Manajemen AWS untuk menonaktifkan Amazon DocumentDB mengekspor log ke. CloudWatch

Untuk menonaktifkan log audit

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Klaster. Lalu, pilih tombol di sebelah kiri nama klaster yang log ekspornya ingin Anda nonaktifkan.
3. Pilih Tindakan, lalu pilih Modifikasi.
4. Gulir ke bawah ke bagian Ekspor log dan pilih Nonaktif.
5. Pilih Lanjutkan.
6. Tinjau perubahan Anda, lalu pilih kapan Anda ingin perubahan ini diterapkan ke klaster Anda.
   - Terapkan selama jendela pemeliharaan terjadwal berikutnya
   - Terapkan segera
7. Pilih Ubah klaster.

Using the AWS CLI

Kode berikut memodifikasi cluster `sample-cluster` dan menonaktifkan log CloudWatch audit.

Example

Untuk Linux, macOS, atau Unix:

```
aws docdb modify-db-cluster \
   --db-cluster-identifier sample-cluster \
   --cloudwatch-logs-export-configuration '{"DisableLogTypes":["audit"]}'
```

Untuk Windows:

```
aws docdb modify-db-cluster ^
   --db-cluster-identifier sample-cluster ^
   --cloudwatch-logs-export-configuration '{"DisableLogTypes":["audit"]}'
```

Output dari operasi ini terlihat seperti berikut (format 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"
            }
        ]
    }
}
```

Menonaktifkan parameter audit_logs

Untuk menonaktifkan parameter `audit_logs` pada klaster, Anda dapat memodifikasi klaster agar menggunakan grup parameter ketika nilai parameter `audit_logs` adalah `disabled`. Atau Anda dapat mengubah nilai parameter `audit_logs` di grup parameter klaster sehingga menjadi `disabled`.

Untuk informasi selengkapnya, lihat topik berikut:

- Memodifikasi cluster Amazon DocumentDB
- Memodifikasi grup parameter cluster Amazon DocumentDB

Mengakses acara audit Anda

Gunakan langkah-langkah berikut untuk mengakses peristiwa audit Anda di Amazon CloudWatch.

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pastikan Anda berada di Wilayah yang sama dengan klaster Amazon DocumentDB Anda.
3. Pilih Log di panel navigasi.
4. Untuk menemukan log audit pada klaster Anda, dari daftar yang tersedia, cari dan pilih **`/aws/docdb/`*`yourClusterName`*`/audit`**.

   Peristiwa audit untuk setiap instans Anda tersedia di bawah setiap nama instans masing-masing.

Memfilter acara audit DML.

Memulai dengan penyaringan audit DHTML

Peristiwa audit DMLdapat difilter sebelum ditulis ke Amazon CloudWatch. Untuk memanfaatkan fitur ini, log audit dan logging DMLharus diaktifkan. Amazon DocumentDB mendukung pemfilteran `atype` padacommand,,,, `user` dan. `namespace` `auditAuthorizationSuccess`

> **Note**
>
> Acara DDL tidak difilter.

Anda dapat mengaktifkan pemfilteran audit kapan saja dengan menentukan filter audit menggunakan`setAuditConfig`,`filter`, dan `auditAuthorizationSuccess` parameter dalam operasi: `db.adminCommand( { command } )`

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter:
         {
            //filter conditions
         },
      auditAuthorizationSuccess: true | false
   }
)
```

Anda juga dapat mengambil pengaturan filter audit dengan menjalankan perintah berikut:

```
db.admin.runCommand( { getAuditConfig: 1})
```

Persyaratan keamanan

Hanya database users/roles dengan tindakan istimewa yang `auditConfigure` dapat menjalankan perintah di atas `admindb` saat menyetel atau mencantumkan filter audit DHTML. Anda dapat menggunakan salah satu peran bawaan dari [`clusterAdmin`,`hostManager`,`root`] atau membuat peran khusus yang memiliki `auditConfigure` hak istimewa. Berikut ini adalah contoh penggunaan peran yang ada dengan hak `auditConfigure` istimewa dan contoh dengan peran khusus.

Pengguna dengan peran bawaan:

```
use admin
db.createUser(
  {
    user: "myClusterAdmin",
    pwd: "password123",
    roles: [ { role: "clusterAdmin", db: "admin" } ]
  }
)
```

Pengguna dengan peran khusus:

```
use admin
db.createRole(
   {
     role: "myRole",
     privileges: [
       { resource: { cluster: true }, actions: [ "auditConfigure" ] }
     ],
     roles: []
   }
)
db.createUser(
  {
    user: "myUser",
    pwd: "myPassword",
    roles: [ { role: "myRole", db: "admin" } ]
  }
)
```

Memfilter kasus penggunaan

Contoh: memfilter acara berdasarkan perintah

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {
        "$and": [
         {
            "param.command":
```

```
               {
                  $in: [ "find","count", "insert", "delete", "update",
 "findandmodify" ]
               }
         }
         ]
      },
      auditAuthorizationSuccess: true
   }
)
```

Contoh: memfilter acara berdasarkan nama pengguna

Dalam contoh ini, hanya pengguna “MyUser” yang akan dicatat:

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {
      "$and": [
         {
            "param.user":
               {
                  $in: [ "myUser" ]
               }
         }
         ]},
      auditAuthorizationSuccess: true})
```

Contoh: penyaringan oleh **`atype`**

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {atype: "authCheck"},
      auditAuthorizationSuccess: true
   })
```

> **Note**
>
> Semua log DML memiliki `authCheck` sebagai`atype`. Hanya DDL yang berbeda`atype`. Jika Anda menempatkan nilai selain `authCheck` di`filter`, itu tidak akan menghasilkan log masuk CloudWatch DML.

Contoh: memfilter dengan menggunakan beberapa filter yang digabungkan oleh operator

```
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**
>
> Di tingkat atas, hanya`$and`,`$or`, dan `$nor` didukung. Operator lain tidak didukung dan akan menyebabkan kesalahan.

Contoh: memfilter berdasarkan acara berdasarkan **`auditAuthorizationSuccess`**

Dalam filter ini, semua perintah yang telah berhasil melewati otorisasi tidak akan dicatat:

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {},
      auditAuthorizationSuccess: false
   }
)
```

Contoh: memfilter dengan **`$in`** dan kondisi **`$nin`**

Saat menggunakan keduanya di `$in` dan`$nin`, perintah tidak akan dicatat karena akan ada “dan” implisit di antara kondisi. Dalam contoh ini, regex akan memblokir `find` perintah sehingga tidak ada yang akan dicatat:

```
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})
```

Contoh: penyaringan oleh **`namespace`**

```
db.admin.runCommand(
```

```
   {
      setAuditConfig: 1,
      filter: {
      "$and": [
         {
            "param.ns":
               {
                  $in: [ "test.foo" ]
               }
         }
         ]},
      auditAuthorizationSuccess: true})
```

Contoh: mengatur ulang ke filter default

Menyetel ulang ke nilai default berarti bahwa setiap peristiwa audit DMLakan dicatat. Untuk mengatur ulang pemfilteran ke nilai default, jalankan perintah berikut:

```
db.admin.runCommand(
   {
      setAuditConfig: 1,
      filter: {},
      auditAuthorizationSuccess: true
   }
)
```

Amazon VPC dan Amazon DocumentDB

Amazon Virtual Private Cloud (Amazon VPC) memungkinkan Anda meluncurkan AWS sumber daya, seperti instans Amazon DocumentDB, ke cloud pribadi virtual (VPC).

Saat menggunakan VPC, Anda dapat mengontrol lingkungan jaringan virtual Anda. Anda dapat memilih rentang alamat IP Anda sendiri, membuat subnet, serta mengonfigurasi perutean dan daftar kontrol akses. Tidak ada biaya tambahan untuk menjalankan cluster Anda di VPC.

Akun memiliki VPC default. Semua cluster baru dibuat di VPC default kecuali Anda menentukan sebaliknya.

Topik

- Cluster DocumentDB dalam VPC

- Mengakses cluster Amazon DocumentDB di VPC
- Buat IPv4-only VPC untuk digunakan dengan cluster DocumentDB
- Buat VPC dual-stack untuk digunakan dengan cluster DocumentDB

Berikut ini, Anda dapat menemukan diskusi tentang fungsionalitas VPC yang relevan dengan cluster Amazon DocumentDB. Untuk informasi tentang Amazon VPC selengkapnya, lihat Panduan Pengguna Amazon VPC.

Cluster DocumentDB dalam VPC

Cluster Amazon DocumentDB Anda berada di cloud pribadi virtual (VPC). VPC adalah jaringan virtual yang secara logis terisolasi dari jaringan virtual lain di. AWS Cloud Amazon VPC memungkinkan Anda meluncurkan AWS sumber daya, seperti cluster Amazon DocumentDB atau instans Amazon EC2, ke dalam VPC. VPC dapat berupa VPC default dari akun Anda atau VPC yang Anda buat. Semua VPC dikaitkan dengan AWS akun Anda.

VPC default memiliki tiga subnet yang dapat digunakan untuk mengisolasi sumber daya di dalam VPC. VPC default juga memiliki gateway internet yang dapat digunakan untuk memberikan akses ke sumber daya di dalam VPC dari luar VPC.

Untuk skenario yang melibatkan cluster Amazon DocumentDB di VPC dan di luar VPC, lihat. Mengakses cluster Amazon DocumentDB di VPC

Topik

- Bekerja dengan cluster di VPC
- Bekerja dengan grup subnet
- Alamat IP Amazon DocumentDB
- Membuat cluster di VPC

Bekerja dengan cluster di VPC

Berikut adalah beberapa tips untuk bekerja dengan cluster di VPC:

- VPC Anda harus memiliki setidaknya dua subnet. Subnet ini harus berada di dua Availability Zone yang berbeda di Wilayah AWS mana Anda ingin menyebarkan cluster Anda. Subnet adalah segmen dari rentang alamat IP VPC yang dapat Anda tentukan dan yang dapat Anda gunakan untuk mengelompokkan cluster berdasarkan keamanan dan kebutuhan operasional Anda.

- Jika Anda ingin klaster Anda di VPC dapat diakses publik, pastikan untuk mengaktifkan atribut VPC dan. `DNS hostnames` `DNS resolution`
- VPC Anda harus memiliki grup subnet yang Anda buat. Anda membuat grup subnet dengan menentukan subnet yang Anda buat. Amazon DocumentDB memilih subnet dan alamat IP dalam grup subnet tersebut untuk diasosiasikan dengan instance utama di cluster Anda. Instance utama menggunakan Availability Zone yang berisi subnet.
- VPC Anda harus memiliki grup keamanan VPC yang memungkinkan akses ke cluster.

  Untuk informasi selengkapnya, lihat Mengakses cluster Amazon DocumentDB di VPC.
- Blok CIDR di setiap subnet Anda harus cukup besar untuk mengakomodasi alamat IP cadangan untuk Amazon DocumentDB untuk digunakan selama aktivitas pemeliharaan, termasuk failover dan penskalaan komputasi. Misalnya, rentang seperti 10.0.0. 0/24 dan 10.0.1. 0/24 biasanya cukup besar.
- VPC dapat memiliki `instance tenancy` atribut salah satu ataudefault. `dedicated` Semua VPC default memiliki `instance tenancy` atribut yang disetel ke default, dan VPC default dapat mendukung kelas instance apa pun.

  Jika Anda memilih untuk memiliki klaster di VPC khusus tempat `instance tenancy` atribut diseteldedicated, kelas instance klaster Anda harus menjadi salah satu jenis instans khusus Amazon EC2 yang disetujui. Misalnya, instans khusus `r5.large` EC2 sesuai dengan kelas `db.r5.large` instance. Untuk informasi tentang penyewaan instans di VPC, lihat instans Amazon EC2 di Panduan Pengguna Amazon Elastic Compute Cloud.

  Untuk informasi selengkapnya tentang jenis instans yang dapat berada dalam instans khusus, lihat instans khusus Amazon EC2 di halaman harga Amazon EC2.

  > **Note**
  >
  > Saat Anda menyetel `instance tenancy` atribut `dedicated` untuk cluster, itu tidak menjamin bahwa cluster akan berjalan pada host khusus.

Bekerja dengan grup subnet

Subnet adalah segmen dari rentang alamat IP VPC yang Anda tetapkan untuk mengelompokkan sumber daya Anda berdasarkan kebutuhan keamanan dan operasional. Grup subnet adalah kumpulan subnet (biasanya pribadi) yang Anda buat di VPC dan yang kemudian Anda tentukan

untuk cluster Anda. Dengan menggunakan grup subnet, Anda dapat menentukan VPC tertentu saat membuat cluster menggunakan atau AWS CLI Amazon DocumentDB API. Jika menggunakan konsol, Anda dapat memilih VPC dan grup subnet yang ingin Anda gunakan.

Setiap grup subnet harus memiliki subnet di setidaknya dua Availability Zone dalam satu tertentu. Wilayah AWS Saat membuat cluster di VPC, Anda memilih grup subnet untuk itu. Dari grup subnet, Amazon DocumentDB memilih subnet dan alamat IP dalam subnet tersebut untuk diasosiasikan dengan instance utama di cluster Anda. Database menggunakan Availability Zone yang berisi subnet. DocumentDB selalu memberikan alamat IP dari subnet yang memiliki ruang alamat IP gratis.

Subnet dalam grup subnet bersifat publik atau pribadi, tergantung pada konfigurasi yang Anda tetapkan untuk daftar kontrol akses jaringan (ACL jaringan) dan tabel perutean. Agar klaster dapat diakses publik, semua subnet dalam grup subnetnya harus bersifat publik. Jika subnet yang terkait dengan klaster yang dapat diakses publik berubah dari publik ke pribadi, hal itu dapat memengaruhi ketersediaan klaster.

Untuk membuat grup subnet yang mendukung mode dual-stack, pastikan setiap subnet yang Anda tambahkan ke grup subnet memiliki blok CIDR Internet Protocol versi 6 (IPv6) yang terkait dengannya. Untuk informasi selengkapnya, lihat Alamat IP Amazon DocumentDB dan dukungan IPv6 untuk VPC Anda di Panduan Pengguna Amazon Virtual Private Cloud.

Saat Amazon DocumentDB membuat cluster di VPC, Amazon DocumentDB menetapkan antarmuka jaringan ke cluster dengan menggunakan alamat IP dari grup subnet Anda. Namun, kami sangat menyarankan agar Anda menggunakan nama Domain Name System (DNS) untuk terhubung ke klaster Anda. Kami merekomendasikan hal ini karena alamat IP pokok berubah selama failover.

> **Note**
>
> Untuk setiap cluster yang Anda jalankan di VPC, pastikan untuk memesan setidaknya satu alamat di setiap subnet di grup subnet untuk digunakan oleh Amazon DocumentDB untuk tindakan pemulihan.

Subnet bersama

Anda dapat membuat cluster di VPC bersama.

Beberapa pertimbangan yang perlu diingat saat menggunakan VPC bersama:

- Anda dapat memindahkan cluster dari subnet VPC bersama ke subnet VPC non-bersama dan sebaliknya.
- Peserta dalam VPC bersama harus membuat grup keamanan di VPC untuk memungkinkan mereka membuat cluster.
- Pemilik dan peserta dalam VPC bersama dapat mengakses database dengan menggunakan kueri DocumentDB. Namun, hanya pembuat sumber daya yang dapat melakukan panggilan API apa pun pada sumber daya.

Alamat IP Amazon DocumentDB

Penentuan alamat IP memungkinkan sumber daya dalam VPC Anda berkomunikasi satu sama lain, dan dengan sumber daya melalui internet. Amazon DocumentDB mendukung protokol pengalamatan IPv4 dan IPv6. Secara default, Amazon DocumentDB dan Amazon VPC menggunakan protokol pengalamatan IPv4. Anda tidak dapat menonaktifkan perilaku ini. Saat Anda membuat VPC, pastikan untuk menentukan blok CIDR IPv4 (rentang alamat IPv4 privat). Anda dapat secara opsional menetapkan blok IPv6 CIDR ke VPC dan subnet Anda, dan menetapkan alamat IPv6 dari blok itu ke cluster di subnet Anda.

> **Note**
>
> Dual-stack mode (pengalamatan IPv6) hanya didukung di Amazon DocumentDB versi 4.0 dan 5.0.

Dukungan untuk protokol IPv6 memperbesar jumlah alamat IP yang didukung. Dengan menggunakan protokol IPv6, Anda memastikan bahwa Anda memiliki ketersediaan alamat yang memadai untuk menghadapi pertumbuhan internet di masa mendatang. Sumber daya DocumentDB baru dan yang sudah ada dapat menggunakan alamat IPv4 dan IPv6 dalam VPC Anda. Mengonfigurasi, mengamankan, dan menerjemahkan lalu lintas jaringan di antara dua protokol yang digunakan di berbagai bagian aplikasi dapat menimbulkan overhead operasional. Anda dapat melakukan standarisasi pada protokol IPv6 untuk sumber daya Amazon DocumentDB untuk menyederhanakan konfigurasi jaringan Anda.

Topik

- Alamat IPv4
- Alamat IPv6

- Dual-stack modus

Alamat IPv4

Saat membuat VPC, Anda harus menentukan rentang alamat IPv4 untuk VPC dalam bentuk blok CIDR, seperti `10.0.0.0/16`. Grup subnet mendefinisikan rentang alamat IP di blok CIDR ini yang dapat digunakan cluster. Alamat IP ini bisa bersifat privat atau publik.

Alamat IPv4 privat adalah alamat IP yang tidak dapat dijangkau dengan Internet. Anda dapat menggunakan alamat IPv4 pribadi untuk komunikasi antara cluster Anda dan sumber daya lainnya, seperti instans Amazon EC2, di VPC yang sama. Setiap cluster memiliki alamat IP pribadi untuk komunikasi di VPC.

Alamat IP publik adalah alamat IPv4 yang tidak dapat dijangkau dengan Internet. Pengalamatan IP publik tidak diperbolehkan untuk cluster DocumentDB Anda. Setiap alamat IP publik harus diselesaikan oleh gateway Internet dan oleh EC2 di subnet publik.

Untuk melihat cara membuat VPC hanya dengan alamat IPv4 pribadi yang dapat Anda gunakan untuk skenario Amazon DocumentDB umum, lihat. Buat IPv4-only VPC untuk digunakan dengan cluster DocumentDB

Alamat IPv6

Anda dapat secara opsional mengaitkan blok CIDR IPv6 ke VPC dan subnet Anda, serta menetapkan alamat IPv6 dari blok tersebut ke instans dalam VPC Anda. Setiap alamat IPv6 bersifat unik secara global.

Blok CIDR IPv6 untuk VPC Anda secara otomatis ditetapkan dari pool alamat IPv6 Amazon. Anda tidak dapat memilih sendiri rentang tersebut.

Saat menghubungkan ke alamat IPv6, pastikan bahwa kondisi berikut terpenuhi:

- Klien dikonfigurasi sehingga lalu lintas klien ke basis data melalui IPv6 diizinkan.
- Grup keamanan DocumentDB yang digunakan oleh cluster dikonfigurasi dengan benar sehingga klien ke lalu lintas database melalui IPv6 diizinkan.
- Tumpukan sistem operasi klien memungkinkan lalu lintas pada alamat IPv6, dan driver dan pustaka sistem operasi dikonfigurasi untuk memilih titik akhir cluster default yang benar (baik IPv4 atau IPv6).

Untuk informasi selengkapnya tentang IPv6, lihat Alamat IP di Panduan Pengguna Amazon Virtual Private Cloud.

Dual-stack modus

Ketika sebuah cluster dapat berkomunikasi melalui protokol pengalamatan IPv4 dan IPv6, itu berjalan dalam mode dual-stack. Jadi, sumber daya dapat berkomunikasi dengan cluster melalui IPv4, IPv6, atau keduanya. DocumentDB menonaktifkan akses Internet Gateway untuk titik akhir IPv6 dari cluster mode dual-stack pribadi. DocumentDB melakukan ini untuk memastikan bahwa titik akhir IPv6 Anda bersifat pribadi dan hanya dapat diakses dari dalam VPC Anda.

Topik

- Dual-stack mode dan grup subnet
- Bekerja dengan cluster mode dual-stack
- Memodifikasi IPv4-only cluster untuk menggunakan mode dual-stack
- Dual-stack mode Ketersediaan wilayah dan versi
- Batasan untuk kluster jaringan dual-stack

Dual-stack mode dan grup subnet

Untuk menggunakan mode dual-stack, pastikan bahwa setiap subnet dalam grup subnet yang Anda kaitkan dengan cluster memiliki blok CIDR IPv6 yang terkait dengannya. Anda dapat membuat grup subnet baru atau memodifikasi grup subnet yang ada untuk memenuhi persyaratan ini. Setelah cluster dalam mode dual-stack, klien dapat terhubung secara normal. Pastikan firewall keamanan klien dan grup keamanan cluster DocumentDB dikonfigurasi secara akurat untuk memungkinkan lalu lintas melalui IPv6. Untuk terhubung, klien menggunakan titik akhir cluster. Aplikasi klien dapat menentukan protokol mana yang lebih disukai saat terhubung ke basis data. Dalam mode dual-stack, cluster mendeteksi protokol jaringan pilihan klien, baik IPv4 atau IPv6, dan menggunakan protokol itu untuk koneksi.

Jika grup subnet berhenti mendukung mode dual-stack karena penghapusan subnet atau pemisahan CIDR, ada risiko status jaringan yang tidak kompatibel untuk cluster yang terkait dengan grup subnet. Selain itu, Anda tidak dapat menggunakan grup subnet saat membuat cluster mode dual-stack baru.

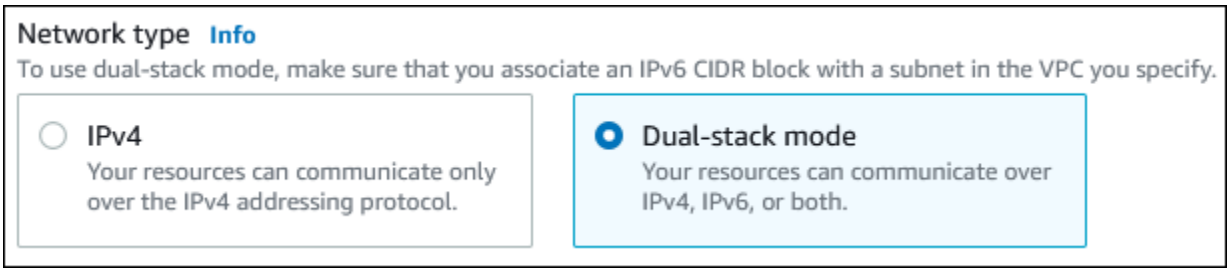
Untuk menentukan apakah grup subnet mendukung mode dual-stack dengan menggunakan Konsol Manajemen AWS, lihat tipe Jaringan pada halaman detail grup subnet. Untuk menentukan apakah grup subnet mendukung mode dual-stack dengan menggunakan AWS CLI, jalankan `describe-db-subnet-groups`perintah dan lihat `SupportedNetworkTypes` di output.

Replika baca diperlakukan sebagai cluster independen dan dapat memiliki tipe jaringan yang berbeda dari cluster utama. Jika Anda mengubah jenis jaringan klaster utama replika baca, replika baca tidak terpengaruh. Saat memulihkan cluster, Anda dapat mengembalikannya ke jenis jaringan apa pun yang didukung.

Bekerja dengan cluster mode dual-stack

Saat Anda membuat atau memodifikasi klaster, Anda dapat menentukan mode dual-stack untuk memungkinkan sumber daya Anda berkomunikasi dengan cluster Anda melalui IPv4, IPv6, atau keduanya.

Bila Anda menggunakan Konsol Manajemen AWS untuk membuat atau memodifikasi cluster, Anda dapat menentukan mode dual-stack di bagian Jenis jaringan. Gambar berikut menunjukkan bagian Jenis jaringan di konsol:



Saat Anda menggunakan AWS CLI untuk membuat atau memodifikasi cluster, atur `--network-type` opsi `DUAL` untuk menggunakan mode dual-stack. Saat Anda menggunakan DocumentDB API untuk membuat atau memodifikasi klaster, atur parameter `DUAL` untuk menggunakan `NetworkType` mode dual-stack. Jika mode dual-stack tidak didukung oleh versi mesin DocumentDB atau grup subnet yang ditentukan, kesalahan akan dikembalikan. `NetworkTypeNotSupported`

Untuk informasi selengkapnya tentang membuat klaster, lihatMembuat cluster Amazon DocumentDB. Untuk informasi selengkapnya tentang memodifikasi klaster, lihatMemodifikasi cluster Amazon DocumentDB.

Untuk menentukan apakah cluster berada dalam mode dual-stack dengan menggunakan konsol, lihat jenis Jaringan pada tab Konektivitas & keamanan untuk cluster.

Memodifikasi IPv4-only cluster untuk menggunakan mode dual-stack

Anda dapat memodifikasi IPv4-only cluster untuk menggunakan mode dual-stack. Untuk melakukannya, ubah jenis jaringan cluster.

Disarankan agar Anda mengubah jenis jaringan cluster Amazon DocumentDB Anda selama jendela pemeliharaan. Anda dapat mengatur jenis jaringan secara manual dengan menggunakan `modify-db-cluster`perintah.

Sebelum memodifikasi cluster untuk menggunakan mode dual-stack, pastikan bahwa grup subnetnya mendukung mode dual-stack. Jika grup subnet yang terkait dengan cluster tidak mendukung mode dual-stack, tentukan grup subnet berbeda yang mendukungnya saat Anda memodifikasi cluster. Memodifikasi grup subnet dari sebuah cluster dapat menyebabkan downtime.

Jika Anda memodifikasi grup subnet klaster sebelum Anda mengubah cluster untuk menggunakan mode dual-stack, pastikan bahwa grup subnet valid untuk cluster sebelum dan sesudah perubahan.

Kami menyarankan Anda menjalankan `ModifyDBCluster` panggilan hanya dengan `NetworkType` parameter yang disetel untuk mengubah jaringan `DUAL` ke mode dual-stack. Menambahkan parameter lain bersama `NetworkType` dengan panggilan API yang sama dapat mengakibatkan waktu henti. Untuk memodifikasi beberapa parameter, pastikan modifikasi jenis jaringan berhasil diselesaikan sebelum mengirim permintaan `ModifyDBCluster` lainnya dengan parameter lain.

Jika Anda tidak dapat terhubung ke cluster setelah perubahan, pastikan firewall keamanan klien dan database serta tabel rute dikonfigurasi secara akurat untuk memungkinkan lalu lintas ke database pada jaringan yang dipilih (baik IPv4 atau IPv6). Anda mungkin juga perlu memodifikasi parameter sistem operasi, pustaka, atau driver untuk terhubung menggunakan alamat IPv6.

Untuk memodifikasi IPv4-only cluster untuk menggunakan mode dual-stack

1. Ubah grup subnet untuk mendukung mode dual-stack, atau buat grup subnet yang mendukung mode dual-stack:

   a. Kaitkan blok CIDR IPv6 dengan VPC Anda.

      Untuk petunjuk, lihat Menambahkan atau menghapus blok CIDR ke VPC Anda di Panduan Pengguna Amazon VPC.

   b. Pasang blok IPv6 CIDR ke semua subnet di grup subnet Anda.

      Untuk petunjuk, lihat Menambahkan atau menghapus blok IPv6 CIDR ke subnet Anda di Panduan Pengguna Amazon Virtual Private Cloud.

   c. Konfirmasikan bahwa grup subnet mendukung mode dual-stack.

      Jika Anda menggunakan Konsol Manajemen AWS, pilih grup subnet, dan pastikan nilai tipe jaringan yang didukung adalah Ganda.

Jika Anda menggunakan AWS CLI, jalankan `describe-db-subnet-groups`perintah, dan pastikan bahwa `SupportedNetworkType` nilai untuk cluster adalah`Dual`.

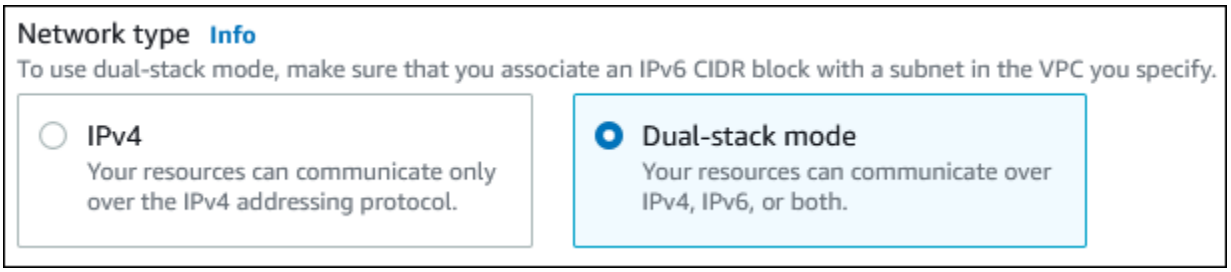
2. Ubah grup keamanan yang terkait dengan cluster untuk memungkinkan koneksi IPv6 ke database, atau buat grup keamanan baru yang memungkinkan koneksi IPv6.

   Untuk petunjuknya, lihat Aturan grup keamanan di Panduan Pengguna Amazon Virtual Private Cloud.

3. Ubah cluster untuk mendukung mode dual-stack. Untuk melakukannya, atur tipe Jaringan ke Dual-stack mode.

   Jika Anda menggunakan konsol, pastikan bahwa pengaturan berikut sudah benar:

   - Tipe jaringan - Dual-stack mode

   - Grup subnet - Grup subnet yang Anda konfigurasikan pada langkah sebelumnya
   - Grup keamanan — Keamanan yang Anda konfigurasikan pada langkah sebelumnya

   Jika Anda menggunakan AWS CLI, pastikan bahwa pengaturan berikut sudah benar:

   - `--network-type` — `dual`
   - `--db-subnet-group-name`— Grup subnet yang Anda konfigurasikan pada langkah sebelumnya
   - `--vpc-security-group-ids`— Grup keamanan VPC yang Anda konfigurasikan pada langkah sebelumnya

   Contoh:

   ```
   aws docdb modify-db-cluster --db-cluster-identifier <cluster-name> --network-type
    "DUAL"
   ```

4. Konfirmasikan bahwa cluster mendukung mode dual-stack.

Jika Anda menggunakan konsol, pilih tab Konektivitas & keamanan untuk cluster dan pastikan bahwa nilai tipe Jaringan adalah Dual-stack mode.

Jika Anda menggunakan AWS CLI, jalankan `describe-db-cluster` perintah, dan pastikan bahwa `NetworkType` nilai untuk cluster adalah`dual`.

Jalankan `dig` perintah pada titik akhir cluster untuk mengidentifikasi alamat IPv6 yang terkait dengannya:

```
dig <db-cluster-endpoint> AAAA
```

Gunakan titik akhir cluster, bukan alamat IPv6, untuk terhubung ke cluster.

Dual-stack mode Ketersediaan wilayah dan versi

Ketersediaan dan dukungan fitur bervariasi di seluruh Wilayah AWS.

Dukungan wilayah

Mode tumpukan ganda untuk Amazon DocumentDB saat ini tersedia di semua komersial. Wilayah AWS

Dukungan versi

Dual-stack mode didukung pada Amazon DocumentDB versi 4.0, 5.0 dan 8.0. Jika Anda tidak dapat mengakses mode dual-stack pada salah satu versi ini, pastikan Anda menjalankan versi patch mesin terbaru di cluster Anda.

Batasan untuk kluster jaringan dual-stack

Batasan berikut berlaku untuk kluster jaringan dual-stack:

- Cluster tidak dapat menggunakan protokol IPv6 secara eksklusif. IPv4 dapat digunakan secara tersendiri, tetapi protokol IPv6 harus digunakan bersama IPv4 (mode tumpukan ganda).
- Amazon DocumentDB tidak mendukung subnet IPv6 asli.
- Cluster yang menggunakan mode dual-stack harus bersifat pribadi. Hal ini tidak boleh dapat diakses oleh publik.

Membuat cluster di VPC

Prosedur berikut membantu Anda membuat cluster di VPC. Untuk menggunakan VPC default, Anda dapat mulai dengan langkah 2, dan menggunakan VPC dan grup subnet yang telah dibuat untuk Anda. Anda juga dapat membuat VPC tambahan, jika diperlukan.

> **Note**
>
> Jika Anda ingin klaster Anda di VPC dapat diakses publik, Anda harus memperbarui informasi DNS untuk VPC dengan mengaktifkan atribut VPC dan. `DNS hostnames` `DNS resolution` Untuk informasi tentang memperbarui DNS untuk instance VPC, lihat Melihat dan memperbarui atribut DNS untuk VPC Anda di Panduan Pengguna Amazon Virtual Private Cloud.

Ikuti langkah-langkah ini untuk membuat cluster di VPC:

- Langkah 1: Buat VPC
- Langkah 2: Buat grup subnet
- Langkah 3: Buat grup keamanan VPC
- Langkah 4: Buat cluster di VPC

Langkah 1: Buat VPC

Buat VPC dengan subnet di setidaknya dua Zona Ketersediaan. Anda menggunakan subnet ini ketika Anda membuat grup subnet. Jika Anda memiliki VPC default, subnet secara otomatis dibuat untuk Anda dalam setiap Zona Ketersediaan di Wilayah AWS.

Untuk informasi selengkapnyaBuat IPv4-only VPC untuk digunakan dengan cluster DocumentDB, lihat, atau lihat Membuat VPC di Panduan Pengguna Amazon Virtual Private Cloud.

Langkah 2: Buat grup subnet

Grup subnet adalah kumpulan subnet (biasanya pribadi) yang Anda buat untuk VPC dan yang kemudian Anda tentukan untuk cluster Anda. Grup subnet memungkinkan Anda untuk menentukan VPC tertentu ketika Anda membuat cluster menggunakan AWS CLI atau DocumentDB API. Jika Anda menggunakan Konsol Manajemen AWS, Anda bisa memilih VPC dan subnet yang ingin Anda gunakan. Setiap grup subnet harus memiliki setidaknya satu subnet di setidaknya dua Availability

Zone di. Wilayah AWS Sebagai praktik terbaik, setiap grup subnet harus memiliki setidaknya satu subnet untuk setiap Availability Zone di. Wilayah AWS

Agar cluster dapat diakses publik, subnet dalam grup subnet harus memiliki gateway internet. Untuk informasi selengkapnya tentang gateway internet untuk subnet, lihat Mengaktifkan akses internet untuk VPC menggunakan gateway internet di Panduan Pengguna Amazon Virtual Private Cloud.

> **Note**
>
> Grup subnet untuk zona lokal hanya dapat memiliki satu subnet.

Saat Anda membuat cluster di VPC, Anda dapat memilih grup subnet. Amazon DocumentDB memilih subnet dan alamat IP (dalam subnet itu) untuk dikaitkan dengan cluster Anda. Jika tidak ada grup subnet, Amazon DocumentDB akan membuat grup subnet default saat Anda membuat klaster. DocumentDB membuat dan mengaitkan Antarmuka Jaringan Elastis ke cluster Anda dengan alamat IP tersebut. Cluster menggunakan Availability Zone yang berisi subnet.

Pada langkah ini, Anda membuat grup subnet dan menambahkan subnet yang Anda buat untuk VPC Anda.

Untuk membuat grup subnet

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Grup Subnet.
3. Pilih Buat.
4. Untuk Nama, ketik nama grup subnet Anda.
5. Untuk Deskripsi, ketikkan deskripsi untuk grup subnet Anda.
6. Di bagian Tambahkan subnet, untuk VPC, pilih VPC default atau VPC yang Anda buat. Kemudian pilih Availability Zones yang menyertakan subnet dari Availability Zones, lalu pilih subnet dari Subnet.
7. Pilih Buat.

   Grup subnet baru Anda muncul di daftar grup Subnet di konsol DocumentDB. Anda dapat memilih grup subnet untuk melihat detail, termasuk semua subnet yang terkait dengan grup, di panel detail di bagian bawah jendela.

Langkah 3: Buat grup keamanan VPC

Sebelum Anda membuat cluster Anda, buat grup keamanan VPC untuk mengaitkannya. Jika Anda tidak membuat grup keamanan VPC, Anda dapat menggunakan grup keamanan default saat membuat klaster. Untuk petunjuk tentang cara membuat grup keamanan untuk klaster Anda, lihatBuat IPv4-only VPC untuk digunakan dengan cluster DocumentDB, atau lihat Mengontrol lalu lintas ke AWS sumber daya Anda menggunakan grup keamanan di Panduan Pengguna Amazon Virtual Private Cloud.

Langkah 4: Buat cluster di VPC

Pada langkah ini, Anda membuat cluster menggunakan nama VPC, grup subnet, dan grup keamanan VPC yang Anda buat di langkah sebelumnya.

> **Note**
>
> Jika Anda ingin klaster Anda di VPC dapat diakses publik, Anda harus mengaktifkan atribut VPC dan. `DNS hostnames` `DNS resolution` Untuk informasi selengkapnya, lihat Melihat dan memperbarui atribut DNS untuk VPC Anda di Panduan Pengguna Amazon Virtual Private Cloud.

Untuk detail tentang cara membuat cluster, lihatMembuat cluster Amazon DocumentDB.

Saat diminta di bagian Konektivitas, masukkan nama VPC, grup subnet, dan grup keamanan VPC.

> **Note**
>
> Memperbarui VPC saat ini tidak didukung untuk cluster DocumentDBDocumentDB.

Mengakses cluster Amazon DocumentDB di VPC

Amazon DocumentDB mendukung skenario berikut untuk mengakses cluster di VPC:

Topik

- Cluster di VPC yang diakses oleh instans Amazon EC2 di VPC yang sama
- Cluster di VPC yang diakses oleh instans Amazon EC2 di VPC yang berbeda

Cluster di VPC yang diakses oleh instans Amazon EC2 di VPC yang sama

Penggunaan umum cluster dalam VPC adalah untuk berbagi data dengan server aplikasi yang berjalan di instans Amazon EC2 di VPC yang sama.

Cara paling sederhana untuk mengelola akses antara instans EC2 dan klaster di VPC yang sama adalah dengan melakukan tindakan berikut:

- Buat grup keamanan VPC untuk kluster Anda. Grup keamanan ini dapat digunakan untuk membatasi akses ke cluster. Misalnya, Anda dapat membuat aturan kustom untuk grup keamanan ini. Ini mungkin memungkinkan akses TCP menggunakan port yang Anda tetapkan ke cluster saat Anda membuatnya dan alamat IP yang Anda gunakan untuk mengakses klaster untuk pengembangan atau tujuan lain.
- Buat grup keamanan VPC untuk memasukkan instans EC2 (server web dan klien) Anda. Grup keamanan ini dapat, jika diperlukan, mengizinkan akses ke instans EC2 dari internet dengan menggunakan tabel perutean VPC. Sebagai contoh, Anda dapat menetapkan aturan pada kelompok keamanan ini untuk memungkinkan akses TCP ke instans EC2 melalui port 22.
- Buat aturan khusus di grup keamanan untuk kluster Anda yang memungkinkan koneksi dari grup keamanan yang Anda buat untuk instans EC2 Anda. Aturan-aturan ini memungkinkan setiap anggota grup keamanan untuk mengakses cluster.

Ada subnet publik dan privat tambahan di Zona Ketersediaan terpisah. Grup subnet DocumentDB memerlukan subnet di setidaknya dua Availability Zone. Subnet tambahan memudahkan untuk beralih ke penyebaran cluster multi-AZ di masa depan.

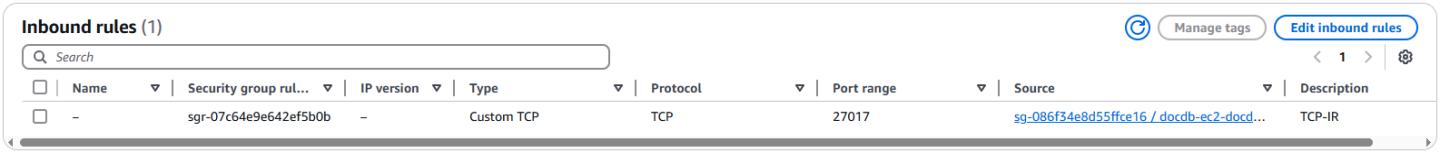
Untuk petunjuk tentang cara membuat VPC dengan subnet publik dan pribadi untuk skenario ini, lihat. Buat IPv4-only VPC untuk digunakan dengan cluster DocumentDB

> **Tip**
>
> Anda dapat mengatur konektivitas jaringan antara instans Amazon EC2 dan cluster DocumentDB secara otomatis saat membuat cluster. Untuk informasi selengkapnya, lihat Hubungkan Amazon EC2 secara otomatis.

Untuk membuat aturan dalam grup keamanan VPC yang memungkinkan koneksi dari grup keamanan lain, lakukan hal berikut:

1. Masuk ke Konsol Manajemen AWS dan buka konsol VPC Amazon di. https://console.aws.amazon.com/vpc
2. Di panel navigasi, cari dan pilih Grup keamanan.
3. Pilih atau buat grup keamanan yang ingin Anda berikan aksesnya ke anggota grup keamanan lain. Ini adalah grup keamanan yang Anda gunakan untuk cluster Anda. Pilih tab Aturan masuk, lalu pilih Edit aturan masuk.
4. Di halaman Edit aturan masuk, pilih Tambahkan aturan.
5. Untuk Jenis, pilih entri yang sesuai dengan port yang Anda gunakan saat membuat klaster, seperti TCP Kustom.
6. Di bidang Sumber, mulailah mengetik ID grup keamanan, yang mencantumkan grup keamanan yang cocok. Pilih grup keamanan dengan anggota yang ingin diberi akses ke sumber daya yang dilindungi oleh grup keamanan ini. Dalam skenario sebelumnya, ini adalah grup keamanan yang Anda gunakan untuk instans EC2 Anda.
7. Jika diperlukan, ulangi langkah-langkah untuk protokol TCP dengan membuat aturan dengan Semua TCP sebagai Jenis dan grup keamanan Anda di bidang Sumber. Jika Anda ingin menggunakan protokol UDP, buat aturan dengan Semua UDP sebagai Jenis dan grup keamanan Anda di Sumber.
8. Pilih Simpan aturan.

Layar berikut menunjukkan aturan masuk dengan grup keamanan sebagai sumbernya.



Untuk informasi selengkapnya tentang menghubungkan ke cluster dari instans EC2 Anda, lihatHubungkan Amazon EC2 secara otomatis.

Cluster di VPC yang diakses oleh instans Amazon EC2 di VPC yang berbeda

Ketika cluster Anda berada di VPC yang berbeda dari instans EC2 yang Anda gunakan untuk mengaksesnya, Anda dapat menggunakan VPC peering untuk mengakses cluster.

Koneksi peering VPC adalah koneksi jaringan antara dua VPC yang memungkinkan Anda merutekan lalu lintas di antara keduanya menggunakan alamat IP privat. Sumber daya dalam VPC mana pun dapat berkomunikasi satu sama lain seolah-olahnya ada di jaringan yang sama. Anda dapat membuat koneksi peering VPC antara VPC Anda sendiri, dengan VPC di akun lain AWS , atau

dengan VPC di akun lain. Wilayah AWS Untuk mempelajari selengkapnya tentang peering VPC, lihat Peering VPC dalam Panduan Pengguna Amazon Virtual Private Cloud.

Buat IPv4-only VPC untuk digunakan dengan cluster DocumentDB

Skenario umum termasuk cluster di cloud pribadi virtual (VPC) berdasarkan layanan Amazon VPC. Misalnya, VPC ini dapat berbagi data dengan layanan atau aplikasi yang berjalan di VPC yang sama. Dalam topik ini, Anda membuat VPC untuk skenario ini.

Topik

- Langkah 1: Buat VPC dengan subnet pribadi dan publik
- Langkah 2: Buat grup keamanan VPC untuk aplikasi publik
- Langkah 3: Buat grup keamanan VPC untuk cluster pribadi
- Langkah 4: Buat grup subnet
- Menghapus VPC

Cluster Anda harus tersedia hanya untuk aplikasi Anda, dan bukan untuk internet publik. Dengan demikian, Anda membuat VPC dengan subnet publik maupun privat. Aplikasi ini di-host di subnet publik, sehingga dapat menjangkau internet publik. Cluster di-host di subnet pribadi. Aplikasi dapat terhubung ke cluster karena di-host dalam VPC yang sama. Tetapi cluster tidak tersedia untuk internet publik, memberikan keamanan yang lebih besar.

Prosedur dalam topik ini mengonfigurasi subnet publik dan pribadi tambahan di Availability Zone terpisah. Subnet ini tidak digunakan oleh prosedur. Grup subnet DocumentDB memerlukan subnet di setidaknya dua Availability Zone. Subnet tambahan membuatnya lebih mudah untuk mengkonfigurasi lebih dari satu instance DocumentDB.

Topik ini menjelaskan konfigurasi VPC untuk cluster Amazon DocumentDB. Untuk informasi tentang Amazon VPC selengkapnya, lihat Panduan Pengguna Amazon VPC.

> **Tip**
>
> Anda dapat mengatur konektivitas jaringan antara instans Amazon EC2 dan cluster DocumentDB secara otomatis saat membuat cluster. Konfigurasi jaringan mirip dengan yang dijelaskan dalam skenario ini. Untuk informasi selengkapnya, lihat Hubungkan Amazon EC2 secara otomatis.

Langkah 1: Buat VPC dengan subnet pribadi dan publik

Gunakan prosedur berikut untuk membuat VPC dengan subnet publik maupun privat.

Untuk membuat VPC dan subnet

1. Buka konsol VPC Amazon di. https://console.aws.amazon.com/vpc
2. Di pojok kanan atas Konsol Manajemen AWS, pilih Wilayah untuk membuat VPC Anda. Contoh ini menggunakan Wilayah AS Barat (Oregon).
3. Di sudut kiri atas, pilih Dasbor VPC . Untuk mulai membuat VPC, pilih Buat VPC.
4. Agar Sumber daya dapat dibuat di bagian pengaturan VPC, pilih VPC dan lainnya.
5. Untuk pengaturan VPC, atur nilai-nilai ini:
   - Generasi otomatis tag nama - **`example`**
   - Blok IPv4 CIDR - **`10.0.0.0/16`**
   - Blok IPv6 CIDR - Tidak ada blok CIDR IPv6
   - Penyewaan - Default
   - Jumlah Availability Zone (AZ) — 2
   - Kustomisasi AZ - Pertahankan nilai default
   - Jumlah subnet publik — 2
   - Jumlah subnet pribadi — 2
   - Kustomisasi subnet blok CIDR - Pertahankan nilai default
   - Gateway NAT ($) - Tidak ada
   - Titik akhir VPC - Tidak ada
   - Opsi DNS - Pertahankan nilai default
6. Pilih Buat VPC.

Langkah 2: Buat grup keamanan VPC untuk aplikasi publik

Selanjutnya, buat grup keamanan untuk akses publik. Untuk terhubung ke instans EC2 publik di VPC Anda, tambahkan aturan masuk ke grup keamanan VPC Anda. Ini memungkinkan lalu lintas untuk terhubung dari internet.

Untuk membuat grup keamanan VPC

1. Buka konsol VPC Amazon di. https://console.aws.amazon.com/vpc
2. Pilih Dasbor VPC, pilih Grup Keamanan, lalu pilih Buat grup keamanan.
3. Di halaman Buat grup keamanan, atur nilai ini:

   - Nama grup keamanan - **`example-securitygroup`**
   - Deskripsi — **`Application security group`**
   - VPC — Pilih VPC yang Anda buat sebelumnya, misalnya: vpc-example.
4. Tambahkan aturan masuk ke grup keamanan.

   a. Tentukan alamat IP yang akan digunakan untuk terhubung ke instans EC2 di VPC Anda menggunakan Secure Shell (SSH). Untuk menentukan alamat IP publik Anda, di jendela atau tab browser yang berbeda, Anda dapat menggunakan layanan di https://checkip.amazonaws.com. Contoh alamat IP adalah `203.0.113.25/32`.

   Dalam banyak kasus, Anda dapat menghubungkan melalui penyedia layanan Internet (ISP) atau dari belakang firewall Anda tanpa alamat IP statis. Jika demikian, temukan rentang alamat IP yang digunakan oleh komputer klien.

   > ⚠ Warning
   >
   > Jika Anda menggunakan `0.0.0.0/0` untuk akses SSH, Anda memungkinkan semua alamat IP untuk mengakses instans publik Anda menggunakan SSH. Hal ini dapat diterima untuk waktu yang singkat di lingkungan pengujian, tetapi tidak aman untuk lingkungan produksi. Dalam produksi, Anda hanya dapat memberikan otorisasi pada alamat IP atau rentang alamat tertentu saja untuk mengakses instans-instans Anda menggunakan SSH.

   b. Di bagian Aturan masuk, pilih Tambahkan aturan.

   c. Atur nilai berikut untuk aturan masuk baru Anda yang akan mengizinkan akses SSH ke instans Amazon EC2 Anda. Setelah Anda melakukan ini, Anda dapat terhubung ke instans EC2 Anda untuk menginstal aplikasi dan utilitas lainnya. Anda juga terhubung ke instans EC2 untuk mengunggah konten untuk aplikasi Anda.

      - Jenis - **`SSH`**
      - Sumber — Alamat IP atau rentang yang Anda buat dari Langkah a, misalnya: **`203.0.113.25/32`**

   d. Pilih Tambahkan aturan.

e. Tetapkan nilai berikut untuk aturan masuk baru Anda untuk mengizinkan akses HTTP ke aplikasi Anda:

- Jenis - **`HTTP`**
- Sumber - **`0.0.0.0/0`**

5. Untuk membuat grup keamanan, pilih Buat grup keamanan.

   Perhatikan ID grup keamanan karena Anda membutuhkannya nanti di prosedur lain.

Langkah 3: Buat grup keamanan VPC untuk cluster pribadi

Untuk menjaga klaster Anda tetap pribadi, buat grup keamanan kedua untuk akses pribadi. Untuk terhubung ke kluster pribadi di VPC Anda, Anda menambahkan aturan masuk ke grup keamanan VPC Anda yang mengizinkan lalu lintas dari aplikasi Anda saja.

Untuk membuat grup keamanan VPC

1. Buka konsol VPC Amazon di. https://console.aws.amazon.com/vpc
2. Pilih Dasbor VPC, pilih Grup Keamanan, lalu pilih Buat grup keamanan.
3. Di halaman Buat grup keamanan, atur nilai ini:
   - Nama grup keamanan - **`example-securitygroup`**
   - Deskripsi — **`Instance security group`**
   - VPC — Pilih VPC yang Anda buat sebelumnya, misalnya: vpc-example
4. Tambahkan aturan masuk ke grup keamanan.

   a. Di bagian Aturan masuk, pilih Tambahkan aturan.

   b. Tetapkan nilai berikut untuk aturan masuk baru Anda untuk mengizinkan lalu lintas DocumentDB di port 27017 dari instans Amazon EC2 Anda. Setelah Anda melakukan ini, Anda dapat terhubung dari aplikasi Anda ke cluster Anda. Dengan demikian, Anda dapat menyimpan dan mengambil data dari aplikasi Anda ke database Anda.

   - Jenis - **`Custom TCP`**
   - Sumber — Pengidentifikasi grup keamanan aplikasi yang Anda buat sebelumnya dalam topik ini, misalnya: sg-9edd5cfb.

   c. Pilih Tambahkan aturan.

d. Tetapkan nilai berikut untuk aturan masuk baru Anda untuk mengizinkan akses HTTP ke aplikasi Anda:

- Jenis - **`HTTP`**
- Sumber - **`0.0.0.0/0`**

5. Untuk membuat grup keamanan, pilih Buat grup keamanan.

Langkah 4: Buat grup subnet

Grup subnet adalah kumpulan subnet yang Anda buat di VPC dan kemudian Anda tentukan untuk cluster Anda. Grup subnet memungkinkan Anda untuk menentukan VPC tertentu saat membuat cluster.

Untuk membuat grup subnet

1. Identifikasi subnet privat untuk basis data Anda di VPC.

   a. Buka konsol VPC Amazon di. https://console.aws.amazon.com/vpc

   b. Pilih Dasbor VPC, lalu pilih Subnet.

   c. Perhatikan ID subnet dari subnet yang Anda buat di Langkah 1 bernama, misalnya: example-subnet-private1-us-west-2a dan example-subnet-private2-us-west-2b. Anda memerlukan ID subnet saat membuat grup subnet Anda.

2. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb

   Pastikan Anda terhubung ke konsol Amazon DocumentDB, bukan ke konsol Amazon VPC.

3. Di panel navigasi, pilih Grup Subnet.
4. Pilih Buat.
5. Pada halaman Create subnet group, tetapkan nilai-nilai ini di bagian Detail grup Subnet:

   - Nama - **`example-db-subnet-group`**
   - Deskripsi — **`Instance security group`**

6. Di bagian Tambahkan subnet, atur nilai-nilai ini:

   - VPC — Pilih VPC yang Anda buat sebelumnya, misalnya: vpc-example

- Availability Zones — Pilih kedua Availability Zone yang dibuat pada Langkah 1. Contoh: us-west-2a dan us-west-2b
- Subnet — Pilih subnet pribadi yang Anda buat di Langkah 1.

7. Pilih Buat.

Grup subnet baru Anda muncul di daftar grup subnet di konsol DocumentDB. Anda dapat memilih grup subnet untuk melihat detail di panel detail. Detail ini mencakup semua subnet yang terkait dengan grup.

> **Note**
>
> Jika Anda membuat VPC ini untuk mengaitkannya dengan cluster DocumentDB, buat cluster dengan mengikuti instruksi di. Membuat cluster Amazon DocumentDB

Menghapus VPC

Anda dapat menghapus VPC dan sumber daya lain yang digunakan di dalamnya, jika tidak lagi diperlukan.

> **Note**
>
> Jika Anda menambahkan sumber daya di VPC yang Anda buat dalam topik ini, Anda mungkin perlu menghapusnya sebelum dapat menghapus VPC. Misalnya, sumber daya ini mungkin menyertakan instans Amazon EC2 atau cluster DocumentDB. Untuk informasi selengkapnya, lihat Menghapus VPC di Panduan Pengguna Amazon VPC.

Untuk menghapus VPC dan sumber daya terkait

1. Hapus grup subnet:

   a. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
   b. Di panel navigasi, pilih Grup Subnet.
   c. Pilih grup subnet yang ingin Anda hapus, seperti example-db-subnet-group.
   d. Pilih Hapus, lalu pilih Hapus di jendela konfirmasi.

2. Catat ID VPC:

    a. Buka konsol VPC Amazon di. https://console.aws.amazon.com/vpc
    b. Pilih Dasbor VPC, lalu pilih VPC Anda.
    c. Dalam daftar, identifikasi VPC yang Anda buat, seperti vpc-example.
    d. Catat ID VPC dari VPC yang Anda buat. ID VPC diperlukan di langkah berikutnya.

3. Hapus grup keamanan:

    a. Buka konsol VPC Amazon di. https://console.aws.amazon.com/vpc
    b. Pilih Dasbor VPC, lalu pilih Grup Keamanan.
    c. Pilih grup keamanan untuk klaster Amazon DocumentDB, seperti example-securitygroup.
    d. Untuk Tindakan, pilih Hapus grup keamanan, lalu pilih Hapus pada dialog konfirmasi.
    e. Kembali ke halaman Grup Keamanan, pilih grup keamanan untuk instans Amazon EC2, seperti example-securitygroup.
    f. Untuk Tindakan, pilih Hapus grup keamanan, lalu pilih Hapus pada dialog konfirmasi.

4. Hapus VPC:

    a. Buka konsol VPC Amazon di. https://console.aws.amazon.com/vpc
    b. Pilih Dasbor VPC, lalu pilih VPC Anda.
    c. Pilih VPC yang ingin Anda hapus, seperti vpc-example.
    d. Untuk Tindakan, pilih Hapus VPC.

        Halaman konfirmasi menunjukkan sumber daya lain yang terkait dengan VPC yang juga akan dihapus, termasuk subnet yang terkait dengannya.

    e. Pada dialog konfirmasi, masukkan**`delete`**, lalu pilih Hapus.

Buat VPC dual-stack untuk digunakan dengan cluster DocumentDB

Skenario umum termasuk cluster di cloud pribadi virtual (VPC) berdasarkan layanan Amazon VPC. VPC ini membagikan data dengan instans Amazon EC2 publik yang dijalankan di VPC yang sama. Dalam topik ini, Anda membuat VPC untuk skenario ini.

Topik

- Langkah 1: Buat VPC dengan subnet pribadi dan publik
- Langkah 2: Buat grup keamanan VPC untuk instans Amazon EC2 publik

- Langkah 3: Buat grup keamanan VPC untuk cluster pribadi
- Langkah 4: Buat grup subnet
- Langkah 5: Buat instans Amazon EC2 dalam mode dual-stack
- Langkah 6: Buat cluster dalam mode dual-stack
- Langkah 7: Hubungkan ke instans Amazon EC2 dan kluster DB
- Hapus VPC

Dalam prosedur ini, Anda membuat VPC untuk skenario ini yang bekerja dengan database yang berjalan dalam mode dual-stack. Dual-stack mode memungkinkan koneksi melalui protokol pengalamatan IPv6. Untuk informasi selengkapnya tentang penentuan alamat IP, lihat Alamat IP Amazon DocumentDB.

Dual-stack kluster jaringan didukung di sebagian besar wilayah. Untuk informasi selengkapnya, lihat Dual-stack mode Ketersediaan wilayah dan versi. Untuk melihat batasan mode tumpukan ganda, lihat Batasan untuk kluster jaringan dual-stack.

Topik ini dan IPv4-only topik membuat subnet publik dan pribadi di VPC yang sama. Untuk informasi tentang membuat klaster Amazon DocumentDB dalam satu VPC dan instans Amazon EC2 di VPC yang berbeda, lihat. Mengakses cluster Amazon DocumentDB di VPC

Cluster DocumentDB Anda harus tersedia hanya untuk instans Amazon EC2 Anda, dan bukan untuk internet publik. Dengan demikian, Anda membuat VPC dengan subnet publik maupun privat. Instans EC2 di-host di subnet publik, sehingga dapat menjangkau internet publik. Cluster di-host di subnet pribadi. Instans EC2 dapat terhubung ke cluster karena di-host dalam VPC yang sama. Namun, cluster tidak tersedia untuk internet publik, memberikan keamanan yang lebih besar.

Prosedur dalam topik ini mengonfigurasi subnet publik dan pribadi tambahan di Availability Zone terpisah. Subnet ini tidak digunakan oleh prosedur. Grup subnet DocumentDB memerlukan subnet di setidaknya dua Availability Zone. Subnet tambahan memudahkan untuk mengkonfigurasi lebih dari satu instance DocumentDB.

Untuk membuat cluster yang menggunakan mode dual-stack, tentukan Dual-stack mod e untuk pengaturan tipe Jaringan. Anda juga dapat memodifikasi cluster dengan pengaturan yang sama. Untuk informasi selengkapnya tentang membuat klaster, lihatMembuat cluster Amazon DocumentDB. Untuk informasi selengkapnya tentang cara mengubah klaster DB, lihat Memodifikasi cluster Amazon DocumentDB.

Topik ini menjelaskan konfigurasi VPC untuk cluster Amazon DocumentDB. Untuk informasi tentang Amazon VPC selengkapnya, lihat Panduan Pengguna Amazon VPC.

Langkah 1: Buat VPC dengan subnet pribadi dan publik

Gunakan prosedur berikut untuk membuat VPC dengan subnet publik maupun privat.

Untuk membuat VPC dan subnet

1. Buka konsol VPC Amazon di. https://console.aws.amazon.com/vpc
2. Di pojok kanan atas Konsol Manajemen AWS, pilih Wilayah untuk membuat VPC Anda. Contoh ini menggunakan Wilayah AS Barat (Oregon).
3. Di sudut kiri atas, pilih Dasbor VPC . Untuk mulai membuat VPC, pilih Buat VPC.
4. Agar Sumber daya dapat dibuat di bagian pengaturan VPC, pilih VPC dan lainnya.
5. Untuk pengaturan VPC, atur nilai-nilai ini:
   - Generasi otomatis tag nama - **`example-dual-stack`**
   - Blok IPv4 CIDR - **`10.0.0.0/16`**
   - Blok IPv6 CIDR - Amazon-provided blok IPv6 CIDR
   - Penyewaan - Default
   - Jumlah Availability Zone (AZ) — 2
   - Kustomisasi AZ - Pertahankan nilai default
   - Jumlah subnet publik — 2
   - Jumlah subnet pribadi — 2
   - Kustomisasi subnet blok CIDR - Pertahankan nilai default
   - Gateway NAT ($) - Tidak ada
   - Pintu keluar hanya gateway internet - Tidak
   - Titik akhir VPC - Tidak ada
   - Opsi DNS - Pertahankan nilai default
6. Pilih Buat VPC.

Langkah 2: Buat grup keamanan VPC untuk instans Amazon EC2 publik

Berikutnya, Anda membuat grup keamanan untuk akses publik. Untuk terhubung ke instans EC2 publik di VPC Anda, tambahkan aturan masuk ke grup keamanan VPC Anda yang mengizinkan lalu lintas untuk terhubung dari internet.

Untuk membuat grup keamanan VPC

1. Buka konsol VPC Amazon di. https://console.aws.amazon.com/vpc
2. Pilih Dasbor VPC, pilih Grup Keamanan, lalu pilih Buat grup keamanan.
3. Di halaman Buat grup keamanan, atur nilai ini:
   - Nama grup keamanan - **`example-dual-stack-securitygroup`**
   - Deskripsi — **`Dual-stack security group`**
   - VPC — Pilih VPC yang Anda buat sebelumnya, misalnya: vpc-example-dual-stack.
4. Tambahkan aturan masuk ke grup keamanan.
   a. Tentukan alamat IP yang akan digunakan untuk terhubung ke instans EC2 di VPC Anda menggunakan Secure Shell (SSH). Untuk menentukan alamat IP publik Anda, di jendela atau tab browser yang berbeda, Anda dapat menggunakan layanan di https://checkip.amazonaws.com.

      Contoh rentang alamat Internet Protocol versi 4 (IPv4) adalah. `203.0.113.25/32` Contoh rentang alamat Protokol Internet versi 6 (IPv6) adalah `2001:db8:1234:1a00::/64`.

      Dalam banyak kasus, Anda dapat menghubungkan melalui penyedia layanan Internet (ISP) atau dari belakang firewall Anda tanpa alamat IP statis. Jika demikian, temukan rentang alamat IP yang digunakan oleh komputer klien.

      > **Warning**
      >
      > Jika Anda menggunakan `0.0.0.0/0` untuk IPv4 atau `::0` untuk IPv6, Anda memungkinkan semua alamat IP untuk mengakses instans publik Anda menggunakan SSH. Hal ini dapat diterima untuk waktu yang singkat di lingkungan pengujian, tetapi tidak aman untuk lingkungan produksi. Dalam produksi, Anda hanya dapat memberikan otorisasi pada alamat IP atau rentang alamat tertentu saja untuk mengakses instans-instans Anda.

b. Di bagian Aturan masuk, pilih Tambahkan aturan.

c. Atur nilai berikut untuk aturan masuk baru Anda yang akan mengizinkan akses SSH ke instans Amazon EC2 Anda. Setelah Anda melakukan ini, Anda dapat terhubung ke instans EC2 Anda untuk menginstal aplikasi atau utilitas lainnya. Tentukan alamat IP agar Anda dapat mengakses instans EC2 Anda:

- Jenis - **`SSH`**
- Sumber — Alamat IP atau rentang yang Anda buat dari Langkah a. Contoh rentang alamat IPv4 adalah **`203.0.113.25/3`** 2. Contoh rentang alamat IPv6 adalah. **`2001:DB8::/32`**

d. Pilih Tambahkan aturan.

5. Untuk membuat grup keamanan, pilih Buat grup keamanan.

   Perhatikan ID grup keamanan karena Anda membutuhkannya nanti di prosedur lain.

Langkah 3: Buat grup keamanan VPC untuk cluster pribadi

Untuk menjaga klaster Anda tetap pribadi, buat grup keamanan kedua untuk akses pribadi. Untuk terhubung ke kluster pribadi di VPC Anda, tambahkan aturan masuk ke grup keamanan VPC Anda. Hal ini mengizinkan lalu lintas dari instans Amazon EC2 Anda saja.

Untuk membuat grup keamanan VPC

1. Buka konsol VPC Amazon di. https://console.aws.amazon.com/vpc
2. Pilih Dasbor VPC, pilih Grup Keamanan, lalu pilih Buat grup keamanan.
3. Di halaman Buat grup keamanan, atur nilai ini:

   - Nama grup keamanan - **`example-dual-stack-cluster-securitygroup`**
   - Deskripsi — **`Dual-stack cluster security group`**
   - VPC — Pilih VPC yang Anda buat sebelumnya, misalnya: vpc-example-dual-stack

4. Tambahkan aturan masuk ke grup keamanan.

   a. Di bagian Aturan masuk, pilih Tambahkan aturan.

   b. Tetapkan nilai berikut untuk aturan masuk baru Anda untuk mengizinkan lalu lintas DocumentDB di port 27017 dari instans Amazon EC2 Anda. Setelah Anda melakukan ini,

Anda dapat terhubung dari instans EC2 Anda ke cluster Anda. Dengan demikian, Anda dapat mengirim data dari instans EC2 Anda ke database Anda.

- Jenis - **`Custom TCP`**
- Sumber — Pengidentifikasi grup keamanan EC2 yang Anda buat sebelumnya dalam topik ini, misalnya: sg-9edd5cfb.

5. Untuk membuat grup keamanan, pilih Buat grup keamanan.

Langkah 4: Buat grup subnet

Grup subnet adalah kumpulan subnet yang Anda buat di VPC dan kemudian Anda tentukan untuk cluster Anda. Dengan menggunakan grup subnet, Anda dapat menentukan VPC tertentu saat membuat cluster. Untuk membuat grup subnet yang `DUAL` kompatibel, semua subnet harus `DUAL` kompatibel. Agar kompatibel dengan `DUAL`, subnet harus memiliki CIDR IPv6 yang dikaitkan dengan subnet tersebut.

Untuk membuat grup subnet

1. Identifikasi subnet privat untuk basis data Anda di VPC.
   a. Buka konsol VPC Amazon di. https://console.aws.amazon.com/vpc
   b. Pilih Dasbor VPC, lalu pilih Subnet.
   c. Perhatikan ID subnet dari subnet yang Anda buat di Langkah 1 bernama, misalnya: example-dual-stack-subnet-private1-us-west-2a dan example-dual-stack-subnet-private2-us-west-2b. Anda memerlukan ID subnet saat membuat grup subnet Anda.
2. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb

   Pastikan Anda terhubung ke konsol Amazon DocumentDB, bukan ke konsol Amazon VPC.
3. Di panel navigasi, pilih Grup Subnet.
4. Pilih Buat.
5. Pada halaman Create subnet group, tetapkan nilai-nilai ini di bagian Detail grup Subnet:
   - Nama - **`example-dual-stack-cluster-subnet-group`**
   - Deskripsi — **`Dual-stack cluster subnet group`**
6. Di bagian Tambahkan subnet, atur nilai-nilai ini:

- VPC — Pilih VPC yang Anda buat sebelumnya, misalnya: vpc-example-dual-stack
- Availability Zones — Pilih kedua Availability Zone yang dibuat pada Langkah 1. Contoh: us-west-2a dan us-west-2b
- Subnet — Pilih subnet pribadi yang Anda buat di Langkah 1.

7. Pilih Buat.

Grup subnet baru Anda muncul di daftar grup subnet di konsol DocumentDB. Anda dapat memilih grup subnet untuk melihat detail di panel detail. Detail ini mencakup semua subnet yang terkait dengan grup.

Langkah 5: Buat instans Amazon EC2 dalam mode dual-stack

Untuk membuat instans Amazon EC2, ikuti petunjuk dalam Meluncurkan instans EC2 menggunakan wizard instans peluncuran di konsol di Panduan Pengguna Amazon Elastic Compute Cloud.

Di halaman Mengonfigurasi Detail Instans, atur nilai-nilai ini dan biarkan nilai lainnya sebagai default:

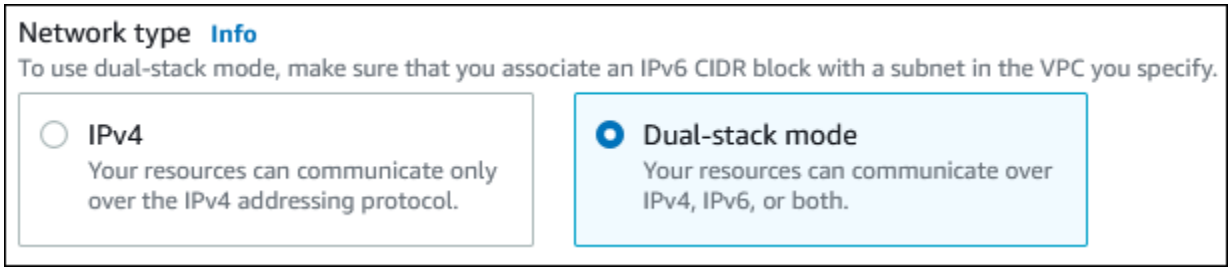
- Jaringan — Pilih VPC yang ada dengan subnet publik dan pribadi, seperti vpc-example-dual-stack-vpc (vpc- identifier) yang dibuat di. Langkah 1: Buat VPC dengan subnet pribadi dan publik
- Subnet — Pilih subnet publik yang ada, seperti subnet - identifier | example-dual-stack-subnet-public1-us-east-2a | us-east-2a dibuat di. Langkah 2: Buat grup keamanan VPC untuk instans Amazon EC2 publik
- Auto-assign IP Publik - Pilih Aktifkan.
- Auto-assign IPv6 IP - Pilih Aktifkan.
- Firewall (grup keamanan) Pilih grup keamanan yang ada. —
- Grup keamanan umum — Pilih grup keamanan yang ada, seperti example-dual-stack-securitygroup yang dibuat. Langkah 2: Buat grup keamanan VPC untuk instans Amazon EC2 publik Pastikan grup keamanan yang Anda pilih menyertakan aturan masuk untuk akses Secure Shell (SSH) dan HTTP.

Langkah 6: Buat cluster dalam mode dual-stack

Pada langkah ini, Anda membuat cluster DB yang berjalan dalam mode dual-stack. !!! Catatan: bagian ini perlu diedit setelah pembaruan IPv6 dibuat ke konsol!!!

Untuk membuat cluster dalam mode dual-stack

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di sudut kanan atas konsol, pilih Wilayah AWS tempat Anda ingin membuat cluster DocumentDB. Contoh ini menggunakan Wilayah AS Timur (Ohio).
3. Pada panel navigasi, silakan pilih Klaster.
4. Pada halaman daftar Clusters, pilih Create.
5. Pada halaman cluster Create Amazon DocumentDB, pastikan bahwa opsi cluster Instance-based dipilih.
6. Di bagian Konektivitas, di bawah Jenis jaringan, pilih Dual-stack mode.



7. Di bagian bawah halaman, aktifkan Tampilkan pengaturan lanjutan.
8. Di bagian Pengaturan jaringan, atur nilai-nilai ini:

   - Virtual private cloud (VPC) - Pilih VPC yang ada dengan subnet publik dan pribadi, seperti vpc-example-dual-stack (vpc - identifier) yang dibuat di. Langkah 1: Buat VPC dengan subnet pribadi dan publik

     VPC tersebut harus memiliki subnet di Zona Ketersediaan yang berbeda.
   - Grup subnet — Pilih grup subnet untuk VPC, seperti example-dual-stack-cluster-subnet-group yang dibuat di. Langkah 4: Buat grup subnet
   - Akses publik — Pilih No.
   - Grup keamanan VPC (firewall) - Pilih Pilih yang ada.
   - Grup keamanan VPC yang ada — Pilih grup keamanan VPC yang sudah ada yang dikonfigurasi untuk akses pribadi, seperti example-dual-stack-cluster-securitygroup yang dibuat di. Langkah 3: Buat grup keamanan VPC untuk cluster pribadi

     Hapus grup keamanan lainnya, seperti grup keamanan default, dengan memilih X yang dikaitkan dengan masing-masing grup keamanan.
   - Availability Zone — Pilih Availability Zone yang Anda buat di Langkah 1. Contoh: us-west-2a.

Untuk menghindari lalu lintas lintas AZ, pastikan cluster dan instans EC2 berada di Availability Zone yang sama.

9. Untuk bagian yang tersisa, tentukan pengaturan cluster Anda. Untuk informasi tentang setiap pengaturan, lihat Membuat cluster Amazon DocumentDB.

Langkah 7: Hubungkan ke instans Amazon EC2 dan kluster DB

Setelah membuat instans Amazon EC2 dan cluster DocumentDB dalam mode dual-stack, Anda dapat terhubung ke masing-masing instans menggunakan protokol IPv6. Untuk menyambung ke instans EC2 menggunakan protokol IPv6, ikuti petunjuk di Connect instans EC2 Anda di Panduan Pengguna Amazon Elastic Compute Cloud.

Untuk menyambung ke cluster DocumentDB Anda dari instans EC2, ikuti petunjuk Langkah 5: Instal MongoDB Shell dalam topik Connect Amazon EC2 secara manual (dan lanjutkan dengan Langkah 6 dan Langkah 7 berikutnya dalam prosedur yang sama).

Hapus VPC

Anda dapat menghapus VPC dan sumber daya lain yang digunakan di dalamnya, jika tidak lagi diperlukan.

> **Note**
>
> Jika Anda menambahkan sumber daya di VPC yang Anda buat dalam topik ini, Anda mungkin perlu menghapusnya sebelum dapat menghapus VPC. Misalnya, sumber daya ini mungkin menyertakan instans Amazon EC2 atau cluster DocumentDB. Untuk informasi selengkapnya, lihat Menghapus VPC di Panduan Pengguna Amazon VPC.

Untuk menghapus VPC dan sumber daya terkait

1. Hapus grup subnet:
   a. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
   b. Di panel navigasi, pilih Grup Subnet.
   c. Pilih grup subnet yang ingin Anda hapus, seperti example-dual-stack-cluster-subnet-group.

d. Pilih Hapus, lalu pilih Hapus di jendela konfirmasi.

2. Catat ID VPC:

   a. Buka konsol VPC Amazon di. https://console.aws.amazon.com/vpc

   b. Pilih Dasbor VPC, lalu pilih VPC Anda.

   c. Dalam daftar, identifikasi VPC yang Anda buat, seperti vpc-example-dual-stack.

   d. Catat ID VPC dari VPC yang Anda buat. ID VPC diperlukan di langkah berikutnya.

3. Hapus grup keamanan:

   a. Buka konsol VPC Amazon di. https://console.aws.amazon.com/vpc

   b. Pilih Dasbor VPC, lalu pilih Grup Keamanan.

   c. Pilih grup keamanan untuk klaster Amazon DocumentDB, seperti example-dual-stack-securitygroup.

   d. Untuk Tindakan, pilih Hapus grup keamanan, lalu pilih Hapus pada dialog konfirmasi.

   e. Kembali ke halaman Grup Keamanan, pilih grup keamanan untuk instans Amazon EC2, seperti example-securitygroup.

   f. Untuk Tindakan, pilih Hapus grup keamanan, lalu pilih Hapus pada dialog konfirmasi.

4. Hapus gateway NAT:

   a. Buka konsol VPC Amazon di. https://console.aws.amazon.com/vpc

   b. Pilih Dasbor VPC, lalu pilih Grup Keamanan.

   c. Pilih gateway NAT dari VPC yang Anda buat. Gunakan ID VPC untuk mengidentifikasi gateway NAT yang benar.

   d. Untuk Tindakan, pilih Hapus gateway NAT.

   e. Pada dialog konfirmasi, masukkan**`delete`**, lalu pilih Hapus.

5. Hapus VPC:

   a. Buka konsol VPC Amazon di. https://console.aws.amazon.com/vpc

   b. Pilih Dasbor VPC, lalu pilih VPC Anda.

   c. Pilih VPC yang ingin Anda hapus, seperti vpc-example-dual-stack.

   d. Untuk Tindakan, pilih Hapus VPC.

   Halaman konfirmasi menunjukkan sumber daya lain yang terkait dengan VPC yang juga akan dihapus, termasuk subnet yang terkait dengannya.

e. Pada dialog konfirmasi, masukkan**`delete`**, lalu pilih Hapus.

6. Lepaskan alamat IP elastis:

   a. Buka konsol EC2 di https://console.aws.amazon.com/ec2.

   b. Pilih Dasbor EC2, lalu pilih IP Elastis.

   c. Pilih alamat IP Elastis yang ingin Anda rilis.

   d. Untuk Tindakan, pilih Rilis alamat IP Elastis.

   e. Pada dialog konfirmasi, pilih Rilis.

Mencadangkan dan memulihkan di Amazon DocumentDB

Amazon DocumentDB (dengan kompatibilitas MongoDB) terus mencadangkan data Anda ke Amazon Simple Storage Service (Amazon S3) selama 1–35 hari sehingga Anda dapat dengan cepat memulihkan ke titik mana pun dalam periode retensi pencadangan. Amazon DocumentDB juga mengambil snapshot otomatis data Anda sebagai bagian dari proses backup terus menerus ini.

> **Note**
>
> Ini adalah bucket Amazon S3 yang dikelola layanan dan Anda tidak akan memiliki akses ke file backup. Jika Anda ingin mengontrol backup Anda sendiri, ikuti petunjuk pada Membuang, Memulihkan, Mengimpor, dan Mengekspor Data.

Anda juga dapat mempertahankan data backup di luar periode retensi pencadangan dengan membuat snapshot manual data klaster Anda. Proses backup tidak memengaruhi performa klaster Anda.

Bagian ini membahas kasus penggunaan untuk kemampuan backup di Amazon DocumentDB dan menunjukkan cara untuk mengelola backup untuk klaster Amazon DocumentDB Anda.

Topik

- Cadangkan dan pulihkan: konsep
- Memahami penggunaan penyimpanan cadangan
- Membuang, memulihkan, mengimpor, dan mengekspor data
- Pertimbangan snapshot cluster
- Membandingkan snapshot otomatis dan manual
- Membuat snapshot cluster manual
- Menyalin snapshot cluster Amazon DocumentDB
- Berbagi snapshot cluster Amazon DocumentDB
- Memulihkan dari snapshot cluster
- Memulihkan ke titik waktu
- Menghapus snapshot cluster

Cadangkan dan pulihkan: konsep

Kata benda	Deskripsi	API (Kata kerja)

Periode retensi cadangan	Periode waktu antara 1 dan 35 hari di mana Anda dapat melakukan pemulihan titik waktu.	`create-db-cluster`
		`modify-db-cluster`
		`restore-db-cluster-to-point-in-time`

Volume penyimpanan Amazon DocumentDB	Volume penyimpanan yang sangat tersedia dan sangat tahan lama yang mereplikasi data enam cara di tiga Availability Zone. Sebuah klaster Amazon DocumentDB sangat tahan lama terlepas dari jumlah instans	`create-db-cluster` `delete-db-cluster`

Kata benda	Deskripsi	API (Kata kerja)
	dalam klaster.	

Periode pencadangan	Periode waktu dalam hari di mana snapshot otomatis diambil.	`create-db-cluster`
		`describe-db-cluster`
		`modify-db-cluster`

Snapshot otomatis	Snapshot harian yang merupakan backup penuh klaster dan dibuat secara otomatis oleh proses backup berkelanjutan di Amazon DocumentDB.	`restore-db-cluster-from-snapshot`
		`describe-db-cluster-snapshot-attributes`
		`describe-db-cluster-snapshots`

Kata benda	Deskripsi	API (Kata kerja)

Snapshot manual	Snapshot yang Anda buat secara manual untuk mempertahankan backup penuh klaster di luar periode 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`

Memahami penggunaan penyimpanan cadangan

Penyimpanan backup Amazon DocumentDB terdiri dari backup berkelanjutan dalam periode retensi pencadangan dan snapshot manual di luar periode retensi. Untuk mengontrol penggunaan penyimpanan backup, Anda dapat mengurangi interval retensi backup, menghapus snapshot manual lama saat tidak lagi diperlukan, atau keduanya. Untuk informasi umum tentang backup Amazon DocumentDB, lihat Mencadangkan dan memulihkan di Amazon DocumentDB. Untuk informasi harga tentang penyimpanan backup Amazon DocumentDB, lihat Harga Amazon DocumentDB.

Untuk mengontrol biaya Anda, Anda dapat memantau jumlah penyimpanan yang digunakan oleh pencadangan berkelanjutan dan snapshot manual yang terus berlanjut melampaui periode penyimpanan. Kemudian Anda dapat mengurangi interval retensi cadangan dan menghapus snapshot manual ketika tidak diperlukan lagi.

Anda dapat menggunakan CloudWatch metrik `AmazonTotalBackupStorageBilled`,`SnapshotStorageUsed`, dan `BackupRetentionPeriodStorageUsed` untuk meninjau dan memantau jumlah penyimpanan yang digunakan oleh backup Amazon DocumentDB Anda, sebagai berikut:

- `BackupRetentionPeriodStorageUsed` menunjukkan jumlah penyimpanan backup yang digunakan, untuk menyimpan backup berkelanjutan pada saat ini. Nilai metrik ini bergantung pada ukuran volume klaster dan jumlah perubahan yang Anda buat selama periode retensi. Namun, untuk tujuan penagihan, metrik tidak melebihi ukuran volume klaster kumulatif selama periode retensi. Misalnya, jika ukuran klaster Anda adalah 100 GiB dan periode retensi Anda adalah dua hari, nilai maksimum untuk `BackRetentionPeriodStorageUsed` adalah 200 GiB (100 GiB + 100 GiB).

- `SnapshotStorageUsed` mewakili jumlah penyimpanan backup yang digunakan, untuk menyimpan snapshot manual di luar periode retensi pencadangan. Snapshot manual yang diambil dalam periode retensi tidak dihitung terhadap penyimpanan backup Anda. Demikian pula, snapshot otomatis tidak dihitung terhadap penyimpanan backup Anda. Ukuran masing-masing snapshot adalah ukuran volume klaster pada saat Anda mengambil snapshot. Nilai `SnapshotStorageUsed` bergantung pada jumlah snapshot yang disimpan dan ukuran setiap snapshot. Misalnya, Anda memiliki satu snapshot di luar periode retensi dan ukuran volume klaster adalah 100 GiB saat snapshot itu diambil. Jumlah `SnapshotStorageUsed` adalah 100 GiB.

- `TotalBackupStorageBilled` mewakili jumlah dari `BackupRetentionPeriodStorageUsed` dan `SnapshotStorageUsed`, dikurangi jumlah penyimpanan backup gratis yang setara dengan ukuran volume klaster selama satu hari. Misalnya, jika ukuran klaster Anda adalah 100 GiB, Anda memiliki satu hari retensi, dan Anda memiliki satu snapshot di luar periode retensi, `TotalBackupStorageBilled`-nya adalah 100 GiB (100 GiB + 100 GiB - 100 GiB).

- Metrik ini dihitung secara independen untuk setiap klaster Amazon DocumentDB.

Anda dapat memantau klaster Amazon DocumentDB dan membuat laporan CloudWatch menggunakan metrik melalui konsol. CloudWatch Untuk informasi selengkapnya tentang cara menggunakan CloudWatch metrik, lihatMemantau Amazon DocumentDB.

Membuang, memulihkan, mengimpor, dan mengekspor data

Anda dapat menggunakan utilitas `mongodump`, `mongorestore`, `mongoexport`, dan `mongoimport` untuk memindahkan data masuk dan keluar dari klaster Amazon DocumentDB Anda. Bagian ini

membahas tujuan masing-masing alat dan konfigurasi ini untuk membantu Anda mencapai performa yang lebih baik.

Topik

- mongodump
- mongorestore
- mongoekspor
- mongoimpor
- Tutorial

mongodump

Utilitas mongodump membuat backup biner (BSON) dari basis data MongoDB. Alat mongodump adalah metode yang disukai untuk membuang data dari deployment MongoDB sumber Anda ketika ingin memulihkannya ke dalam klaster Amazon DocumentDB Anda karena efisiensi ukuran yang dicapai dengan menyimpan data dalam format biner.

Bergantung pada sumber daya yang tersedia pada instance atau mesin yang Anda gunakan untuk menjalankan perintah, Anda dapat mempercepat mongodump dengan meningkatkan jumlah koleksi paralel yang dibuang dari default 1 menggunakan opsi. `--numParallelCollections` Aturan praktis yang baik adalah memulai dengan satu pekerja per vCPU pada instans utama klaster Amazon DocumentDB Anda.

> **Note**
>
> Kami merekomendasikan MongoDB Database Tools hingga dan termasuk versi 100.6.1 untuk Amazon DocumentDB. Anda dapat mengakses unduhan MongoDB Database Tools di sini.

Contoh penggunaan

Berikut ini adalah contoh penggunaan utilitas mongodump dalam klaster 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

Utilitas `mongorestore` memungkinkan Anda untuk memulihkan backup biner (BSON) dari basis data yang telah dibuat dengan utilitas `mongodump`. Anda dapat meningkatkan performa pemulihan dengan meningkatkan jumlah pekerja untuk setiap koleksi selama pemulihan dengan opsi `--numInsertionWorkersPerCollection` (default-nya adalah 1). Aturan praktis yang baik adalah memulai dengan satu pekerja per vCPU pada instans utama klaster Amazon DocumentDB Anda.

Contoh penggunaan

Berikut ini adalah contoh penggunaan utilitas `mongorestore` dalam klaster 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>
```

mongoekspor

Alat `mongoexport` mengekspor data di Amazon DocumentDB ke format file JSON, CSV, atau TSV. Alat `mongoexport` adalah metode yang disukai untuk mengekspor data yang harus dapat dibaca oleh manusia atau mesin.

> **Note**
>
> `mongoexport` tidak secara langsung mendukung ekspor paralel. Namun, dimungkinkan untuk meningkatkan performa dengan mengeksekusi beberapa tugas `mongoexport` secara bersamaan untuk koleksi yang berbeda.

Contoh penggunaan

Berikut ini adalah contoh penggunaan alat `mongoexport` dalam klaster 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
```

mongoimpor

Alat `mongoimport` mengimpor isi file JSON, CSV, atau TSV ke dalam klaster Amazon DocumentDB. Anda dapat menggunakan parameter `--numInsertionWorkers` untuk memparalelkan dan mempercepat impor (default-nya adalah 1).

Contoh penggunaan

Berikut ini adalah contoh penggunaan alat `mongoimport` dalam klaster 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

Tutorial berikut ini menjelaskan cara menggunakan utilitas mongodump, `mongorestore`, `mongoexport`, dan `mongoimport` untuk memindahkan data masuk dan keluar dari klaster Amazon DocumentDB.

1. Prasyarat — Sebelum Anda mulai, pastikan bahwa klaster Amazon DocumentDB Anda telah disediakan dan bahwa Anda memiliki akses ke Instans Amazon EC2 di VPC yang sama dengan klaster Anda. Untuk informasi selengkapnya, lihat Connect menggunakan Amazon EC2.

   Untuk dapat menggunakan alat utilitas mongo, Anda harus menginstal paket mongodb-org-tools di instans EC2 Anda, sebagai berikut.

   ```
   sudo yum install mongodb-org-tools-4.0.18
   ```

   Karena Amazon DocumentDB menggunakan enkripsi Keamanan Lapisan Pengangkutan (TLS) secara default, Anda juga harus mengunduh file otoritas sertifikat (CA) Amazon RDS untuk menggunakan shell mongo guna menghubungkan, sebagai berikut.

   ```
   wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
   ```

2. Unduh data sampel — Untuk tutorial ini, Anda akan mengunduh beberapa data sampel yang berisi informasi tentang restoran.

   ```
   wget https://raw.githubusercontent.com/ozlerhakan/mongodb-json-files/master/
   datasets/restaurant.json
   ```

3. Impor data sampel ke Amazon DocumentDB — Karena data dalam format JSON logis, Anda akan menggunakan utilitas `mongoimport` untuk mengimpor data ke dalam klaster Amazon DocumentDB Anda.

   ```
   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. Dump data dengan **`mongodump`** — Sekarang setelah Anda memiliki data di klaster Amazon DocumentDB, Anda dapat melakukan dump biner dari data tersebut menggunakan utilitas `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. Jatuhkan koleksi **`restaurants`** — Sebelum Anda memulihkan koleksi `restaurants` di basis data `business`, Anda harus terlebih dahulu menjatuhkan koleksi yang sudah ada dalam basis data tersebut, sebagai berikut.

```
use business
```

```
db.restaurants.drop()
```

6. Pulihkan data dengan **`mongorestore`** — Dengan dump biner data dari Langkah 3, Anda sekarang dapat menggunakan utilitas `mongorestore` untuk memulihkan data Anda ke klaster Amazon DocumentDB Anda.

```
mongorestore --ssl \
    --host="tutorialCluster.us-east-1.docdb.amazonaws.com:27017" \
    --numParallelCollections 4 \
    --username=<yourUsername> \
    --password=<yourPassword> \
    --sslCAFile global-bundle.pem restaurantDump.bson
```

7. Ekspor data menggunakan **`mongoexport`** — Untuk menyelesaikan tutorial, ekspor data dari klaster Anda dalam format file JSON, tidak berbeda dari file yang Anda impor di Langkah 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. Validasi — Anda dapat memvalidasi bahwa output dari Langkah 5 menghasilkan hasil yang sama seperti Langkah 1 dengan perintah berikut.

```
wc -l restaurant.json
```

Output dari perintah ini:

```
2548 restaurant.json
```

```
wc -l restaurant2.json
```

Output dari perintah ini:

```
2548 restaurant2.json
```

Pertimbangan snapshot cluster

Amazon DocumentDB membuat snapshot otomatis harian dari klaster Anda selama jendela backup klaster Anda. Amazon DocumentDB menyimpan snapshot otomatis dari klaster Anda sesuai periode retensi pencadangan yang Anda tentukan. Jika perlu, Anda dapat memulihkan klaster Anda ke setiap titik waktu selama periode retensi pencadangan. Snapshot otomatis tidak terjadi saat operasi penyalinan dijalankan di Wilayah yang sama untuk klaster yang sama.

Topik

- Penyimpanan cadangan
- Periode pencadangan
- Periode retensi cadangan
- Salin enkripsi snapshot cluster

Selain untuk snapshot klaster otomatis, Anda juga dapat membuat snapshot klaster secara manual. Anda dapat menyalin snapshot otomatis dan manual. Untuk informasi selengkapnya, lihat Membuat snapshot cluster manual dan Menyalin snapshot cluster Amazon DocumentDB.

> **Note**
>
> Klaster Anda harus dalam status tersedia agar snapshot otomatis diambil.
> Anda tidak dapat membagikan snapshot klaster otomatis Amazon DocumentDB. Sebagai solusinya, Anda dapat membuat snapshot manual dengan menyalin snapshot otomatis, lalu membagikan salinan tersebut. Untuk informasi selengkapnya tentang menyalin snapshot, lihat Menyalin snapshot cluster Amazon DocumentDB. Untuk informasi selengkapnya tentang memulihkan klaster dari snapshot, lihat Memulihkan dari snapshot cluster.

Penyimpanan cadangan

Penyimpanan cadangan Amazon DocumentDB Anda untuk Wilayah AWS masing-masing terdiri dari penyimpanan cadangan yang diperlukan untuk periode retensi cadangan Anda, yang mencakup snapshot cluster otomatis dan manual di Wilayah tersebut. Periode retensi cadangan default adalah 1 hari. Untuk informasi selengkapnya tentang harga penyimpanan backup, lihat Harga Amazon DocumentDB.

Ketika Anda menghapus sebuah klaster, semua snapshot otomatisnya dihapus dan tidak dapat dipulihkan. Namun, snapshot manual tidak dihapus ketika Anda menghapus klaster. Jika Anda memilih agar Amazon DocumentDB membuat snapshot akhir (snapshot manual) sebelum klaster Anda dihapus, Anda dapat menggunakan snapshot akhir untuk memulihkan klaster Anda.

Untuk informasi selengkapnya tentang snapshot dan penyimpanan, lihat Memahami penggunaan penyimpanan cadangan.

Periode pencadangan

Snapshot otomatis terjadi setiap hari selama jendela backup yang dipilih. Jika snapshot memerlukan waktu lebih dari yang dialokasikan ke jendela backup, proses backup berlanjut hingga selesai, meskipun jendela backup telah berakhir. Jendela backup tidak dapat menindih jendela pemeliharaan mingguan untuk klaster.

Jika Anda tidak menentukan jendela backup yang diinginkan saat Anda membuat klaster, Amazon DocumentDB menetapkan jendela backup 30 menit default. Jendela ini dipilih secara acak dari blok waktu 8 jam yang terkait dengan Wilayah klaster Anda. Anda dapat mengubah jendela backup pilihan Anda dengan memodifikasi klaster. Untuk informasi selengkapnya, lihat Memodifikasi cluster Amazon DocumentDB.

Nama Wilayah	Region	Blok Waktu UTC
AS Timur (Ohio)	us-east-2	03:00-11:00
US East (Northern Virginia)	us-east-1	03:00-11:00
AS Barat (Oregon)	us-west-2	06:00-14:00
Africa (Cape Town)	af-south-1	03:00 — 11:00
Asia Pasifik (Hong Kong)	ap-east-1	06:00-14:00
Asia Pasifik (Hyderabad)	ap-south-2	06:30 — 14:30
Asia Pasifik (Malaysia)	ap-southeast-5	13:00-21:00
Asia Pasifik (Mumbai)	ap-south-1	06:00-14:00
Asia Pacific (Osaka)	ap-northeast-3	12:00-20:00
Asia Pasifik (Seoul)	ap-northeast-2	13:00-21:00
Asia Pasifik (Singapura)	ap-southeast-1	14:00-22:00
Asia Pasifik (Sydney)	ap-southeast-2	12:00-20:00
Asia Pasifik (Jakarta)	ap-southeast-3	08:00-16:00
Asia Pacific (Melbourne)	ap-southeast-4	11:00-19:00
Asia Pasifik (Thailand)	ap-tenggara 7	15:00-23:00
Asia Pasifik (Tokyo)	ap-northeast-1	13:00-21:00
Kanada (Pusat)	ca-central-1	03:00-11:00
Kanada Barat (Calgary)	ca-west-1	18:00-02:00
Tiongkok (Beijing)	cn-north-1	06:00-14:00
Tiongkok (Ningxia)	cn-northwest-1	06:00-14:00

Nama Wilayah	Region	Blok Waktu UTC
Eropa (Frankfurt)	eu-central-1	21:00-05:00
Europe (Zurich)	eu-central-2	02:00-10:00
Eropa (Irlandia)	eu-west-1	22:00-06:00
Eropa (London)	eu-west-2	22:00-06:00
Europe (Milan)	eu-south-1	02:00-10:00
Eropa (Paris)	eu-west-3	23:59-07:29
Eropa (Spanyol)	eu-south-2	02:00 — 10:00
Eropa (Stockholm)	eu-north-1	04:00 — 12:00
Meksiko (Tengah)	mx-pusat-1	03:00-11:00
Timur Tengah (UAE)	me-central-1	05:00 — 13:00
Amerika Selatan (Sao 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

Periode retensi cadangan

Periode retensi cadangan adalah jumlah hari backup otomatis dipertahankan sebelum dihapus secara otomatis. Amazon DocumentDB mendukung periode retensi cadangan selama 1-35 hari.

Anda dapat mengatur periode retensi cadangan saat Anda membuat klaster. Jika Anda tidak secara eksplisit mengatur periode retensi cadangan, periode retensi cadangan default selama 1 hari ditetapkan ke klaster Anda. Setelah Anda membuat klaster, Anda dapat memodifikasi periode retensi cadangan dengan memodifikasi cluster menggunakan salah satu Konsol Manajemen AWS atau. AWS CLI Untuk informasi selengkapnya, lihat Memodifikasi cluster Amazon DocumentDB.

Salin enkripsi snapshot cluster

Enkripsi cluster dan snapshot didasarkan pada kunci enkripsi KMS. ID kunci KMS adalah Amazon Resource Name (ARN), pengidentifikasi kunci KMS, atau alias kunci KMS untuk kunci enkripsi KMS.

Pedoman dan batasan berikut berlaku:

- Enkripsi disimpulkan dari cluster saat membuat snapshot. Jika cluster dienkripsi, snapshot cluster itu dienkripsi dengan kunci KMS yang sama. Jika cluster tidak dienkripsi, snapshot tidak dienkripsi.
- Jika Anda menyalin snapshot kluster terenkripsi dari akun Amazon Web Services, Anda dapat menentukan nilai `KmsKeyId` untuk mengenkripsi salinan dengan kunci enkripsi KMS baru. Jika Anda tidak menentukan nilai untuk`KmsKeyId`, maka salinan snapshot cluster dienkripsi dengan kunci KMS yang sama dengan snapshot cluster sumber.
- Jika Anda menyalin snapshot kluster terenkripsi yang dibagikan dari akun Amazon Web Services lainnya, Anda harus menentukan nilainya. `KmsKeyId`
- Untuk menyalin snapshot kluster terenkripsi ke Wilayah Amazon Web Services lainnya, atur `KmsKeyId` ke ID kunci KMS yang ingin Anda gunakan untuk mengenkripsi salinan snapshot klaster di Wilayah tujuan. Kunci enkripsi KMS khusus untuk Wilayah Amazon Web Services tempat mereka dibuat, dan Anda tidak dapat menggunakan kunci enkripsi dari satu Wilayah Amazon Web Services di Wilayah Amazon Web Services lainnya.
- Jika Anda menyalin snapshot klaster yang tidak terenkripsi dan menentukan nilai untuk parameter `KmsKeyId`, kesalahan akan dikembalikan.

Membandingkan snapshot otomatis dan manual

Berikut ini adalah fitur utama dari snapshot otomatis dan manual Amazon DocumentDB (dengan kompatibilitas MongoDB).

Snapshot otomatis Amazon DocumentDB memiliki fitur utama berikut:

- Penamaan snapshot otomatis — Nama snapshot otomatis mengikuti pola `rds:<cluster-name>-yyyy-mm-dd-hh-mm`, dengan `yyyy-mm-dd-hh-mm` mewakili tanggal dan waktu snapshot dibuat.
- Dibuat secara otomatis sesuai jadwal — Ketika Anda membuat atau memodifikasi klaster, Anda dapat mengatur Periode retensi cadangan ke nilai bilangan bulat dari 1 sampai 35 hari. Secara default, klaster baru memiliki periode retensi cadangan selama 1 hari. Periode retensi cadangan

menentukan jumlah hari snapshot otomatis disimpan sebelum dihapus secara otomatis. Anda tidak dapat menonaktifkan backup otomatis di klaster Amazon DocumentDB.

Selain mengatur periode retensi cadangan, Anda juga mengatur jendela backup, waktu hari dimana snapshot otomatis dibuat.

- Menghapus snapshot otomatis — Snapshot otomatis dihapus ketika Anda menghapus klaster snapshot otomatis. Anda tidak dapat menghapus snapshot otomatis secara manual.
- Tambahan — Selama periode retensi cadangan, pembaruan basis data dicatat sehingga ada catatan inkremental dari perubahan.
- Memulihkan dari snapshot otomatis — Anda dapat memulihkan dari snapshot otomatis menggunakan Konsol Manajemen AWS atau AWS CLI. Saat Anda memulihkan dari snapshot menggunakan AWS CLI, Anda harus menambahkan instance secara terpisah setelah cluster tersedia.
- Membagikan — Anda tidak dapat membagikan snapshot klaster otomatis Amazon DocumentDB. Sebagai solusinya, Anda dapat membuat snapshot manual dengan menyalin snapshot otomatis, lalu membagikan salinan tersebut. Untuk informasi selengkapnya tentang menyalin snapshot, lihat Menyalin snapshot cluster Amazon DocumentDB. Untuk informasi selengkapnya tentang memulihkan klaster dari snapshot, lihat Memulihkan dari snapshot cluster.
- Anda dapat memulihkan dari titik mana pun dalam periode retensi cadangan — Karena pembaruan basis data dicatat secara bertahap, Anda dapat memulihkan klaster Anda ke setiap titik waktu dalam periode retensi cadangan.

  Ketika Anda memulihkan dari snapshot otomatis atau dari pemulihan point-in-time menggunakan AWS CLI, Anda harus menambahkan instance secara terpisah setelah cluster tersedia.

Snapshot manual Amazon DocumentDB memiliki fitur utama berikut:

- Dibuat sesuai permintaan - Snapshot manual Amazon DocumentDB dibuat sesuai permintaan menggunakan Amazon DocumentDB Management Console atau. AWS CLI
- Menghapus snapshot manual — Sebuah snapshot manual dihapus hanya ketika Anda secara eksplisit menghapusnya menggunakan konsol Amazon DocumentDB atau AWS CLI. Snapshot manual tidak dihapus saat Anda menghapus klasternya.
- Backup lengkap — Ketika snapshot manual diambil, backup lengkap dari data klaster Anda dibuat dan disimpan.

- Penamaan snapshot manual — Anda menentukan nama snapshot manual. Amazon DocumentDB tidak menambahkan cap `datetime` ke nama, sehingga Anda harus menambahkan informasi itu jika Anda ingin itu disertakan dalam nama.
- Memulihkan dari snapshot manual — Anda dapat memulihkan dari snapshot manual menggunakan konsol atau AWS CLI. Saat Anda memulihkan dari snapshot menggunakan AWS CLI, Anda harus menambahkan instance secara terpisah setelah cluster tersedia.
- Service Quotas — Anda dibatasi maksimal 100 snapshot manual per. Wilayah AWS
- Membagikan — Anda dapat berbagi snapshot klaster manual, yang dapat disalin oleh Akun AWS yang diotorisasi. Anda dapat berbagi snapshot manual yang terenkripsi atau tidak terenkripsi. Untuk informasi selengkapnya tentang menyalin snapshot, lihat Menyalin snapshot cluster Amazon DocumentDB.
- Anda memulihkan ke saat snapshot manual diambil — Saat Anda memulihkan dari snapshot manual, Anda memulihkan ke saat snapshot manual diambil.

  Saat Anda memulihkan dari snapshot menggunakan AWS CLI, Anda harus menambahkan instance secara terpisah setelah cluster tersedia.

Membuat snapshot cluster manual

Anda dapat membuat snapshot manual menggunakan Konsol Manajemen AWS atau AWS CLI. Jumlah waktu yang dibutuhkan untuk membuat snapshot berbeda-beda sesuai dengan ukuran dari basis data Anda. Saat Anda membuat snapshot, Anda harus melakukan hal berikut:

1. Mengidentifikasi klaster mana yang akan dicadangkan.
2. Berikan nama snapshot Anda. Hal ini memungkinkan Anda untuk memulihkan dari itu nanti.

Using the Konsol Manajemen AWS

Untuk membuat snapshot manual menggunakan Konsol Manajemen AWS, Anda dapat mengikuti salah satu metode di bawah ini.

1. Metode 1:

   1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
   2. Di panel navigasi, pilih Snapshot.

> **Tip**
>
> Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu (≡) di pojok kiri atas halaman.

3. Pada halaman Snapshot, pilih Buat.
4. Pada halaman Membuat snapshot klaster:
   a. Pengidentifikasi klaster — Dari daftar tarik-turun klaster, pilih klaster yang ingin Anda buat snapshotnya.
   b. Pengidentifikasi snapshot — Masukkan nama untuk snapshot Anda.

      Batasan penamaan snapshot:

      - Panjangnya adalah [1–255] huruf, angka, atau tanda hubung.
      - Karakter pertama harus berupa huruf.
      - Tidak dapat diakhiri dengan tanda hubung atau berisi dua tanda hubung berurutan.
      - Harus unik untuk semua klaster (di Amazon RDS, Amazon Neptune, dan Amazon DocumentDB) per akun AWS , per Wilayah.

   c. Pilih Buat.

2. Metode 2:

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Klaster.

> **Tip**
>
> Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu (≡) di pojok kiri atas halaman.

3. Pada halaman Klaster, pilih tombol di sebelah kiri pada klaster yang ingin Anda snapshot.
4. Dari menu Tindakan, pilih Ambil snapshot.
5. Pada halaman Membuat snapshot klaster:

a. Pengidentifikasi snapshot — Masukkan nama untuk snapshot Anda.

   Batasan penamaan snapshot:

   - Panjangnya adalah [1–63] huruf, angka, atau tanda hubung.
   - Karakter pertama harus berupa huruf.
   - Tidak dapat diakhiri dengan tanda hubung atau berisi dua tanda hubung berurutan.
   - Harus unik untuk semua klaster (di Amazon RDS, Amazon Neptune, dan Amazon DocumentDB) per akun AWS , per Wilayah.

b. Pilih Buat.

Using the AWS CLI

Untuk membuat snapshot cluster menggunakan AWS CLI, gunakan `create-db-cluster-snapshot` operasi dengan parameter berikut.

Parameter

- **`--db-cluster-identifier`** — Diperlukan. Nama klaster yang ingin Anda ambil snapshotnya. Klaster ini harus ada dan tersedia.
- **`--db-cluster-snapshot-identifier`** — Diperlukan. Nama snapshot manual yang Anda buat.

Contoh berikut membuat snapshot bernama `sample-cluster-snapshot` untuk klaster bernama `sample-cluster`.

Untuk Linux, macOS, atau Unix:

```
aws docdb create-db-cluster-snapshot \
    --db-cluster-identifier sample-cluster \
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

Untuk Windows:

```
aws docdb create-db-cluster-snapshot ^
    --db-cluster-identifier sample-cluster ^
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

Output dari operasi ini terlihat seperti berikut.

```
{
    "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"
    }
}
```


Menyalin snapshot cluster Amazon DocumentDB

Di Amazon DocumentDB, Anda dapat menyalin snapshot dalam Wilayah AWS hal yang sama atau berbeda. Wilayah AWS Anda juga dapat menyalin snapshot bersama ke akun Anda dalam hal yang sama Wilayah AWS atau berbeda Wilayah AWS. Untuk informasi selengkapnya tentang berbagi snapshot, lihat Berbagi snapshot cluster Amazon DocumentDB.

> **Note**
>
> Amazon DocumentDB menagih Anda berdasarkan jumlah backup dan data snapshot yang Anda simpan dan periode waktu Anda menyimpannya. Untuk informasi selengkapnya tentang penyimpanan yang terkait dengan backup dan snapshot Amazon DocumentDB,

lihat Memahami penggunaan penyimpanan cadangan. Untuk informasi harga tentang penyimpanan Amazon DocumentDB, lihat Harga Amazon DocumentDB.

Topik

- Menyalin snapshot bersama
- Menyalin snapshot di seluruh Wilayah AWS
- Batasan
- Menangani enkripsi
- Pertimbangan grup parameter
- Menyalin snapshot cluster

Menyalin snapshot bersama

Anda dapat menyalin snapshot yang dibagikan kepada Anda oleh AWS akun lain. Jika Anda menyalin snapshot terenkripsi yang telah dibagikan dari AWS akun lain, Anda harus memiliki akses ke kunci AWS KMS enkripsi yang digunakan untuk mengenkripsi snapshot. Untuk informasi selengkapnya, lihat Menangani enkripsi.

Menyalin snapshot di seluruh Wilayah AWS

Saat Anda menyalin snapshot ke snapshot Wilayah AWS yang berbeda dari snapshot sumber Wilayah AWS, setiap salinan adalah snapshot lengkap. Salinan snapshot lengkap berisi semua data dan data meta yang diperlukan untuk memulihkan cluster Amazon DocumentDB.

Batasan

Berikut adalah beberapa batasan saat Anda menyalin snapshot:

- Jika Anda menghapus snapshot sumber sebelum snapshot target tersedia, salinan snapshot mungkin gagal. Verifikasi bahwa snapshot target memiliki status `AVAILABLE` sebelum Anda menghapus snapshot sumber.
- Anda dapat memiliki maksimum lima permintaan penyalinan snapshot yang sedang berlangsung ke satu Wilayah tujuan per akun.
- Bergantung pada yang Wilayah AWS terlibat dan jumlah data yang akan disalin, salinan snapshot lintas wilayah dapat memakan waktu berjam-jam untuk diselesaikan. Dalam beberapa kasus,

mungkin ada sejumlah besar permintaan salinan snapshot lintas wilayah dari Wilayah AWS sumber tertentu. Dalam kasus ini, Amazon DocumentDB mungkin menempatkan permintaan salinan lintas wilayah baru dari Wilayah AWS sumber tersebut ke dalam antrian hingga beberapa salinan yang sedang berlangsung selesai. Tidak ada informasi kemajuan yang ditampilkan tentang penyalinan permintaan saat berada di antrean. Informasi progres akan ditampilkan saat penyalinan dimulai.

Menangani enkripsi

Anda dapat menyalin snapshot yang telah dienkripsi menggunakan kunci enkripsi AWS KMS . Jika Anda menyalin snapshot terenkripsi, salinan snapshot tersebut juga harus dienkripsi. Jika Anda menyalin snapshot terenkripsi dalam yang sama Wilayah AWS, Anda dapat mengenkripsi salinan dengan kunci AWS KMS enkripsi yang sama dengan snapshot asli, atau Anda dapat menentukan kunci enkripsi yang berbeda. AWS KMS Jika Anda menyalin snapshot terenkripsi di seluruh Wilayah, Anda tidak dapat menggunakan kunci AWS KMS enkripsi yang sama untuk salinan seperti yang digunakan untuk snapshot sumber, karena kuncinya. AWS KMS Region-specific Sebagai gantinya, Anda harus menentukan AWS KMS kunci yang valid di tujuan Wilayah AWS n.

Snapshot sumber tetap terenkripsi selama proses penyalinan. Untuk informasi selengkapnya, lihat Perlindungan data di Amazon DocumentDB.

> **Note**
>
> Untuk snapshot klaster Amazon DocumentDB, Anda tidak dapat mengenkripsi snapshot klaster yang tidak dienkripsi saat Anda menyalin snapshot.

Pertimbangan grup parameter

Saat Anda menyalin snapshot lintas Wilayah, salinan tersebut tidak menyertakan grup parameter yang digunakan oleh klaster Amazon DocumentDB asli. Saat Anda memulihkan snapshot untuk membuat cluster baru, cluster itu mendapatkan grup parameter default untuk Wilayah AWS itu dibuat. Untuk memberikan klaster baru parameter yang sama seperti aslinya, Anda harus melakukan hal berikut:

1. Di tujuan Wilayah AWS, buat grup parameter cluster Amazon DocumentDB dengan pengaturan yang sama dengan cluster asli. Jika sudah ada di yang baru Wilayah AWS, Anda dapat menggunakan yang itu.

2. Setelah memulihkan snapshot di tujuan Wilayah AWS, ubah cluster Amazon DocumentDB baru dan tambahkan grup parameter baru atau yang sudah ada dari langkah sebelumnya. Untuk informasi selengkapnya, lihat Memodifikasi cluster Amazon DocumentDB.

Menyalin snapshot cluster

Anda dapat menyalin cluster Amazon DocumentDB menggunakan atau Konsol Manajemen AWS , sebagai berikut AWS CLI.

Using the Konsol Manajemen AWS

Untuk membuat salinan snapshot cluster menggunakan Konsol Manajemen AWS, selesaikan langkah-langkah berikut. Prosedur ini berfungsi untuk menyalin snapshot cluster terenkripsi atau tidak terenkripsi, di wilayah yang sama atau di seluruh Wilayah. Wilayah AWS

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Snapshot, lalu pilih tombol di sebelah kiri snapshot yang ingin Anda salin.

   > **Tip**
   >
   > Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu (≡) di pojok kiri atas halaman.

3. Dari menu Tindakan, pilih Salin.
4. Di halaman Buat Salinan snapshot klaster yang dihasilkan, selesaikan bagian Pengaturan.

   a. Wilayah Tujuan — Opsional. Untuk menyalin snapshot cluster ke yang lain Wilayah AWS, pilih itu Wilayah AWS untuk Wilayah Tujuan.

   b. Pengidentifikasi snapshot baru — Masukkan nama untuk snapshot baru.

      Batasan penamaan snapshot target:

      - Tidak dapat berupa nama snapshot yang sudah ada.
      - Panjangnya adalah [1—63] huruf, angka, atau tanda hubung.
      - Karakter pertama harus berupa huruf.

- Tidak dapat diakhiri dengan tanda hubung atau berisi dua tanda hubung berurutan.
- Harus unik untuk semua cluster di Amazon RDS, Neptunus, dan Amazon DocumentDB per, per Wilayah. Akun AWS

c. Salin tag — Untuk menyalin tag apa pun yang Anda miliki di snapshot sumber ke salinan snapshot Anda, pilih Salin tag.

5. Lengkapi Encryption-at-restbagian ini.

a. Enkripsi saat istirahat — Jika snapshot Anda tidak dienkripsi, opsi ini tidak tersedia untuk Anda karena Anda tidak dapat membuat salinan terenkripsi dari snapshot yang tidak terenkripsi. Jika snapshot Anda dienkripsi, Anda dapat mengubah yang AWS KMS key digunakan selama enkripsi saat istirahat.

Untuk informasi selengkapnya tentang mengenkripsi salinan snapshot, lihat. Salin enkripsi snapshot cluster

Untuk informasi selengkapnya tentang enkripsi diam, lihat Mengenkripsi data Amazon DocumentDB saat istirahat.

b. AWS KMS Kunci - Dari daftar drop-down, pilih salah satu dari berikut ini:

- (default) aws/rds - Nomor akun dan ID AWS KMS kunci tercantum mengikuti opsi ini.
- <some-key-name> — Jika Anda membuat kunci, kunci tersebut tercantum dan tersedia untuk Anda pilih.
- Masukkan ARN kunci — Dalam kotak ARN, masukkan Amazon Resource Name (ARN) untuk kunci AWS KMS Anda. Format ARN adalah `arn:aws:kms:<region>:<accountID>:key/<key-id>` .

6. Untuk membuat salinan snapshot yang dipilih, pilih Salin snapshot. Sebagai alternatif, Anda dapat memilih Batalkan untuk tidak membuat salinan snapshot.

Using the AWS CLI

Untuk membuat salinan snapshot klaster yang tidak terenkripsi menggunakan AWS CLI, gunakan operasi `copy-db-cluster-snapshot` dengan parameter berikut. Jika Anda menyalin snapshot ke yang lain Wilayah AWS, jalankan perintah di mana snapshot akan disalin. Wilayah AWS

- **`--source-db-cluster-snapshot-identifier`** — Diperlukan. Pengidentifikasi snapshot klaster untuk membuat salinan. Snapshot klaster harus ada dan dalam status tersedia. Jika Anda menyalin snapshot ke yang lain Wilayah AWS atau jika Anda menyalin snapshot cluster

bersama, pengenal ini harus dalam format ARN dari snapshot cluster sumber. Parameter ini tidak peka huruf besar kecil.

- **`--target-db-cluster-snapshot-identifier`** — Diperlukan. Pengidentifikasi snapshot klaster baru yang akan dibuat dari snapshot klaster sumber. Parameter ini tidak peka huruf besar kecil.

  Batasan penamaan snapshot target:

  - Tidak dapat berupa nama snapshot yang sudah ada.
  - Panjangnya adalah [1—63] huruf, angka, atau tanda hubung.
  - Karakter pertama harus berupa huruf.
  - Tidak dapat diakhiri dengan tanda hubung atau berisi dua tanda hubung berurutan.
  - Harus unik untuk semua cluster di Amazon RDS, Neptunus, dan Amazon DocumentDB per, per Wilayah. Akun AWS

- **`--source-region`**— Jika Anda menyalin snapshot ke yang lain Wilayah AWS, tentukan dari Wilayah AWS mana snapshot cluster terenkripsi akan disalin.

  Jika Anda menyalin snapshot ke Wilayah AWS lain dan Anda tidak menentukan `--source-region`, Anda harus menentukan opsi `pre-signed-url` sebagai gantinya. `pre-signed-url`Nilai harus berupa URL yang berisi permintaan tanda tangan Versi 4 Tanda Tangan untuk `CopyDBClusterSnapshot` tindakan yang akan dipanggil di sumber Wilayah AWS tempat snapshot cluster disalin. Untuk mempelajari lebih lanjut tentang`pre-signed-url`, lihat CopyDBClusterSnapshot.

- **`--kms-key-id`** — Pengidentifikasi kunci KMS agar kunci dapat digunakan untuk mengenkripsi salinan snapshot klaster.

  Jika Anda menyalin snapshot cluster terenkripsi ke yang lain Wilayah AWS, parameter ini diperlukan. Anda harus menentukan kunci KMS untuk tujuan Wilayah AWS.

  Jika Anda menyalin snapshot cluster terenkripsi dalam hal yang sama Wilayah AWS, parameter AWS KMS kuncinya adalah opsional. Salinan snapshot cluster dienkripsi dengan AWS KMS kunci yang sama dengan snapshot cluster sumber. Jika Anda ingin menentukan kunci AWS KMS enkripsi baru yang akan digunakan untuk mengenkripsi salinan, Anda dapat melakukannya menggunakan parameter ini.

- **`--copy-tags`**— Opsional. Tag dan nilai yang akan disalin.

Untuk membatalkan operasi penyalinan setelah ia berlangsung, Anda dapat menghapus snapshot target yang diidentifikasi oleh `--target-db-cluster-snapshot-identifier` atau `TargetDBClusterSnapshotIdentifier` saat snapshot klaster tersebut berada dalam status menyalin.

Example

Contoh 1: Salin snapshot yang tidak terenkripsi ke Wilayah yang sama

AWS CLI Contoh berikut membuat salinan `sample-cluster-snapshot` bernama `sample-cluster-snapshot-copy` Wilayah AWS sama dengan snapshot sumber. Saat salinan dibuat, semua tag di snapshot asli disalin ke salinan snapshot.

Untuk Linux, macOS, atau 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
```

Untuk 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
```

Output dari operasi ini terlihat seperti berikut.

```
{
    "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

Contoh 2: Salin snapshot yang tidak terenkripsi Wilayah AWS

AWS CLI Contoh berikut membuat salinan`sample-cluster-snapshot`, yang memiliki ARN`arn:aws:rds:us-east-1:123456789012:cluster-snapshot:sample-cluster-snapshot`. Salinan ini diberi nama `sample-cluster-snapshot-copy` dan berada Wilayah AWS di tempat perintah dijalankan.

Untuk Linux, macOS, atau 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
```

Untuk 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
```

Output dari operasi ini terlihat seperti berikut.

```
{
```

```
    "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

Contoh 3: Salin snapshot terenkripsi Wilayah AWS

AWS CLI Contoh berikut membuat salinan `sample-cluster-snapshot` dari Wilayah us-west-2 ke Wilayah us-east-1. Perintah ini dipanggil di Wilayah us-east-1.

Untuk Linux, macOS, atau 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
```

Untuk 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
```

Output dari operasi ini terlihat seperti berikut.

```
{
    "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

Contoh 4: Salin snapshot bersama yang tidak terenkripsi Wilayah AWS

AWS CLI Contoh berikut, akun -123456789012, membuat salinan snapshot cluster tidak terenkripsi yang `sample-cluster-snapshot` dibagikan oleh akun - dari Wilayah us-east-1

999999999999 ke Wilayah us-west-2. Perintah ini disebut di Wilayah us-west-2. Untuk informasi selengkapnya tentang berbagi snapshot, lihat Berbagi snapshot.

Untuk Linux, macOS, atau 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
```

Untuk 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
```

Output dari operasi ini terlihat seperti berikut.

```
{
 "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

Contoh 5: Salin snapshot bersama terenkripsi Wilayah AWS

AWS CLI Contoh berikut, akun - `123456789012` membuat salinan snapshot cluster terenkripsi yang `sample-cluster-snapshot` dibagikan oleh akun - dari Wilayah us-east-1 `999999999999` ke Wilayah us-west-2. Snapshot target dienkripsi dengan kunci KMS yang dikelola pelanggan -. `arn:aws:kms:us-west-2:123456789012:key/6c1f3264-1797-472b-ba37-03011e682d28` Perintah ini disebut di Wilayah us-west-2. Untuk informasi selengkapnya tentang berbagi snapshot, lihat Berbagi snapshot.

Untuk Linux, macOS, atau 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
```

Untuk 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
```

Output dari operasi ini terlihat seperti berikut.

```
{
 "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**
>
> Untuk informasi selengkapnya tentang mengenkripsi salinan snapshot, lihat. Salin enkripsi snapshot cluster
> Untuk informasi selengkapnya tentang enkripsi diam, lihat Mengenkripsi data Amazon DocumentDB saat istirahat.

Berbagi snapshot cluster Amazon DocumentDB

Menggunakan Amazon DocumentDB, Anda dapat membagikan snapshot cluster manual dengan cara berikut:

- Berbagi snapshot cluster manual, baik terenkripsi atau tidak terenkripsi, memungkinkan akun resmi AWS untuk menyalin snapshot.
- Berbagi snapshot cluster manual, baik terenkripsi atau tidak terenkripsi, memungkinkan AWS akun resmi untuk secara langsung memulihkan cluster dari snapshot alih-alih mengambil salinannya dan memulihkannya.

> **Note**
>
> Untuk membagikan snapshot cluster otomatis, buat snapshot cluster manual dengan menyalin snapshot otomatis, lalu bagikan salinan itu. Proses ini juga berlaku untuk sumber daya yang AWS dihasilkan oleh Backup.

Anda dapat berbagi snapshot manual dengan hingga 20 lainnya Akun AWS. Anda juga dapat berbagi snapshot manual yang tidak terenkripsi sebagai publik sehingga snapshot tersebut tersedia untuk semua akun . Saat membagikan snapshot sebagai publik, pastikan bahwa tidak ada informasi privat yang dimasukkan ke dalam snapshot publik Anda.

Saat berbagi snapshot manual dengan yang lain Akun AWS, dan Anda memulihkan cluster dari snapshot bersama menggunakan AWS CLI atau Amazon DocumentDB API, Anda harus menentukan Nama Sumber Daya Amazon (ARN) dari snapshot bersama sebagai pengenal snapshot.

Berbagi snapshot terenkripsi

Pembatasan berikut berlaku untuk berbagi snapshot terenkripsi:

- Anda tidak dapat berbagi snapshot terenkripsi sebagai publik.
- Anda tidak dapat membagikan snapshot yang telah dienkripsi menggunakan kunci AWS KMS enkripsi default akun yang membagikan snapshot.

Ikuti langkah-langkah ini untuk berbagi snapshot terenkripsi.

1. Bagikan kunci enkripsi AWS Key Management Service (AWS KMS) yang digunakan untuk mengenkripsi snapshot dengan akun apa pun yang Anda inginkan untuk dapat mengakses snapshot.

   Anda dapat berbagi kunci AWS KMS enkripsi dengan AWS akun lain dengan menambahkan akun lain ke kebijakan AWS KMS kunci. Untuk detail tentang memperbarui kebijakan utama, lihat Menggunakan Kebijakan Utama di AWS KMS di Panduan AWS Key Management Service Pengembang. Untuk contoh pembuatan kebijakan kunci, lihat Membuat kebijakan IAM untuk memungkinkan penyalinan snapshot yang dienkripsi nanti dalam topik ini.
2. Gunakan AWS CLI, seperti yang ditunjukkan di bawah ini, untuk membagikan snapshot terenkripsi dengan akun lain.

Memungkinkan akses ke AWS KMS kunci enkripsi

Akun AWS Agar orang lain dapat menyalin snapshot terenkripsi yang dibagikan dari akun Anda, akun yang Anda bagikan snapshot Anda harus memiliki akses ke AWS KMS kunci yang mengenkripsi snapshot. Untuk mengizinkan akun lain mengakses AWS KMS kunci, perbarui kebijakan kunci untuk AWS KMS kunci dengan ARN akun yang Anda bagikan sebagai prinsipal dalam kebijakan utama AWS KMS . Kemudian izinkan tindakan `kms:CreateGrant`.

Setelah Anda memberikan akses akun ke kunci AWS KMS enkripsi Anda, untuk menyalin snapshot terenkripsi Anda, akun tersebut harus membuat pengguna AWS Identity and Access Management (IAM) jika belum memilikinya. Selain itu, akun itu juga harus melampirkan kebijakan IAM ke pengguna IAM yang memungkinkan pengguna untuk menyalin snapshot terenkripsi menggunakan kunci Anda. AWS KMS Akun harus pengguna IAM dan tidak dapat menjadi Akun AWS identitas root karena pembatasan AWS KMS keamanan.

Dalam contoh kebijakan kunci berikut, pengguna 123451234512 adalah pemilik kunci enkripsi. AWS KMS Pengguna 123456789012 adalah akun yang kuncinya sedang dibagikan dengannya. Kebijakan kunci yang diperbarui ini memberikan akses akun ke AWS KMS kunci. Ini dilakukan dengan memasukkan ARN untuk Akun AWS identitas root untuk pengguna 123456789012 sebagai prinsipal untuk kebijakan, dan dengan mengizinkan tindakan. `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}}
        }
    ]
}
```

Membuat kebijakan IAM untuk memungkinkan penyalinan snapshot yang dienkripsi

Ketika eksternal Akun AWS memiliki akses ke AWS KMS kunci Anda, pemilik akun tersebut dapat membuat kebijakan untuk mengizinkan pengguna IAM yang dibuat untuk akun tersebut menyalin snapshot terenkripsi yang dienkripsi dengan kunci tersebut. AWS KMS

Contoh berikut menunjukkan kebijakan yang dapat dilampirkan ke pengguna IAM untuk Akun AWS 123456789012. Kebijakan ini memungkinkan pengguna IAM untuk menyalin snapshot bersama dari akun 123451234512 yang telah dienkripsi dengan kunci di Wilayah 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
                }
            }
        }
    ]
}
```

Untuk detail tentang memperbarui kebijakan kunci, lihat Menggunakan Kebijakan Kunci di AWS KMS dalam Panduan Developer AWS Key Management Service .

Berbagi snapshot

Anda dapat membagikan snapshot kluster manual Amazon DocumentDB (atau salinan snapshot otomatis) menggunakan atau: Konsol Manajemen AWS AWS CLI

Using the Konsol Manajemen AWS

Untuk membagikan snapshot menggunakan Konsol Manajemen AWS, selesaikan langkah-langkah berikut:

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Snapshot.
3. Pilih snapshot manual yang ingin Anda bagikan.
4. Di menu tarik-turun Tindakan, pilih Bagikan.
5. Pilih salah satu opsi berikut untuk visibilitas snapshot DB:

   - Jika sumbernya tidak terenkripsi, pilih Publik untuk mengizinkan semua AWS akun memulihkan klaster dari snapshot manual Anda. Atau pilih Private untuk mengizinkan hanya AWS akun yang Anda tentukan untuk memulihkan cluster dari snapshot manual Anda.

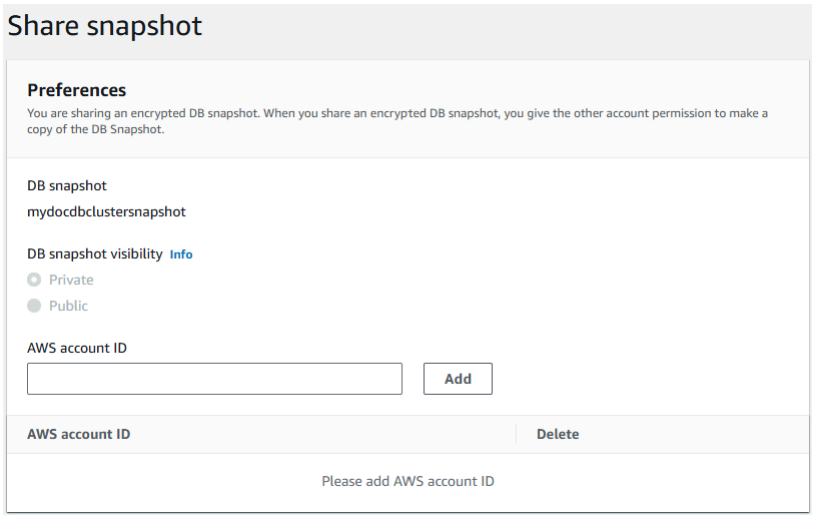
     > **Warning**
     >
     > Jika Anda mengatur visibilitas snapshot DB ke Publik, semua AWS akun dapat memulihkan cluster dari snapshot manual Anda dan memiliki akses ke data Anda. Jangan membagikan snapshot cluster manual yang berisi informasi pribadi sebagai Publik.

   - Jika sumbernya dienkripsi, Visibilitas snapshot DB ditetapkan sebagai Privat karena snapshot terenkripsi tidak dapat dibagikan sebagai publik.

     > **Note**
     >
     > Snapshot yang telah dienkripsi dengan default tidak AWS KMS key dapat dibagikan.

6. Untuk ID AWS Akun, masukkan pengenal AWS akun untuk akun yang ingin Anda izinkan untuk memulihkan klaster dari snapshot manual Anda, lalu pilih Tambah. Ulangi untuk menyertakan pengenal AWS akun tambahan, hingga 20 AWS akun.

   Jika Anda membuat kesalahan saat menambahkan pengenal AWS akun ke daftar akun yang diizinkan, Anda dapat menghapusnya dari daftar dengan memilih Hapus di sebelah kanan pengenal AWS akun yang salah.



7. Setelah Anda menambahkan pengenal untuk semua AWS akun yang ingin Anda izinkan untuk memulihkan snapshot manual, pilih Simpan untuk menyimpan perubahan Anda.

Using the AWS CLI

Untuk berbagi snapshot menggunakan AWS CLI, gunakan operasi Amazon DocumentDB`modify-db-snapshot-attribute`. Gunakan `--values-to-add` parameter untuk menambahkan daftar ID untuk Akun AWS yang berwenang untuk mengembalikan snapshot manual.

Contoh berikut memungkinkan dua Akun AWS pengidentifikasi, 123451234512 dan 123456789012, untuk mengembalikan snapshot bernama. `manual-snapshot1` Itu juga menghapus nilai atribut `all` untuk menandai snapshot sebagai privat.

Untuk Linux, macOS, atau Unix:

```
aws docdb modify-db-cluster-snapshot-attribute \
    --db-cluster-snapshot-identifier sample-cluster-snapshot \
    --attribute-name restore \
    --values-to-add '["123451234512","123456789012"]'
```

Untuk Windows:

```
aws docdb modify-db-cluster-snapshot-attribute ^
    --db-cluster-snapshot-identifier sample-cluster-snapshot ^
    --attribute-name restore ^
    --values-to-add '["123451234512","123456789012"]'
```

Output dari operasi ini terlihat seperti berikut.

```
{
    "DBClusterSnapshotAttributesResult": {
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot",
        "DBClusterSnapshotAttributes": [
            {
                "AttributeName": "restore",
                "AttributeValues": [
                    "123451234512",
                    "123456789012"
                ]
            }
        ]
    }
}
```

Untuk menghapus Akun AWS pengenal dari daftar, gunakan `--values-to-remove` parameter. Contoh berikut mencegah Akun AWS ID 123456789012 memulihkan snapshot.

Untuk Linux, macOS, atau Unix:

```
aws docdb modify-db-cluster-snapshot-attribute \
    --db-cluster-snapshot-identifier sample-cluster-snapshot \
    --attribute-name restore \
    --values-to-remove '["123456789012"]'
```

Untuk Windows:

```
aws docdb modify-db-cluster-snapshot-attribute ^
    --db-cluster-snapshot-identifier sample-cluster-snapshot ^
    --attribute-name restore ^
    --values-to-remove '["123456789012"]'
```

Output dari operasi ini terlihat seperti berikut.

```
{
    "DBClusterSnapshotAttributesResult": {
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot",
        "DBClusterSnapshotAttributes": [
            {
```

```
                "AttributeName": "restore",
                "AttributeValues": [
                    "123451234512"
                ]
            }
        ]
    }
}
```

Memulihkan dari snapshot cluster

Amazon DocumentDB (dengan kompatibilitas MongoDB) membuat snapshot klaster dari volume penyimpanan Anda. Anda dapat membuat klaster baru dengan memulihkankannya dari snapshot klaster. Saat Anda memulihkan klaster, berikan nama snapshot klaster untuk memulihkannya dan nama untuk klaster baru yang dibuat oleh pemulihan. Anda tidak dapat memulihkan dari snapshot ke klaster yang ada karena klaster baru dibuat saat Anda memulihkan.

Ketika Anda memulihkan sebuah klaster dari snapshot klaster:

- Tindakan ini memulihkan hanya klaster, dan bukan instans untuk klaster tersebut. Anda harus memanggil tindakan `create-db-instance` guna membuat instans untuk klaster yang dipulihkan, menentukan pengidentifikasi klaster yang dipulihkan di `--db-cluster-identifier`. Anda dapat membuat instans hanya setelah klaster tersedia.
- Anda tidak dapat memulihkan snapshot terenkripsi ke klaster yang tidak terenkripsi. Namun, Anda dapat memulihkan snapshot yang tidak terenkripsi ke cluster terenkripsi dengan menentukan kuncinya. AWS KMS
- Untuk memulihkan cluster dari snapshot terenkripsi, Anda harus memiliki akses ke kunci. AWS KMS

> **Note**
>
> Anda tidak dapat memulihkan klaster 3.6 ke klaster 4.0 tetapi Anda dapat bermigrasi dari satu versi klaster ke versi lainnya. Untuk informasi selengkapnya, kunjungi Migrasi ke Amazon DocumentDB.

Using the Konsol Manajemen AWS

Prosedur berikut menunjukkan cara untuk memulihkan klaster Amazon DocumentDB dari snapshot klaster menggunakan Konsol Manajemen Amazon DocumentDB.

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Snapshot, lalu pilih tombol di sebelah kiri snapshot yang ingin Anda gunakan untuk memulihkan klaster.

   > **Tip**
   >
   > Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu (≡) di pojok kiri atas halaman.

3. Pada menu Tindakan, pilih Pemulihan.
4. Pada halaman Pulihkan snapshot, selesaikan bagian Konfigurasi.

   a. Pengidentifikasi klaster — Nama untuk klaster baru. Anda dapat menerima nama yang disediakan Amazon DocumentDB atau mengetikkan nama yang Anda inginkan. Nama yang disediakan Amazon DocumentDB adalah dalam format `docdb-` ditambah stempel waktu UTC; misalnya, `docdb-yyyy-mm-dd-hh-mm-ss`.

   b. Kelas instans — Kelas instans untuk klaster baru. Anda dapat menerima kelas instans default atau memilih kelas instans dari daftar tarik-turun.

   c. Jumlah instans — Jumlah instans yang ingin Anda buat dengan klaster ini. Anda dapat menerima default dari 3 instance (1 replika primer read/write dan 2 read-only) atau memilih jumlah instance dari daftar drop-down.

5. Untuk konfigurasi penyimpanan Cluster, pilih opsi penyimpanan.

   > **Note**
   >
   > Konfigurasi penyimpanan Amazon I/O-Optimized DocumentDB hanya tersedia di versi mesin Amazon DocumentDB 5.0 dan 8.0.

6. Jika Anda puas dengan konfigurasi klaster, pilih Pulihkan klaster dan tunggu sementara klaster Anda dipulihkan.

7. Jika Anda lebih suka untuk mengubah beberapa konfigurasi, seperti menentukan Amazon VPC atau grup keamanan non-default, pilih Tampilkan pengaturan lanjutan dari kiri bawah halaman, dan kemudian lanjutkan dengan langkah-langkah berikut.

   a. Selesaikan bagian Pengaturan jaringan.

      - Virtual Private Cloud (VPC) — Terima VPC saat ini, atau pilih VPC dari daftar tarik-turun.
      - Grup Subnet — Terima grup subnet `default`, atau pilih salah satu dari daftar tarik-turun.
      - Grup Keamanan VPC — Terima grup keamanan `default (VPC)`, atau pilih salah satu dari daftar.

   b. Lengkapi bagian Opsi klaster.

      - Port basis data — Terima port default, `27017`, atau gunakan panah atas atau bawah untuk mengatur port yang ingin Anda gunakan untuk koneksi aplikasi.

   c. Lengkapi bagian Enkripsi.

      - Enkripsi saat istirahat — Jika snapshot Anda dienkripsi, opsi ini tidak tersedia untuk Anda. Jika itu tidak dienkripsi, Anda dapat memilih salah satu hal berikut:
        - Untuk mengenkripsi semua data klaster Anda, pilih Aktifkan enkripsi saat istirahat. Jika Anda memilih opsi ini, Anda harus menunjuk kunci KMS.
        - Untuk tidak mengenkripsi data klaster Anda, pilih Nonaktifkan enkripsi saat istirahat. Jika Anda memilih opsi ini, Anda telah selesai dengan bagian enkripsi.
      - AWS KMS Kunci - Pilih salah satu dari berikut ini dari daftar drop-down:
        - (default) aws/rds - Nomor akun dan ID AWS KMS kunci tercantum mengikuti opsi ini.
        - Customer-managed kunci — Opsi ini hanya tersedia jika Anda membuat kunci enkripsi IAM di konsol AWS Identity and Access Management (IAM). Anda dapat memilih kunci untuk mengenkripsi klaster Anda.
        - Masukkan kunci ARN — Di kotak ARN, masukkan Nama Sumber Daya Amazon (ARN) untuk kunci Anda. AWS KMS Format ARN adalah `arn:aws:kms:<region>:<accountID>:key/<key-id>`.

   d. Lengkapi bagian Ekspor log.

      - Pilih jenis log yang akan dipublikasikan CloudWatch - Pilih salah satu dari berikut ini:

- Diaktifkan - Memungkinkan cluster Anda untuk mengekspor logging DDL ke Amazon CloudWatch Logs.
- Dinonaktifkan - Mencegah klaster Anda mengekspor log DDL ke Amazon CloudWatch Logs. Dinonaktifkan adalah default.

- IAM role — Dari daftar, pilih Peran Tertaut Layanan RDS.

e. Lengkapi bagian Tag.

- Tambahkan Tag — Di kotak Kunci, masukkan nama untuk tag untuk klaster Anda. Di kotak Nilai, secara opsional masukkan nilai tag. Tag digunakan dengan kebijakan AWS Identity and Access Management (IAM) untuk mengelola akses ke sumber daya Amazon DocumentDB dan untuk mengontrol tindakan apa yang dapat diterapkan pada sumber daya.

f. Lengkapi bagian Perlindungan penghapusan.

- Aktifkan perlindungan penghapusan — Melindungi klaster agar tidak terhapus secara tidak sengaja. Saat opsi ini diaktifkan, Anda tidak dapat menghapus klaster.

8. Pilih Pulihkan klaster.

Using the AWS CLI

Untuk mengembalikan cluster dari snapshot menggunakan AWS CLI, gunakan `restore-db-cluster-from-snapshot` operasi dengan parameter berikut. Untuk informasi selengkapnya, lihat RestoreDBClusterFromSnapshot.

- **`--db-cluster-identifier`** — Diperlukan. Nama klaster yang dibuat oleh operasi. Sebuah klaster dengan nama ini tidak boleh ada sebelum operasi ini.

  Aturan penamaan klaster:

  - Panjangnya adalah [1—63] huruf, angka, atau tanda hubung.
  - Karakter pertama harus berupa huruf.
  - Tidak dapat diakhiri dengan tanda hubung atau berisi dua tanda hubung berurutan.
  - Harus unik untuk semua cluster di Amazon RDS, Neptunus, dan Amazon DocumentDB per, per Wilayah. Akun AWS

- **`--snapshot-identifier`** — Diperlukan. Nama snapshot yang digunakan untuk memulihkan. Snapshot dengan nama ini harus ada dan dalam status tersedia.

- **`--engine`** — Diperlukan. Harus berupa `docdb`.
- **`--storage-type standard | iopt1`**— Opsional. Default: `standard`.
- **`--kms-key-id`**— Opsional. ARN pengenal AWS KMS kunci yang akan digunakan saat memulihkan snapshot terenkripsi atau mengenkripsi cluster saat memulihkan dari snapshot yang tidak terenkripsi. Menyediakan ID AWS KMS kunci menghasilkan cluster yang dipulihkan dienkripsi dengan AWS KMS kunci, terlepas dari apakah snapshot dienkripsi atau tidak.

  Format `--kms-key-id` adalah `arn:aws:kms:<region>:<accountID>:key/<key-id>`. Jika Anda tidak menentukan nilai untuk parameter `--kms-key-id`, maka hal berikut akan terjadi:

  - Jika snapshot `--snapshot-identifier` dienkripsi, maka cluster yang dipulihkan dienkripsi menggunakan AWS KMS kunci yang sama yang digunakan untuk mengenkripsi snapshot.
  - Jika snapshot di `--snapshot-identifier` tidak dienkripsi, maka klaster yang dipulihkan tidak dienkripsi.

Untuk Linux, macOS, atau 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
```

Untuk 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
```

Output dari operasi ini terlihat seperti berikut.

```
{
    "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
    }
}
```

Setelah status klaster tersedia, buat setidaknya satu instans untuk klaster.

Untuk Linux, macOS, atau 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
```

Untuk 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
```

Output dari operasi ini terlihat seperti berikut.

```
{
    "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"
            }
        ],
        "AavailabilityZone": "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"
    }
}
```

Memulihkan ke titik waktu

Anda dapat memulihkan klaster ke titik waktu mana pun yang berada dalam periode retensi cadangan klaster menggunakan Konsol Manajemen AWS or AWS Command Line Interface (AWS CLI).

> **Note**
>
> Anda tidak dapat melakukan pemulihan titik waktu dari klaster 3.6 ke klaster 4.0 tetapi Anda dapat bermigrasi dari satu versi klaster ke versi lainnya. Untuk informasi selengkapnya, kunjungi Migrasi ke Amazon DocumentDB.

Ingatlah hal-hal berikut saat memulihkan klaster ke titik waktu tertentu.

- Klaster baru dibuat dengan konfigurasi yang sama seperti klaster sumber, kecuali klaster baru dibuat dengan grup parameter default. Untuk mengatur grup parameter klaster baru ke grup

parameter klaster sumber, modifikasi klaster setelah tersedia. Untuk informasi selengkapnya tentang memodifikasi klaster, lihat Memodifikasi cluster Amazon DocumentDB.

Using the Konsol Manajemen AWS

Anda dapat memulihkan klaster ke titik waktu dalam periode retensi pencadangannya dengan menyelesaikan hal berikut menggunakan Konsol Manajemen AWS.

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Pada panel navigasi, silakan pilih Klaster. Dalam daftar klaster, pilih tombol di sebelah kiri pada klaster yang ingin Anda pulihkan.

   > **Tip**
   >
   > Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu (≡) di pojok kiri atas halaman.

3. Pada menu Tindakan, pilih Pulihkan ke titik waktu.
4. Lengkapi bagian Waktu pemulihan, yang menentukan tanggal dan waktu untuk memulihkan ke.

   a. Tanggal pemulihan — Pilih atau masukkan tanggal yang berada antara Waktu pemulihan paling awal dan Waktu pemulihan terbaru.

   b. Waktu pemulihan— Pilih atau masukkan jam, menit, dan detik yang berada antara Waktu pemulihan paling awal dan Waktu pemulihan terbaru.

5. Lengkapi bagian Konfigurasi.

   a. Pengidentifikasi klaster — Terima pengidentifikasi default, atau masukkan pengidentifikasi yang Anda inginkan.

      Aturan penamaan klaster:

      - Panjangnya adalah [1—63] huruf, angka, atau tanda hubung.
      - Karakter pertama harus berupa huruf.
      - Tidak dapat diakhiri dengan tanda hubung atau berisi dua tanda hubung berurutan.

- Harus unik untuk semua cluster di Amazon RDS, Neptunus, dan Amazon DocumentDB per, per Wilayah. Akun AWS

b. Kelas instans — Dari daftar tarik-turun, pilih kelas instans yang Anda inginkan untuk instans klaster.

c. Jumlah instans — Dari daftar tarik-turun, pilih jumlah instans yang Anda ingin buat ketika klaster dipulihkan.

6. Untuk konfigurasi penyimpanan Cluster, pilih opsi penyimpanan.

> **Note**
>
> Konfigurasi penyimpanan Amazon I/O-Optimized DocumentDB hanya tersedia di versi mesin Amazon DocumentDB 5.0 dan 8.0.

7. Tidak wajib. Untuk mengonfigurasi pengaturan jaringan, opsi klaster, dan mengaktifkan ekspor log, pilih Tampilkan pengaturan lanjutan, dan kemudian lengkapi bagian berikut. Jika tidak, lanjutkan dengan langkah berikutnya.

- Pengaturan jaringan
  1. Virtual Private Cloud (VPC) — Dari daftar tarik-turun, pilih VPC yang ingin Anda gunakan untuk klaster ini.
  2. Grup Subnet — Dari daftar tarik-turun, pilih grup subnet untuk klaster ini.
  3. Grup keamanan VPC — Dari daftar tarik-turun, pilih grup keamanan VPC untuk klaster ini.

- Opsi cluster
  1. Port — Terima port default (27017), atau gunakan panah atas dan bawah untuk mengatur port untuk berkomunikasi dengan klaster ini.

- Ekspor log
  1. Log audit - Pilih opsi ini untuk mengaktifkan mengekspor log audit ke Amazon CloudWatch Logs. Jika Anda memilih opsi ini, Anda harus mengaktifkan `audit_logs` di grup parameter kustom klaster. Untuk informasi selengkapnya, lihat Mengaudit acara Amazon DocumentDB.

2. Log profiler - Pilih opsi ini untuk mengaktifkan mengekspor log profiler operasi ke Amazon Logs. CloudWatch Jika Anda memilih opsi ini, Anda juga harus mengubah parameter berikut dalam grup parameter kustom klaster:
   - `profiler` — Atur ke `enabled`.
   - `profiler_threshold_ms` — Atur ke nilai `[0-INT_MAX]` untuk mengatur ambang batas untuk operasi pembuatan profil.
   - `profiler_sampling_rate` — Atur ke nilai `[0.0-1.0]` untuk mengatur persentase operasi lambat ke profil.

   Untuk informasi selengkapnya, lihat Membuat profil operasi Amazon DocumentDB.
3. Log profiler - Ekspor log profiler ke Amazon CloudWatch
4. IAM role — Dari daftar tarik-turun, pilih Peran Tertaut Layanan RDS.

- Tanda
  1. Tambahkan Tag — Di kotak Kunci, masukkan nama untuk tag untuk klaster Anda. Di kotak Nilai, secara opsional masukkan nilai tag. Tag digunakan dengan kebijakan AWS Identity and Access Management (IAM) untuk mengelola akses ke sumber daya Amazon DocumentDB dan untuk mengontrol tindakan apa yang dapat diterapkan ke sumber daya.
- Perlindungan penghapusan
  1. Aktifkan perlindungan penghapusan — Melindungi klaster agar tidak terhapus secara tidak sengaja. Saat opsi ini diaktifkan, Anda tidak dapat menghapus klaster.

8. Untuk memulihkan klaster, pilih Buat klaster. Atau, Anda dapat memilih Batalkan untuk membatalkan operasi.

Using the AWS CLI

Untuk memulihkan klaster ke titik waktu menggunakan periode retensi cadangan snapshot, gunakan operasi `restore-db-cluster-to-point-in-time` dengan parameter berikut.

- **`--db-cluster-identifier`** — Diperlukan. Nama klaster baru yang akan dibuat. Klaster ini tidak boleh ada sebelum operasi. Nilai parameter harus memenuhi aturan berikut.

  Aturan penamaan klaster:

  - Panjangnya adalah [1—63] huruf, angka, atau tanda hubung.
  - Karakter pertama harus berupa huruf.

  - Tidak dapat diakhiri dengan tanda hubung atau berisi dua tanda hubung berurutan.
  - Harus unik untuk semua cluster di Amazon RDS, Neptunus, dan Amazon DocumentDB per, per Wilayah. Akun AWS
- **`--restore-to-time`** — Tanggal dan waktu UTC untuk memulihkan klaster. Misalnya, `2018-06-07T23:45:00Z`.

  Aturan Waktu:

  - Harus sebelum waktu pemulihan terbaru untuk klaster.
  - Harus ditentukan jika parameter `--use-latest-restorable-time` tidak disediakan.
  - Tidak dapat ditentukan jika parameter `--use-latest-restorable-time` adalah `true`.
  - Tidak dapat ditentukan jika nilai parameter `--restore-type` adalah `copy-on-write`.
- **`--source-db-cluster-identifier`** — Nama klaster sumber yang akan dipulihkan. Klaster ini harus ada dan tersedia.
- **`--use-latest-restorable-time`** atau **`--no-use-latest-restorable-time`** — Apakah untuk memulihkan ke waktu backup terbaru yang dapat dipulihkan. Tidak dapat ditentukan jika parameter `--restore-to-time` disediakan.
- **`--storage-type standard | iopt1`**— Opsional. Default: `standard`.

AWS CLI Operasi `restore-db-cluster-to-point-in-time` hanya mengembalikan cluster, bukan instance untuk cluster itu. Anda harus memanggil operasi `create-db-instance` guna membuat instans untuk klaster yang dipulihkan, menentukan pengidentifikasi klaster yang dipulihkan di `--db-cluster-identifier`. Anda dapat membuat instans hanya setelah operasi `restore-db-cluster-to-point-in-time` telah selesai dan klaster yang dipulihkan tersedia.

Example

Contoh berikut membuat `sample-cluster-restored` dari snapshot `sample-cluster-snapshot` ke waktu pemulihan terbaru.

Untuk Linux, macOS, atau 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
```

Untuk 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

Contoh berikut membuat `sample-cluster-restored` dari snapshot `sample-cluster-snapshot` hingga 03:15 pada 11 Desember 2018 (UTC), yang berada dalam periode retensi cadangan `sample-cluster`.

Untuk Linux, macOS, atau 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
```

Untuk 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
```

Output dari operasi ini terlihat seperti berikut.

```
{
    "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
    }
}
```

Menghapus snapshot cluster

Snapshot manual adalah cadangan lengkap yang dihapus hanya ketika Anda menghapusnya secara manual menggunakan Konsol Manajemen AWS atau AWS CLI. Anda tidak dapat secara manual menghapus snapshot otomatis karena snapshot otomatis dihapus hanya ketika periode retensi snapshot berakhir atau Anda menghapus klaster snapshot.

Using the Konsol Manajemen AWS

Untuk menghapus snapshot cluster manual menggunakan Konsol Manajemen AWS, selesaikan langkah-langkah berikut.

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Snapshot.

> **Tip**
>
> Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu (≡) di pojok kiri atas halaman.

3. Dalam daftar snapshot, pilih tombol di sisi kiri snapshot yang ingin Anda hapus. Jenis snapshot harus manual.

   1. Anda dapat memverifikasi bahwa jenis snapshot adalah manual dengan memeriksa apakah itu terdaftar sebagai `manual` atau `automatic` pada kolom Jenis.

4. Dari menu Tindakan, pilih Hapus. Jika opsi Hapus tidak tersedia, Anda mungkin memilih snapshot otomatis.
5. Pada layar konfirmasi penghapusan, untuk menghapus snapshot, pilih Hapus. Untuk tetap menyimpan snapshot, pilih Batalkan.

Using the AWS CLI

Sebuah snapshot klaster manual Amazon DocumentDB adalah backup lengkap yang dapat Anda hapus secara manual menggunakan AWS CLI. Anda tidak dapat menghapus snapshot otomatis secara manual.

Untuk menghapus snapshot cluster manual menggunakan AWS CLI, gunakan `delete-db-cluster-snapshot` operasi dengan parameter berikut.

Parameter

- **`--db-cluster-snapshot-identifier`** — Diperlukan. Nama snapshot manual yang akan dihapus.

Contoh berikut menghapus snapshot klaster `sample-cluster-snapshot`.

Untuk Linux, macOS, atau Unix:

```
aws docdb delete-db-cluster-snapshot \
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

Untuk Windows:

```
aws docdb delete-db-cluster-snapshot ^
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

Output dari operasi ini mencantumkan detail snapshot klaster yang Anda hapus.

Mengelola sumber daya Amazon DocumentDB

Bagian ini mencakup berbagai komponen dan tugas terkaitnya untuk mengelola implementasi Amazon DocumentDB (dengan kompatibilitas MongoDB) Anda.

Topik

- Ikhtisar tugas operasional Amazon DocumentDB
- Ikhtisar cluster global Amazon DocumentDB
- Mengelola cluster Amazon DocumentDB
- Mengelola instans Amazon DocumentDB
- Mengelola grup subnet Amazon DocumentDB
- Amazon DocumentDB Ketersediaan dan replikasi tinggi
- Mengelola indeks Amazon DocumentDB
- Mengelola kompresi dokumen tingkat koleksi
- Mengelola kompresi berbasis kamus di Amazon DocumentDB 8.0
- Mengelola acara Amazon DocumentDB
- Memilih wilayah dan zona ketersediaan
- Mengelola grup parameter cluster Amazon DocumentDB
- Memahami titik akhir Amazon DocumentDB
- Memahami Amazon DocumentDB Amazon Resource Name (ARN)
- Menandai sumber daya Amazon DocumentDB
- Memelihara Amazon DocumentDB
- Memahami peran terkait layanan
- Menggunakan aliran perubahan dengan Amazon DocumentDB
- Menggunakan Collation di Amazon DocumentDB 8.0
- Menggunakan Tampilan di Amazon DocumentDB 8.0
- Penggunaan AWS Lambda dengan aliran perubahan

Ikhtisar tugas operasional Amazon DocumentDB

Bagian ini mencakup tugas operasional untuk klaster Amazon DocumentDB Anda, dan cara menyelesaikan tugas ini menggunakan. AWS CLI

Topik

- Menambahkan replika ke cluster Amazon DocumentDB
- Menjelaskan cluster dan instance
- Membuat snapshot cluster
- Memulihkan dari snapshot
- Menghapus instance dari cluster
- Menghapus klaster

Menambahkan replika ke cluster Amazon DocumentDB

Setelah Anda membuat instans primer untuk klaster Amazon DocumentDB, Anda dapat menambahkan satu atau beberapa replika. Replika adalah instans baca-saja yang melayani dua tujuan:

- Skalabilitas — Jika Anda memiliki banyak klien yang memerlukan akses bersamaan, Anda dapat menambahkan lebih banyak replika untuk penskalaan baca.
- Ketersediaan tinggi — Jika instans primer gagal, Amazon DocumentDB secara otomatis gagal ke instans replika dan menetapkannya sebagai primer baru. Jika replika gagal, instans lain di klaster masih dapat melayani permintaan hingga node yang gagal dapat dipulihkan.

Setiap klaster Amazon DocumentDB dapat mendukung hingga 15 replika.

> **Note**
>
> Untuk toleransi kesalahan maksimum, Anda harus menerapkan replika di Availability Zone yang terpisah. Ini membantu memastikan bahwa klaster Amazon DocumentDB Anda dapat terus berfungsi, bahkan jika seluruh Availability Zone menjadi tidak tersedia.

AWS CLI Contoh berikut menunjukkan cara menambahkan replika baru. Parameter `--availability-zone` menempatkan replika di Availability Zone yang ditentukan.

```
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
```

Menjelaskan cluster dan instance

AWS CLI Contoh berikut mencantumkan semua cluster Amazon DocumentDB di Wilayah. Untuk fitur pengelolaan tertentu seperti pengelolaan siklus hidup klaster dan instans, Amazon DocumentDB memanfaatkan teknologi operasi yang dibagikan dengan Amazon RDS. Parameter filter `filterName=engine,Values=docdb` hanya mengembalikan klaster Amazon DocumentDB.

Untuk informasi lebih lanjut tentang mendeskripsikan dan memodifikasi klaster, lihat Siklus hidup klaster Amazon DocumentDB.

```
aws docdb describe-db-clusters --filter Name=engine,Values=docdb
```

Output dari operasi ini terlihat seperti berikut ini.

```
{
    "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",
            ...
        }
    ]
}
```

AWS CLI Contoh berikut mencantumkan instance dalam cluster Amazon DocumentDB. Untuk informasi lebih lanjut tentang mendeskripsikan dan memodifikasi cluster, lihat Siklus hidup instance Amazon DocumentDB.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].[DBClusterMembers]'
```

Output terlihat seperti di bawah ini. Dalam output ini, ada dua instans. Instans primer adalah `sample-instance-1` (`"IsClusterWriter": true`). Ada juga instans replika, `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
            }
        ]
    ]
]
```

Membuat snapshot cluster

Snapshot klaster adalah cadangan lengkap data di klaster Amazon DocumentDB Anda. Saat snapshot sedang dibuat, Amazon DocumentDB membaca data Anda langsung dari volume klaster. Karena ini, Anda dapat membuat snapshot meskipun klaster Anda tidak memiliki instans yang berjalan pada saat itu. Jumlah waktu yang diperlukan untuk membuat snapshot bergantung pada ukuran volume klaster Anda.

Amazon DocumentDB mendukung pencadangan otomatis, yang terjadi setiap hari selama jendela pencadangan pilihan — periode waktu 30 menit pada siang hari. AWS CLI Contoh berikut menunjukkan cara melihat jendela cadangan untuk cluster Anda:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].PreferredBackupWindow'
```

Output menunjukkan jendela cadangan (dalam UTC):

```
[
    "00:18-00:48"
]
```

Anda dapat menentukan jendela pencadangan saat membuat klaster Amazon DocumentDB Anda. Anda juga dapat mengubah jendela pencadangan, seperti yang ditunjukkan pada contoh berikut. Jika Anda tidak menentukan jendela pencadangan, Amazon DocumentDB secara otomatis menugaskan satu ke klaster Anda.

```
aws docdb modify-db-cluster \
```

```
    --db-cluster-identifier sample-cluster \
    --preferred-backup-window "02:00-02:30"
```

Selain pencadangan otomatis, Anda dapat membuat snapshot klaster secara manual kapan saja. Saat melakukannya, Anda menentukan klaster mana yang ingin Anda cadangkan, dan nama unik untuk snapshot Anda sehingga Anda dapat memulihkannya nanti.

AWS CLI Contoh berikut menunjukkan cara membuat snapshot data Anda.

```
aws docdb create-db-cluster-snapshot \
    --db-cluster-identifier sample-cluster \
    --db-cluster-snapshot-identifier sample-cluster-snapshot
```

Memulihkan dari snapshot

Anda dapat memulihkan snapshot klaster ke klaster Amazon DocumentDB baru. Untuk melakukannya, Anda memberikan nama snapshot dan nama klaster baru. Anda tidak dapat memulihkan dari snapshot ke klaster yang ada; sebagai gantinya, Amazon DocumentDB membuat klaster baru saat Anda memulihkan dan kemudian mengisinya dengan data snapshot Anda.

Contoh berikut menunjukkan semua snapshot untuk klaster `sample-cluster`.

```
aws docdb describe-db-cluster-snapshots \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusterSnapshots[*].[DBClusterSnapshotIdentifier,SnapshotType,Status]'
```

Outputnya terlihat seperti berikut ini. Snapshot manual adalah snapshot yang Anda buat secara manual, sedangkan snapshot otomatis dibuat oleh Amazon DocumentDB dalam jendela pencadangan klaster.

```
[
        "sample-cluster-snapshot",
        "manual",
        "available"
    ],
    [
        "rds:sample-cluster",
        "automated",
        "available"
    ]
```

```
]
```

Contoh berikut menunjukkan cara memulihkan klaster Amazon DocumentDB dari snapshot.

```
aws docdb restore-db-cluster-from-snapshot \
    --engine docdb \
    --db-cluster-identifier new-sample-cluster \
    --snapshot-identifier sample-cluster-snapshot
```

Klaster baru tidak memiliki instans yang terkait dengannya; jadi jika Anda ingin berinteraksi dengan klaster, Anda harus menambahkan instans ke dalamnya.

```
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
```

Anda dapat menggunakan AWS CLI operasi berikut untuk memantau kemajuan pembuatan cluster dan instance. Saat status klaster dan instans tersedia, Anda dapat terhubung ke titik akhir klaster baru dan mengakses data Anda.

```
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]'
```

Menghapus instance dari cluster

Amazon DocumentDB menyimpan semua data Anda dalam volume klaster. Data tetap ada dalam volume klaster tersebut, meskipun Anda menghapus semua instans dari klaster Anda. Jika Anda perlu mengakses data lagi, Anda dapat menambahkan instans ke klaster kapan saja, dan melanjutkan dari bagian terakhir yang Anda tinggalkan.

Contoh berikut menunjukkan cara menghapus instans dari klaster Amazon DocumentDB Anda.

```
aws docdb delete-db-instance \
    --db-instance-identifier sample-instance
```

Menghapus klaster

Sebelum Anda dapat menghapus klaster Amazon DocumentDB, Anda harus menghapus semua instansnya terlebih dahulu. AWS CLI Contoh berikut mengembalikan informasi tentang contoh dalam cluster. Jika operasi ini mengembalikan pengidentifikasi instans apa pun, Anda harus menghapus setiap instans. Untuk informasi selengkapnya, lihat Menghapus instance dari cluster.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].DBClusterMembers[*].DBInstanceIdentifier'
```

Jika tidak ada lagi instans yang tersisa, Anda dapat menghapus klaster. Pada saat itu, Anda harus memilih salah satu dari opsi berikut:

- Buat snapshot akhir — Tangkap semua data klaster dalam snapshot sehingga Anda dapat membuat ulang instans baru dengan data tersebut nanti. Contoh berikut menunjukkan cara melakukannya:

  ```
  aws docdb delete-db-cluster \
      --db-cluster-identifier sample-cluster \
      --final-db-snapshot-identifier sample-cluster-snapshot
  ```

- Lewati snapshot terakhir — Buang semua data klaster secara permanen. Ini tidak bisa dibalik. Contoh berikut menunjukkan cara melakukannya:

  ```
  aws docdb delete-db-cluster \
      --db-cluster-identifier sample-cluster \
      --skip-final-snapshot
  ```

Ikhtisar cluster global Amazon DocumentDB

Apa itu klaster global?

Cluster global terdiri dari satu wilayah primer dan hingga 10 wilayah sekunder hanya-baca. Anda menerbitkan operasi tulis langsung ke klaster primer di wilayah primer dan Amazon DocumentDB

secara otomatis mereplikasi data ke wilayah sekunder menggunakan infrastruktur khusus. Latensi biasanya di bawah satu detik.

Bagaimana klaster global berguna?

- Pemulihan dari pemadaman di seluruh wilayah — Jika terjadi pemadaman di seluruh wilayah, Anda dapat mempromosikan salah satu cluster sekunder ke cluster utama dalam beberapa menit, dengan Tujuan Waktu Pemulihan (RTO) khas kurang dari satu menit. Recovery Point Objective (RPO) biasanya diukur dalam hitungan detik, tetapi ini tergantung pada lag di seluruh jaringan pada saat kegagalan.
- Pembacaan global dengan latensi lokal — Jika Anda memiliki kantor di seluruh dunia, Anda dapat menggunakan klaster global untuk terus memperbarui sumber informasi utama Anda di wilayah primer. Kantor di wilayah Anda yang lain dapat mengakses informasi di wilayah mereka sendiri, dengan latensi lokal.
- Klaster sekunder yang dapat diskalakan — Anda dapat menskalakan klaster sekunder dengan menambahkan lebih banyak instans baca-saja ke wilayah sekunder. Klaster sekunder bersifat baca-saja, sehingga dapat mendukung hingga 16 instans replika hanya-baca daripada batas biasa 15 untuk satu klaster.
- Replikasi cepat dari klaster primer ke sekunder — Replikasi yang dilakukan oleh klaster global memiliki dampak performa yang kecil pada klaster basis data primer. Sumber daya instans DB sepenuhnya dikhususkan untuk melayani beban kerja baca dan tulis aplikasi.
- Ubah aliran - Anda dapat membaca aliran perubahan dari klaster sekunder di kluster global, mengurangi beban pada klaster utama, dan memungkinkan Anda menskalakan beban kerja baca dari wilayah sekunder.

Apa keterbatasan terkini klaster global?

- Klaster global tidak didukung di Amazon DocumentDB v3.6.
- Cluster global didukung pada semua jenis instans kecuali db.t3, db.t4g, dan db.r4.
- Cluster global tidak tersedia di Wilayah berikut: Amerika Selatan (São Paulo), Eropa (Milan), Tiongkok (Beijing), dan Tiongkok (Ningxia).
- Switchover dan failover global tidak didukung ketika wilayah menggunakan versi mesin yang berbeda. Failover manual didukung ketika ada ketidakcocokan versi mesin.
- Hanya klaster primer yang melakukan operasi penulisan. Klien yang melakukan operasi tulis terhubung ke titik akhir klaster dari klaster primer.

- Anda dapat memiliki maksimal 10 wilayah sekunder dan satu wilayah utama untuk cluster Anda.
- Klaster sekunder tidak dapat dihentikan. Sebuah klaster primer tidak dapat dihentikan jika ia memiliki klaster sekunder yang terkait dengannya. Hanya klaster wilayah yang tidak memiliki klaster sekunder yang dapat dihentikan.
- Replika yang dilampirkan ke klaster sekunder dapat dimulai ulang dalam keadaan tertentu. Jika instans wilayah primer dimulai ulang atau gagal, replika di region sekunder juga dimulai ulang. Klaster kemudian tidak tersedia sampai semua replika kembali disinkronkan dengan instans penulis klaster basis data primer. Perilaku ini yang diharapkan. Pastikan Anda memahami dampak terhadap klaster global Anda sebelum membuat perubahan pada klaster primer Anda.

Topik

- Panduan memulai cepat: cluster global
- Mengelola klaster global Amazon DocumentDB
- Connect ke klaster global Amazon DocumentDB
- Memantau klaster global Amazon DocumentDB
- Pemulihan bencana dan klaster global Amazon DocumentDB

Panduan memulai cepat: cluster global

Topik

- Konfigurasi
- Membuat klaster global Amazon DocumentDB
- Menambahkan sebuah Wilayah AWS ke cluster global Amazon DocumentDB
- Menggunakan snapshot untuk klaster global Amazon DocumentDB Anda

Konfigurasi

Cluster global Amazon DocumentDB mencakup setidaknya dua. Wilayah AWS Wilayah primer mendukung klaster yang memiliki satu instance (penulis) primer dan hingga 15 instance replika, sedangkan Region sekunder menjalankan cluster hanya-baca yang seluruhnya terdiri dari hingga 16 instance replika. Sebuah cluster global dapat memiliki hingga 10 Wilayah sekunder. Tabel mencantumkan klaster, instans, dan replika maksimum yang diizinkan dalam klaster global.

Deskripsi	Primer Wilayah AWS	Sekunder Wilayah AWS
klaster	1	10 (maksimum)
Instans penulis	1	0
Read-only contoh (replika Amazon DocumentDB), per cluster	15 (maks)	16 (total)
Read-only contoh (maks diperbolehkan, mengingat jumlah sebenarnya dari Wilayah sekunder)	15 - s	s = jumlah total sekunder Wilayah AWS

Klaster memiliki persyaratan khusus berikut:

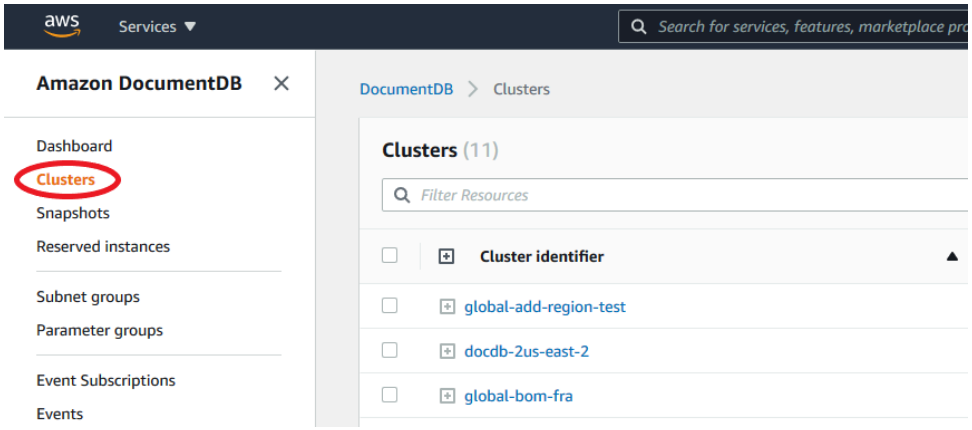
- Persyaratan kelas instance database - Anda hanya dapat menggunakan kelas `db.r5` dan `db.r6g` instance.
- Wilayah AWS persyaratan — Cluster primer harus berada dalam satu Wilayah, dan setidaknya satu cluster sekunder harus berada di Wilayah yang berbeda dari akun yang sama. Anda dapat membuat hingga 10 cluster sekunder (hanya-baca), dan masing-masing harus berada di Wilayah yang berbeda. Dengan kata lain, tidak ada dua cluster yang bisa berada di Wilayah yang sama.
- Persyaratan penamaan — Nama yang Anda pilih untuk setiap cluster Anda harus unik, di semua Wilayah. Anda tidak dapat menggunakan nama yang sama untuk cluster yang berbeda meskipun mereka berada di Wilayah yang berbeda.

Membuat klaster global Amazon DocumentDB

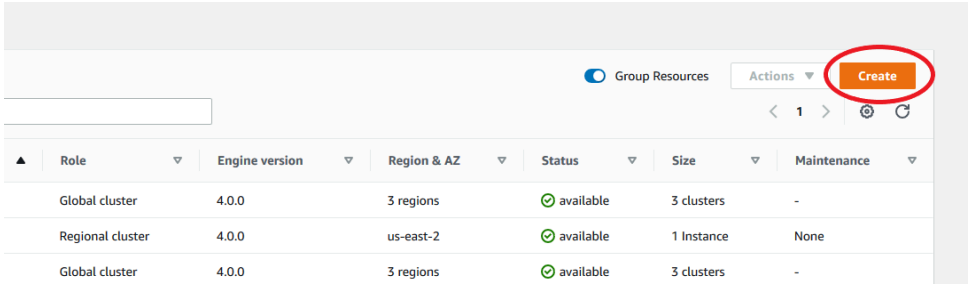
Apakah Anda siap untuk membangun klaster global pertama Anda? Pada bagian ini kami akan menjelaskan cara membuat klaster global baru dengan cluster dan contoh database baru, menggunakan Konsol Manajemen AWS atau AWS CLI dengan instruksi berikut.

Menggunakan Konsol Manajemen AWS

1. Di Konsol Manajemen AWS, navigasikan ke Amazon DocumentDB.
2. Ketika Anda sampai ke konsol Amazon DocumentDB, pilih Klaster.

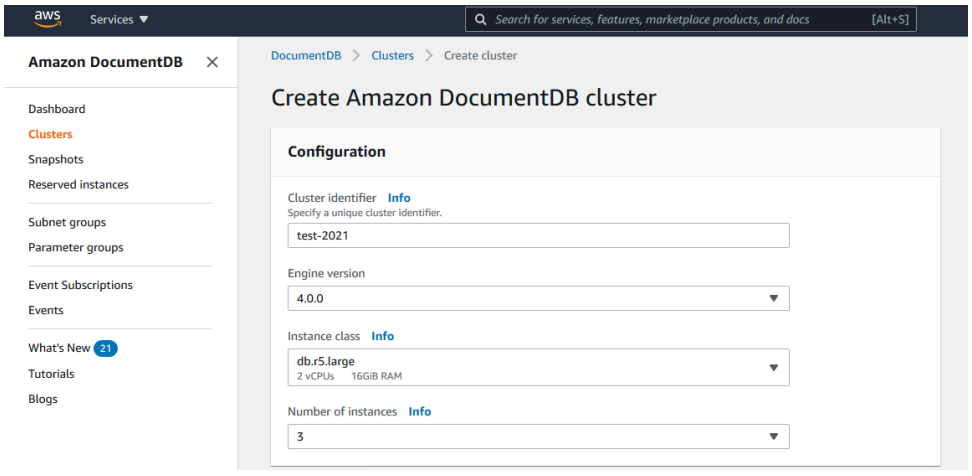


3. Pilih Buat.

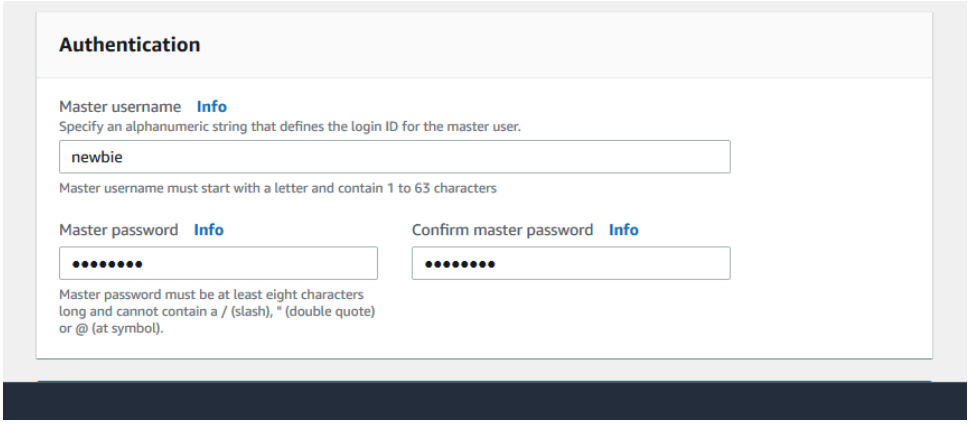


4. Mengisi bagian Konfigurasi dari formulir Buat Klaster Amazon DocumentDB dengan sesuai:

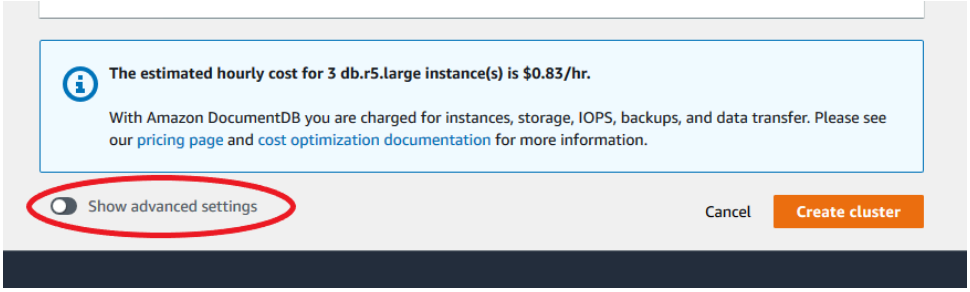
   - Pengidentifikasi klaster: Anda dapat memasukkan pengidentifikasi unik untuk instans ini atau mengizinkan Amazon DocumentDB menyediakan pengidentifikasi instans berdasarkan pengidentifikasi klaster.
   - Versi mesin: Pilih 4.0.0
   - Kelas instans: Pilih db.r5.large
   - Jumlah instans: Pilih 3.



5. Di Autentikasi, isi nama pengguna utama dan kata sandi utama.

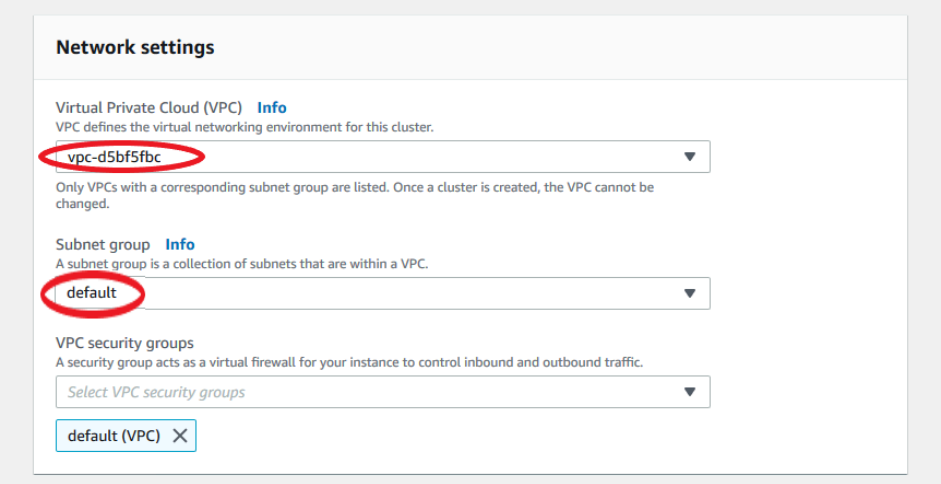


6. Pilih Tampilkan pengaturan lanjutan.

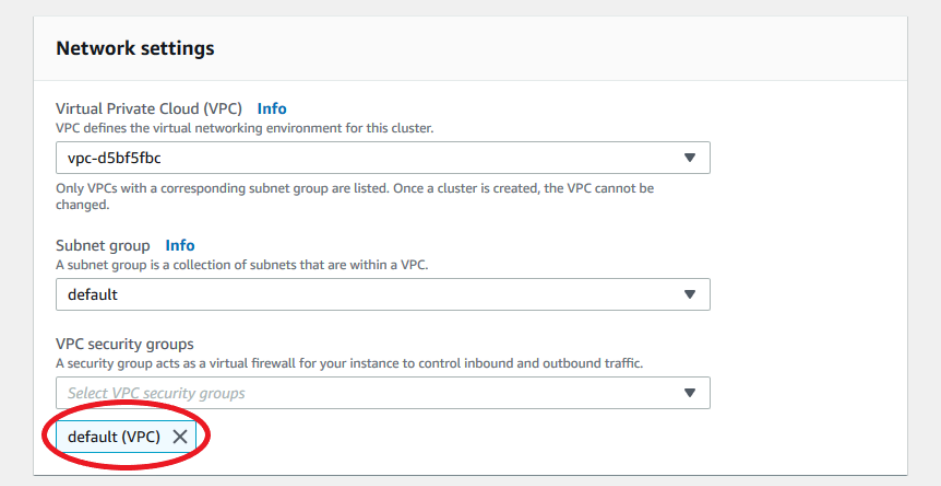


7. Di bagian Pengaturan jaringan:

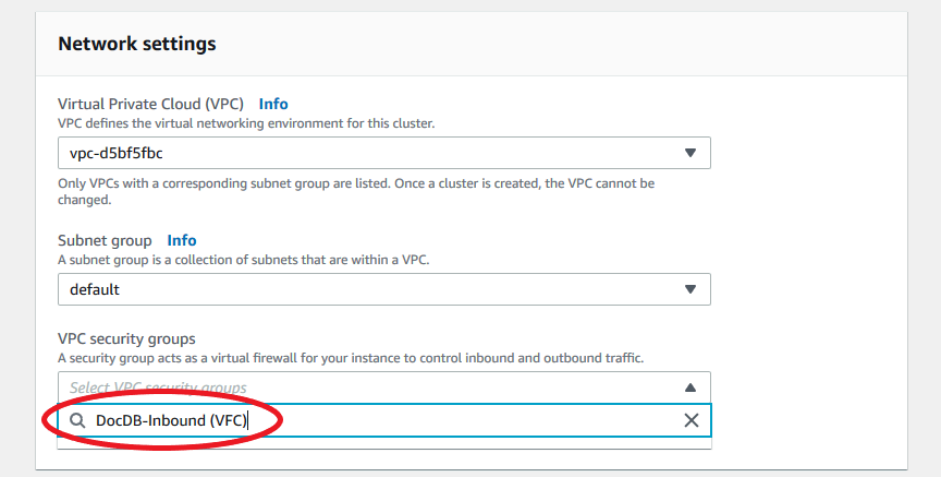
   - Simpan opsi default untuk grup Virtual Private Cloud (VPC) dan Subnet.



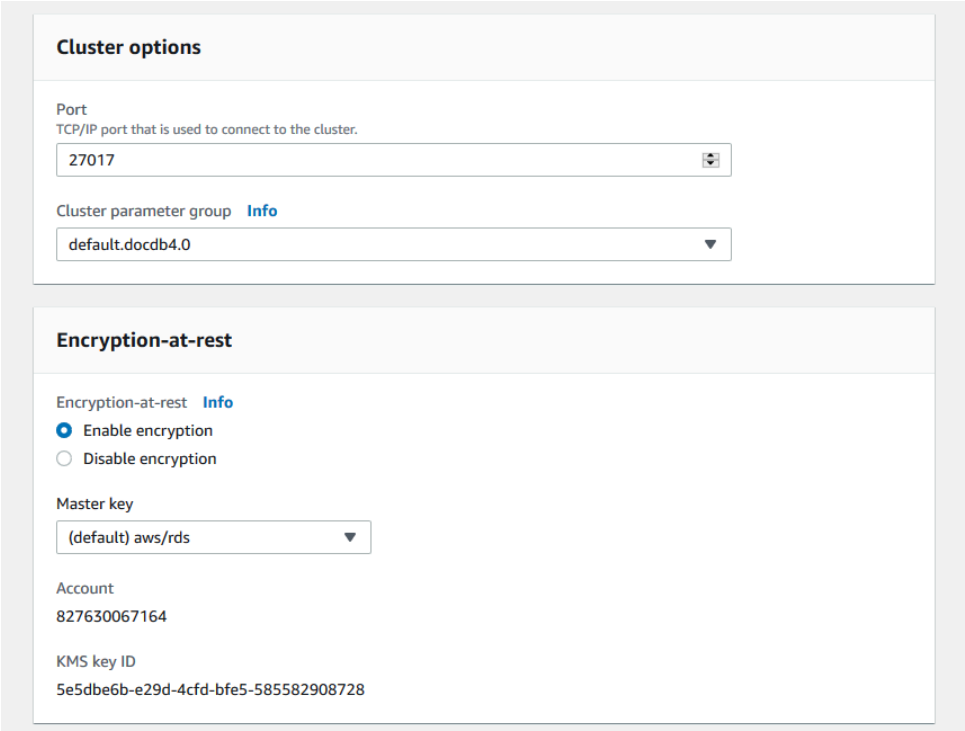
   - Untuk grup keamanan VPC, default (VPC) seharusnya sudah ditambahkan.



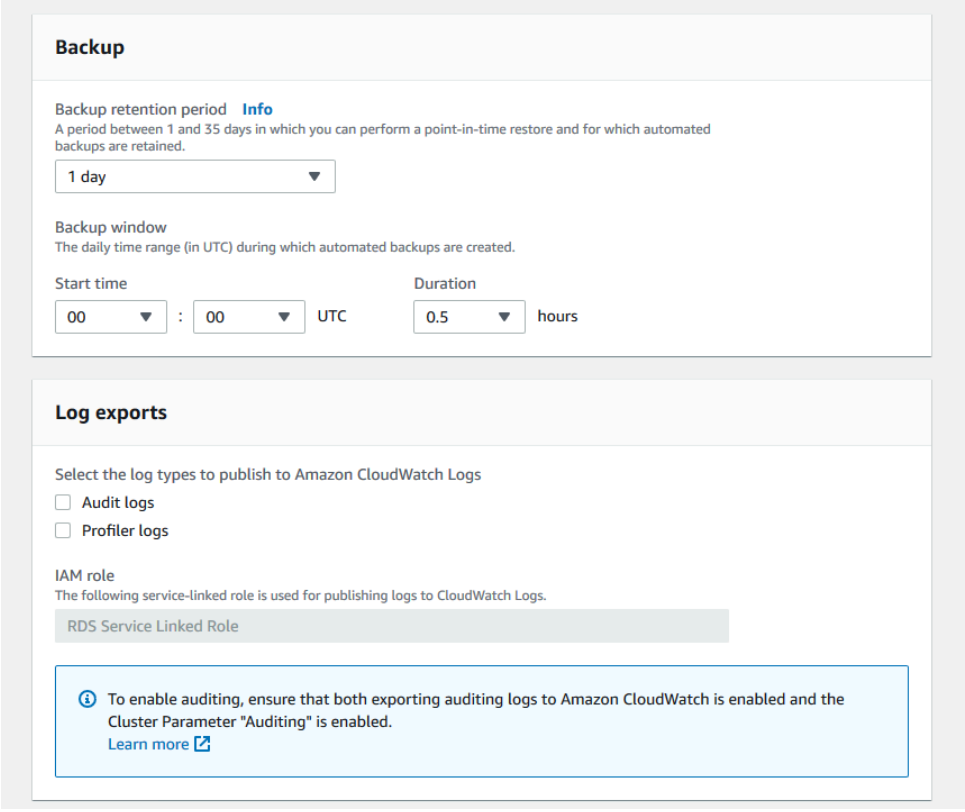
- Ketik DocDB ke dalam bidang grup keamanan VPC dan pilih DocDB-Inbound (VFC).



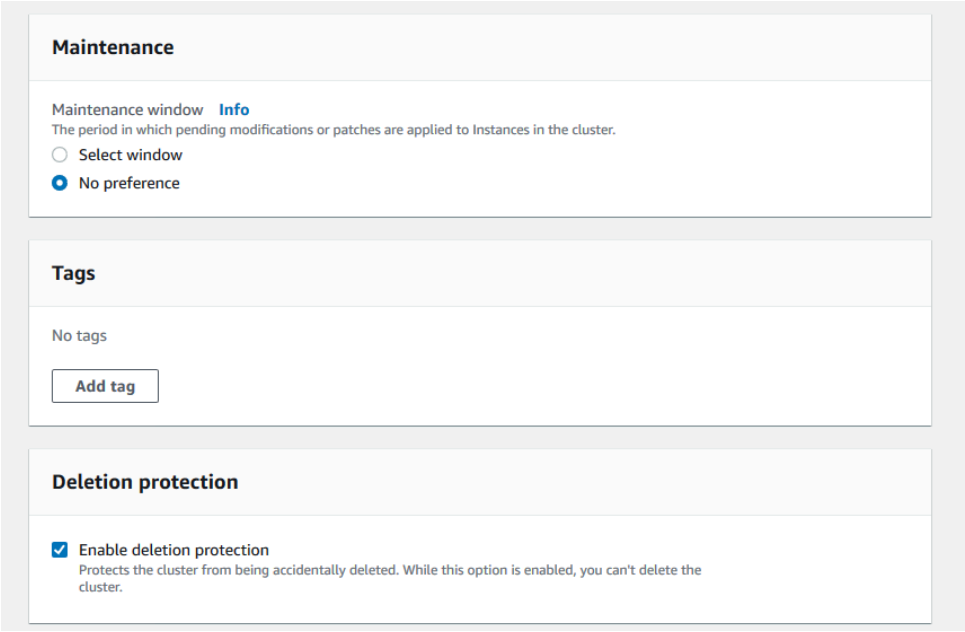
8. Untuk opsi Cluster dan Encryption-at-rest, biarkan pada pilihan default.



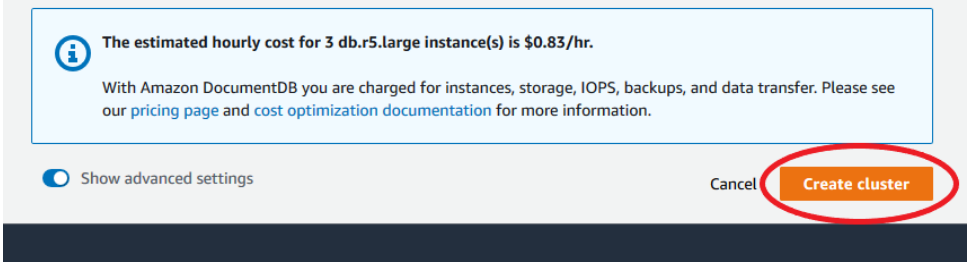
9. Untuk ekspor Backup dan Log, biarkan pada pilihan default.



10. Untuk perlindungan Pemeliharaan, Tag, dan Penghapusan, tinggalkan pilihan default.



11. Sekarang pilih Buat cluster.



Menggunakan AWS CLI

Untuk membuat cluster Regional Amazon DocumentDB, hubungi create-global-cluster. AWS CLI AWS CLI Perintah berikut membuat cluster Amazon DocumentDB bernama. `global-cluster-id` Untuk informasi selengkapnya tentang perlindungan penghapusan, lihat Menghapus cluster Amazon DocumentDB.

Juga, `--engine-version` adalah parameter opsional yang default ke versi mesin utama terbaru. Versi mesin default saat ini adalah `5.0.0` (catatan: Amazon DocumentDB 8.0 tersedia tetapi harus ditentukan secara eksplisit sebagai). `8.0.0` Ketika versi mesin utama baru dirilis, versi mesin default untuk `--engine-version` akan diperbarui untuk mencerminkan versi mesin utama terakhir. Akibatnya, untuk beban kerja produksi, dan terutama yang bergantung pada skrip, otomatisasi, atau CloudFormation templat, kami menyarankan Anda secara eksplisit menentukan `--engine-version` ke versi utama yang dimaksud.

Jika `vpc-security-group-id` ditentukan `db-subnet-group-name` atau tidak, Amazon DocumentDB akan menggunakan grup subnet default dan grup keamanan Amazon VPC untuk Wilayah tertentu.

Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi Anda sendiri.

Untuk Linux, macOS, atau 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
```

Untuk 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
```

Output dari operasi ini terlihat seperti berikut (format 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"
    }
}
```

Dibutuhkan beberapa menit untuk membuat klaster. Anda dapat menggunakan Konsol Manajemen AWS atau AWS CLI untuk memantau status cluster Anda. Untuk informasi selengkapnya, lihat Memantau status klaster Amazon DocumentDB.

> **Important**
>
> Saat Anda menggunakan AWS CLI untuk membuat klaster Regional Amazon DocumentDB, tidak ada instance yang dibuat. Akibatnya, Anda harus secara eksplisit membuat instans primer dan semua instans replika yang Anda butuhkan. Anda dapat menggunakan konsol atau AWS CLI untuk membuat instance. Untuk informasi selengkapnya, lihat Menambahkan instance Amazon DocumentDB ke klaster dan CreateDBCluster di Referensi API Amazon DocumentDB.

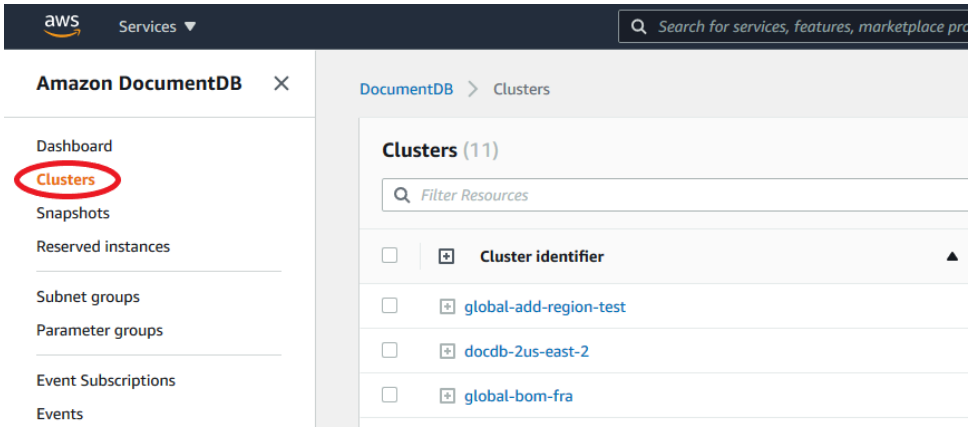
Setelah kluster Regional Anda tersedia, Anda dapat menambahkan klaster sekunder di Wilayah lain dengan petunjuk berikut: Menambahkan sebuah Wilayah AWS ke cluster global Amazon DocumentDB Saat Anda menambahkan Region, kluster Regional Anda menjadi klaster utama Anda, dan Anda memiliki klaster sekunder baru di Wilayah yang Anda pilih.

Menambahkan sebuah Wilayah AWS ke cluster global Amazon DocumentDB

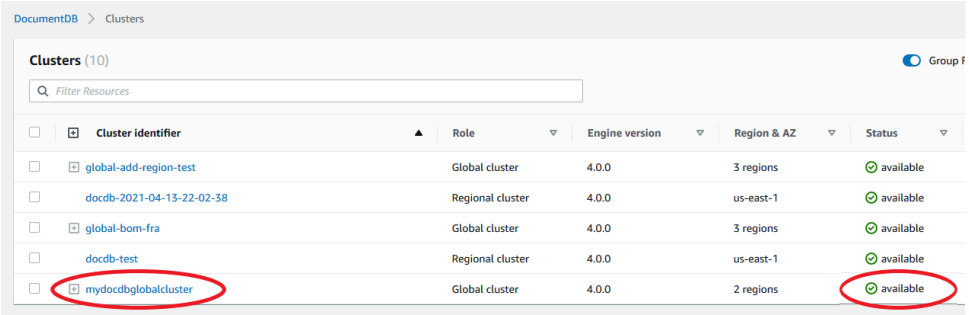
Sebuah cluster global membutuhkan setidaknya satu cluster sekunder di Region yang berbeda dari cluster primer, dan Anda dapat menambahkan hingga 10 cluster sekunder. Perhatikan bahwa untuk setiap klaster sekunder yang Anda tambahkan, Anda harus mengurangi jumlah replika yang diizinkan di klaster primer sebanyak satu. Misalnya, jika klaster global Anda memiliki 10 Wilayah sekunder, klaster utama Anda hanya dapat memiliki 5 (bukan 15) replika. Untuk informasi selengkapnya, lihat Persyaratan konfigurasi klaster global Amazon DocumentDB.

Menggunakan Konsol Manajemen AWS

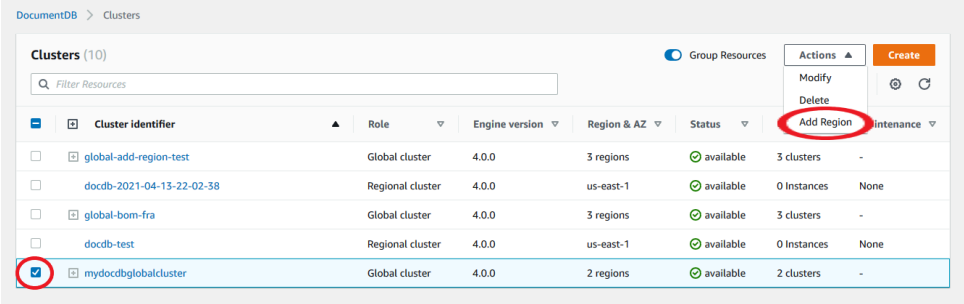
1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon DocumentDB.
2. Pada panel navigasi, silakan pilih Klaster.



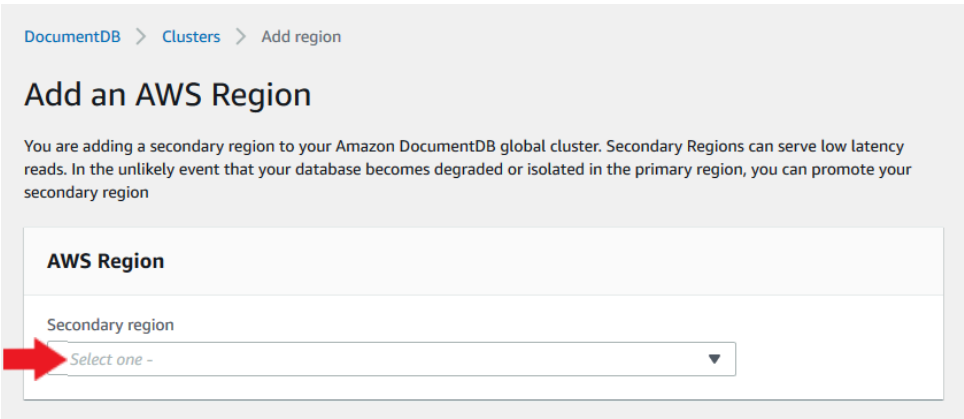
3. Pilih klaster yang ingin Anda tambahkan klaster sekundernya. Pastikan bahwa klaster adalah `Available`.



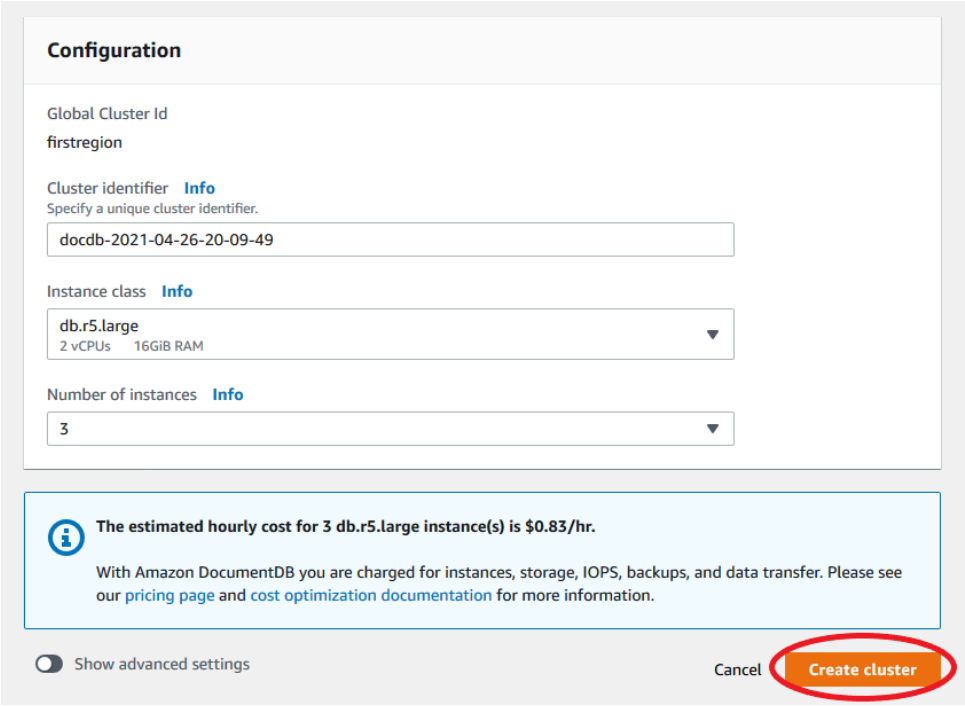
4. Pilih daftar dropdown untuk Tindakan dan kemudian pilih Tambah Wilayah.



5. Pada Wilayah AWS halaman Add an, pilih Region sekunder. Perhatikan bahwa Anda tidak dapat memilih Region yang sudah memiliki cluster sekunder untuk cluster global yang sama. Juga, itu tidak bisa menjadi Wilayah yang sama dengan cluster utama. Jika ini adalah Wilayah pertama yang Anda tambahkan, Anda juga harus menentukan pengidentifikasi cluster global pilihan Anda.



6. Lengkapi bidang yang tersisa untuk cluster sekunder di Wilayah baru, lalu pilih Buat cluster. Setelah Anda selesai menambahkan Wilayah, Anda dapat melihatnya di daftar Cluster di Konsol Manajemen AWS.



Menggunakan AWS CLI

- Gunakan perintah CLI `create-db-cluster` dengan nama (`--global-cluster-identifier`) dari klaster global Anda. Untuk Parameter lainnya, lakukan hal berikut:
  - Untuk`--region`, pilih yang berbeda Wilayah AWS dari Wilayah utama Anda.
  - Pilih nilai spesifik untuk parameter `--engine` dan `--engine-version`.
  - Untuk cluster terenkripsi, tentukan primer Anda Wilayah AWS sebagai enkripsi `--source-region` untuk.

Contoh berikut membuat klaster Amazon DocumentDB baru dan melampirkannya ke klaster global sebagai klaster sekunder baca-saja. Pada langkah terakhir, instans ditambahkan ke klaster baru.

Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi Anda sendiri.

Untuk Linux, macOS, atau 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
```

Untuk 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
```

Menggunakan snapshot untuk klaster global Amazon DocumentDB Anda

Anda dapat memulihkan snapshot klaster Amazon DocumentDB untuk digunakan sebagai titik awal klaster global Anda. Untuk melakukan ini, Anda harus memulihkan snapshot dan membuat klaster baru. Ini akan berfungsi sebagai klaster primer klaster global Anda. Anda kemudian dapat menambahkan Wilayah lain ke cluster yang dipulihkan, sehingga mengubahnya menjadi cluster global.

Mengelola klaster global Amazon DocumentDB

Anda melakukan sebagian besar operasi manajemen pada klaster individu yang membentuk klaster global. Saat Anda memilih Sumber daya terkait grup pada halaman Klaster di konsol tersebut, Anda melihat klaster primer dan klaster sekunder dikelompokkan di bawah klaster global terkait.

Tab Konfigurasi untuk klaster global menunjukkan Wilayah AWS tempat cluster berjalan, versi, dan pengidentifikasi klaster global.

Topik

- Memodifikasi klaster global Amazon DocumentDB
- Memodifikasi parameter klaster global Amazon DocumentDB
- Menghapus klaster dari klaster global Amazon DocumentDB
- Menghapus klaster dari klaster global Amazon DocumentDB
- Membuat klaster Amazon DocumentDB tanpa kepala di wilayah sekunder

Memodifikasi klaster global Amazon DocumentDB

Halaman Cluster dalam Konsol Manajemen AWS daftar semua cluster global Anda, menampilkan cluster primer dan cluster sekunder untuk masing-masing cluster. Klaster global memiliki penganturan konfigurasi sendiri. Secara khusus, ia memiliki wilayah yang terkait dengan cluster primer dan sekundernya.

Saat Anda membuat perubahan pada klaster global, Anda memiliki kesempatan untuk membatalkan perubahan.

Saat Anda memilih Lanjutkan, Anda mengonfirmasi perubahan.

Memodifikasi parameter klaster global Amazon DocumentDB

Anda dapat mengonfigurasi grup parameter klaster secara independen untuk setiap klaster dalam klaster global. Sebagian besar parameter berfungsi sama seperti untuk jenis klaster Amazon DocumentDB lainnya. Kami menyarankan Anda menjaga pengaturan yang konsisten di antara semua klaster dalam basis data global. Tindakan tersebut membantu mencegah perubahan perilaku yang tidak terduga jika Anda menaikkan klaster sekunder menjadi klaster primer.

Misalnya, gunakan pengaturan zona waktu dan rangkaian karakter yang sama untuk mencegah perilaku yang tidak konsisten jika klaster yang berbeda mengambil alih peran sebagai klaster primer.

Menghapus klaster dari klaster global Amazon DocumentDB

Ada beberapa situasi ketika Anda mungkin ingin menghapus klaster dari klaster global Anda. Misalnya, Anda mungkin ingin menghapus klaster dari klaster global jika klaster primer menjadi terdegradasi atau terisolasi. Ini kemudian menjadi klaster yang disediakan mandiri yang dapat

digunakan untuk membuat klaster global baru. Untuk mempelajari selengkapnya, lihat Melakukan failover manual untuk klaster global Amazon DocumentDB.

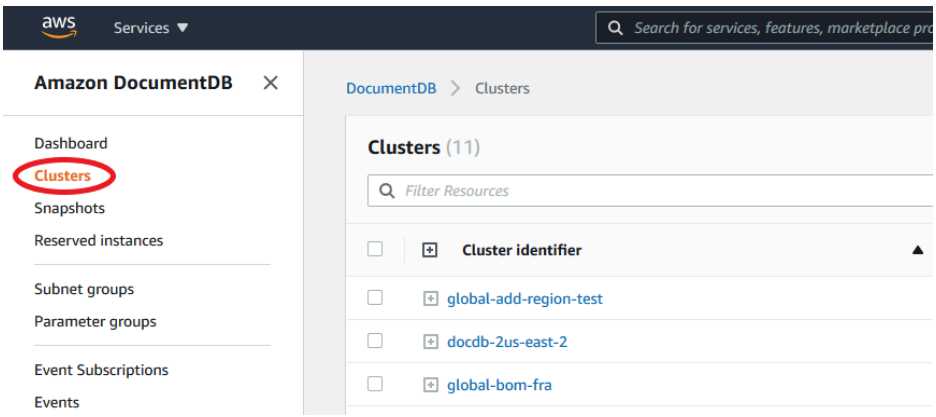
Anda juga mungkin ingin menghapus klaster karena Anda ingin menghapus klaster global yang tidak lagi Anda perlukan. Anda tidak dapat menghapus klaster global sampai setelah Anda melepaskan semua kluster terkait, meninggalkan yang primer untuk terakhir. Untuk informasi selengkapnya, lihat Menghapus klaster dari klaster global Amazon DocumentDB.

> **Note**
>
> Ketika sebuah cluster terlepas dari cluster global, itu tidak lagi disinkronkan dengan yang utama. Ini menjadi cluster yang disediakan mandiri dengan kemampuan penuh. read/write Selain itu, tidak lagi terlihat di konsol Amazon DocumentDB. Itu hanya terlihat ketika Anda memilih wilayah di konsol tempat cluster berada.

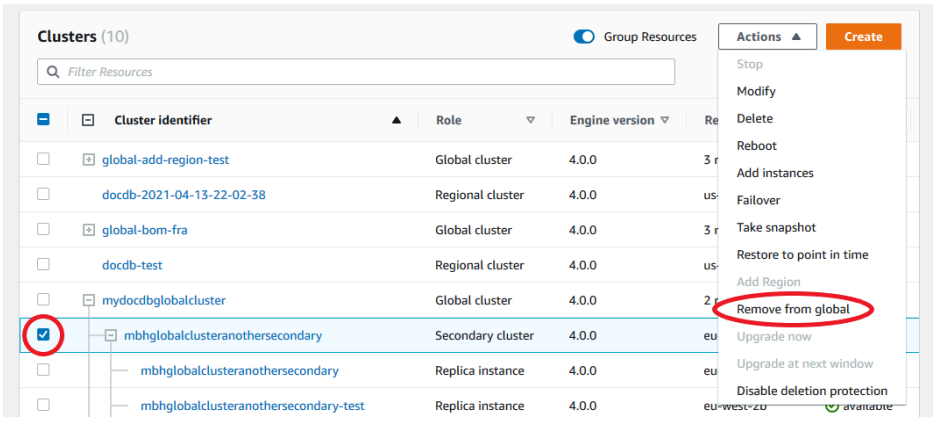
Anda dapat menghapus cluster dari cluster global Anda menggunakan Konsol Manajemen AWS, the AWS CLI, atau RDS API.

Using the Konsol Manajemen AWS

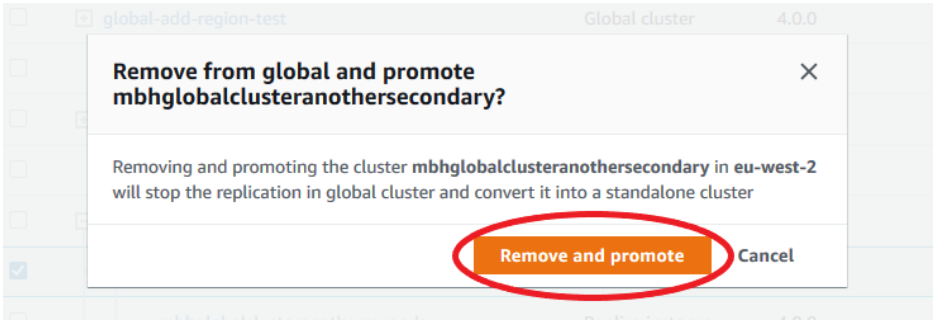
1. Masuk ke Konsol Manajemen AWS dan navigasikan ke konsol Amazon DocumentDB.
2. Pilih Klaster di navigasi sisi kiri.



3. Perluas klaster global sehingga Anda dapat melihat semua klaster sekunder. Pilih klaster sekunder yang ingin Anda hapus. Pilih Tindakan, dan di menu yang turun, pilih Hapus dari Global.



4. Prompt akan muncul, meminta Anda untuk mengonfirmasi bahwa Anda ingin melepaskan sekunder dari klaster global. Pilih Hapus dan promosikan untuk menghapus klaster dari klaster global.



Sekarang klaster itu tidak lagi berfungsi sebagai sekunder dan tidak lagi disinkronkan dengan klaster primer. Ini adalah cluster mandiri dengan read/write kemampuan penuh.

Setelah Anda menghapus atau menghapus semua klaster sekunder, maka Anda dapat menghapus klaster primer dengan cara yang sama. Anda tidak dapat melepaskan atau menghapus klaster primer dari klaster global sampai setelah Anda menghapus semua klaster sekunder. Klaster global mungkin tetap berada di daftar Klaster, dengan nol region dan AZ. Anda dapat menghapus jika tidak ingin lagi menggunakan klaster global ini.

Using the AWS CLI

Untuk menghapus klaster dari klaster global, jalankan perintah `remove-from-global-cluster` CLI dengan parameter berikut:

- `--global-cluster-identifier` — Nama (pengenal) klaster global Anda.
- `--db-cluster-identifier` — Nama setiap klaster untuk menghapus dari klaster global.

Contoh berikut pertama-tama menghapus klaster sekunder dan kemudian klaster primer dari klaster global.

Untuk Linux, macOS, atau 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
```

Ulangi perintah `remove-from-global-cluster --db-cluster-identifier secondary_cluster_ARN` untuk setiap wilayah sekunder di klaster global Anda.

Untuk 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
```

Ulangi perintah `remove-from-global-cluster --db-cluster-identifier secondary_cluster_ARN` untuk setiap wilayah sekunder di klaster global Anda.

Menghapus klaster dari klaster global Amazon DocumentDB

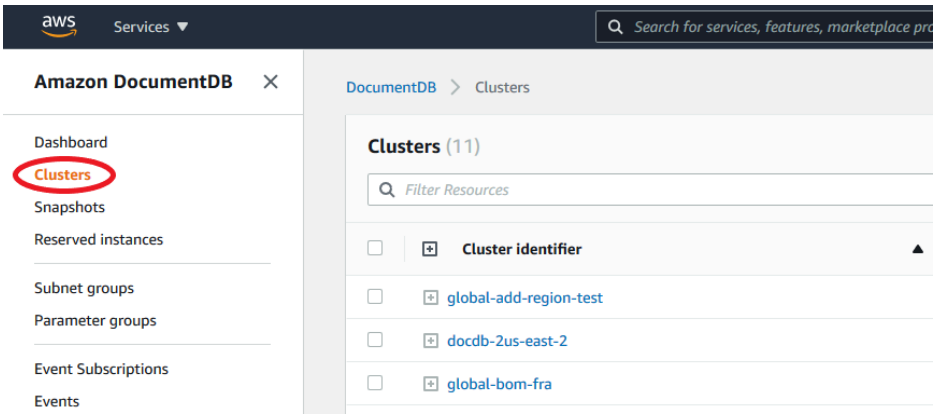
Untuk menghapus klaster global, lakukan hal berikut:

- Hapus semua klaster sekunder dari klaster global. Setiap klaster menjadi klaster yang berdiri sendiri. Lihat bagian sebelumnya,Menghapus klaster dari klaster global Amazon DocumentDB.
- Dari setiap klaster mandiri, hapus semua replika.

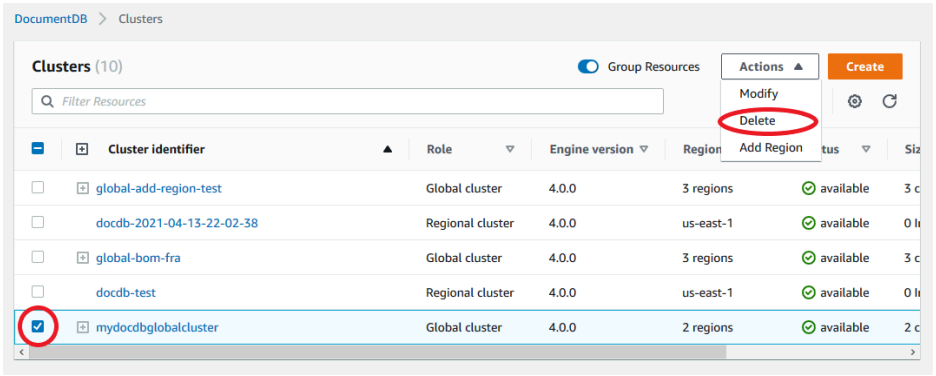
- Hapus klaster primer dari klaster global. Ini menjadi klaster yang berdiri sendiri.
- Dari klaster primer, hapus semua replika terlebih dahulu, lalu hapus instans primer. Menghapus instans primer dari klaster mandiri yang baru juga biasanya menghapus klaster dan klaster global.

Using the Konsol Manajemen AWS

1. Masuk ke Konsol Manajemen AWS dan navigasikan ke konsol Amazon DocumentDB.
2. Pilih Klaster dan temukan klaster global yang ingin Anda hapus.



3. Dengan klaster global Anda dipilih, pilih Hapus dari menu Tindakan.



Konfirmasikan bahwa semua klaster dihapus dari klaster global. Klaster global harus menunjukkan nol region dan AZ dan ukuran nol klaster. Jika klaster global berisi klaster apa pun, Anda belum dapat menghapusnya. Pertama-tama Anda harus mengikuti instruksi pada langkah sebelumnya Menghapus klaster dari klaster global Amazon DocumentDB.

Using the AWS CLI

Untuk menghapus cluster global, jalankan perintah `delete-global-cluster` CLI dengan nama Wilayah AWS dan pengidentifikasi cluster global, seperti yang ditunjukkan pada contoh berikut.

Untuk Linux, macOS, atau Unix:

```
aws docdb --region primary_region delete-global-cluster \
   --global-cluster-identifier global_cluster_id
```

Untuk Windows:

```
aws docdb --region primary_region delete-global-cluster ^
   --global-cluster-identifier global_cluster_id
```

Membuat klaster Amazon DocumentDB tanpa kepala di wilayah sekunder

Meskipun klaster global Amazon DocumentDB memerlukan setidaknya satu klaster sekunder yang Wilayah AWS berbeda dari yang utama, Anda dapat menggunakan konfigurasi tanpa kepala untuk klaster sekunder. Cluster Amazon DocumentDB sekunder tanpa kepala adalah cluster tanpa instance. Jenis konfigurasi ini dapat menurunkan biaya untuk cluster global. Dalam cluster Amazon DocumentDB, komputasi dan penyimpanan dipisahkan. Tanpa instance, Anda tidak dikenakan biaya untuk komputasi, hanya untuk penyimpanan. Jika diatur dengan benar, volume penyimpanan sekunder tanpa kepala tetap sinkron dengan cluster utama.

Anda menambahkan cluster sekunder seperti yang biasa Anda lakukan saat membuat klaster global Amazon DocumentDB. Namun, setelah cluster primer memulai replikasi ke sekunder, Anda menghapus instance hanya-baca dari cluster sekunder. Cluster sekunder ini sekarang dianggap “tanpa kepala” karena tidak lagi memiliki Instance. Namun, volume penyimpanan tetap sinkron dengan cluster Amazon DocumentDB utama.

> **Important**
>
> Kami hanya merekomendasikan klaster tanpa kepala untuk pelanggan yang dapat mentolerir kegagalan di seluruh wilayah selama 15+ menit. Ini karena pemulihan dari kegagalan di seluruh wilayah dengan cluster sekunder tanpa kepala akan mengharuskan pengguna untuk membuat instance baru setelah gagal. Sebuah instance baru dapat memakan waktu ~ 10-15 menit untuk tersedia.

Cara Menambahkan Cluster Sekunder Tanpa Kepala ke Cluster Global Anda

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon DocumentDB.

2. Pilih Klaster di navigasi sisi kiri.
3. Pilih cluster global yang membutuhkan cluster sekunder. Pastikan bahwa cluster utama adalah`Available`.
4. Untuk Tindakan, pilih Tambahkan region.
5. Pada halaman Tambah wilayah, pilih wilayah sekunder.

> **Note**
>
> Anda tidak dapat memilih wilayah yang sudah memiliki klaster sekunder untuk cluster global yang sama. Selain itu, wilayah tersebut tidak boleh sama dengan klaster primer.

6. Lengkapi bidang yang tersisa untuk cluster sekunder di wilayah baru. Ini adalah opsi konfigurasi yang sama seperti untuk instance cluster apa pun.
7. Tambahkan wilayah. Setelah Anda selesai menambahkan wilayah ke cluster global Anda, Anda akan melihatnya dalam daftar `Clusters` di Konsol Manajemen AWS.
8. Periksa status cluster sekunder dan instance pembacanya sebelum melanjutkan, dengan menggunakan Konsol Manajemen AWS atau AWS CLI. Berikut adalah contoh perintah jika Anda menggunakan AWS CLI:

```
$ aws docdb describe-db-clusters --db-cluster-identifier secondary-cluster-id --
query '*[].[Status]' --output text
```

   Diperlukan beberapa menit agar status cluster sekunder yang baru ditambahkan berubah dari pembuatan menjadi tersedia. Ketika cluster tersedia, Anda dapat menghapus instance pembaca.
9. Pilih instance pembaca di cluster sekunder, lalu pilih Hapus.
10. Setelah menghapus instance pembaca, cluster sekunder tetap menjadi bagian dari cluster global. Seharusnya tidak ada contoh yang terkait dengannya.

> **Note**
>
> Anda dapat menggunakan klaster Amazon DocumentDB sekunder tanpa kepala ini untuk memulihkan klaster global Amazon DocumentDB Anda secara manual dari pemadaman yang tidak direncanakan di wilayah utama jika terjadi pemadaman seperti itu.

Connect ke klaster global Amazon DocumentDB

Cara Anda terhubung ke klaster global bergantung pada apakah Anda perlu menulis ke klaster atau membaca dari klaster:

- Untuk permintaan atau kueri baca-saja, Anda menyambungkan ke reader endpoint untuk klaster di Wilayah AWS.
- Untuk menjalankan pernyataan bahasa manipulasi data (DML) atau bahasa definisi data (DDL), Anda terhubung ke titik akhir klaster untuk klaster primer. Endpoint ini mungkin berbeda Wilayah AWS dari aplikasi Anda.

Saat Anda melihat klaster global di konsol, Anda dapat melihat semua titik akhir tujuan umum yang terkait dengan semua klasternya.

Bagaimana Anda terhubung ke klaster global bergantung pada apakah Anda perlu menulis ke basis data atau membaca dari basis data. Untuk DDL, DML, dan operasi baca yang ingin Anda layani dari wilayah primer, Anda harus terhubung ke klaster primer Anda. Kami menyarankan Anda menyambungkan ke klaster primer menggunakan titik akhir klaster dalam mode set replika, dengan preferensi baca `secondaryPreferred=true`. Ini akan merutekan lalu lintas tulis ke instans penulis klaster primer Anda dan lalu lintas baca ke instans replika klaster primer Anda.

Untuk lintas wilayah, baca saja lalu lintas, Anda harus terhubung ke salah satu cluster sekunder Anda. Kami menyarankan Anda terhubung ke cluster sekunder Anda menggunakan titik akhir cluster dalam mode set replika. Karena semua instance adalah instance replika hanya-baca, Anda harus menentukan preferensi baca selain `primary` (misalnya,, `secondary` atau). `secondaryPreferred` `nearest` Untuk meminimalkan latensi, pilih reader endpoint mana saja yang ada di wilayah Anda atau wilayah terdekat dengan Anda.

Memantau klaster global Amazon DocumentDB

Amazon DocumentDB (dengan kompatibilitas MongoDB) terintegrasi CloudWatch dengan sehingga Anda dapat mengumpulkan dan menganalisis metrik operasional untuk cluster Anda. Anda dapat memantau metrik ini menggunakan CloudWatch konsol, konsol Amazon DocumentDB, () AWS Command Line Interface ,AWS CLI atau API. CloudWatch

Untuk memantau klaster global, gunakan CloudWatch metrik berikut.

Metrik	Deskripsi
`GlobalClusterReplicatedWriteIO`	Jumlah rata-rata I/O operasi penulisan yang ditagih direplikasi dari volume cluster di volume primer Wilayah AWS ke volume cluster di sekunder Wilayah AWS, dilaporkan pada interval 5 menit. Banyaknya `ReplicatedWriteIOs` yang direplikasi untuk setiap wilayah sekunder sama dengan jumlah `VolumeWriteIOPs` dalam wilayah yang dilakukan oleh wilayah primer.
`GlobalClusterDataTransferBytes`	Jumlah data yang ditransfer dari cluster primer Wilayah AWS ke cluster sekunder Wilayah AWS, diukur dalam byte.
`GlobalClusterReplicationLag`	Jumlah lag, dalam milidetik, saat mereplikasi peristiwa perubahan dari cluster primer Wilayah AWS ke cluster sekunder Wilayah AWS

Untuk informasi selengkapnya tentang cara melihat metrik ini, lihat Melihat CloudWatch data.

Pemulihan bencana dan klaster global Amazon DocumentDB

Topik

- Melakukan failover terkelola untuk klaster global Amazon DocumentDB
- Melakukan failover manual untuk klaster global Amazon DocumentDB
- Melakukan peralihan untuk klaster global Amazon DocumentDB
- Membuka blokir peralihan atau failover klaster global

Dengan menggunakan klaster global, Anda dapat pulih dari bencana seperti kegagalan wilayah dengan cepat. Pemulihan dari bencana biasanya diukur dengan menggunakan nilai untuk RTO dan RPO.

- Tujuan waktu pemulihan (RTO) — Waktu yang dibutuhkan sistem untuk kembali ke kondisi kerja setelah bencana. Dengan kata lain, RTO mengukur waktu henti. Untuk cluster global, RTO dalam hitungan menit.
- Tujuan titik pemulihan (RPO) — Jumlah data yang dapat hilang (diukur dalam waktu). Untuk klaster global, RPO biasanya diukur dalam hitungan detik.
- Untuk memulihkan dari pemadaman yang tidak direncanakan, Anda dapat melakukan failover lintas wilayah ke salah satu sekunder di klaster global Anda. Jika klaster global Anda memiliki beberapa wilayah sekunder, pastikan Anda melepaskan semua wilayah sekunder yang ingin Anda promosikan sebagai primer. Kemudian, Anda mempromosikan salah satu wilayah sekunder tersebut menjadi Wilayah AWS primer yang baru. Terakhir, Anda membuat klaster baru di setiap wilayah sekunder lainnya dan melampirkan klaster tersebut ke klaster global Anda.

Melakukan failover terkelola untuk klaster global Amazon DocumentDB

Pendekatan ini ditujukan untuk kelangsungan bisnis saat terjadi bencana alam Regional yang riil atau pemadaman tingkat layanan secara menyeluruh.

Selama failover terkelola, klaster utama Anda gagal ke Region sekunder pilihan Anda sementara topologi replikasi klaster global Amazon DocumentDB yang ada dipertahankan. Klaster sekunder yang dipilih mempromosikan salah satu simpul hanya-bacanya ke status penulis penuh. Langkah ini memungkinkan klaster untuk mengambil peran sebagai klaster primer. Basis data Anda tidak akan tersedia untuk sementara saat klaster ini mengambil peran barunya. Data yang tidak direplikasi dari primer lama ke cluster sekunder yang dipilih mungkin hilang saat sekunder ini menjadi primer baru. Volume primer lama melakukan upaya terbaik untuk mengambil snapshot sebelum menyinkronkan dengan primer baru sehingga data yang tidak direplikasi dipertahankan pada snapshot.

> **Note**
>
> Anda hanya dapat melakukan failover klaster lintas wilayah terkelola di klaster global Amazon DocumentDB jika klaster primer dan semua klaster sekunder memiliki versi engine yang sama. Jika versi mesin Anda tidak kompatibel, Anda dapat melakukan failover secara manual dengan mengikuti langkah-langkah dalam Melakukan failover manual untuk klaster global Amazon DocumentDB.
> Jika versi mesin wilayah tidak cocok, failover akan diblokir. Periksa peningkatan yang tertunda dan terapkan untuk memastikan semua versi mesin wilayah cocok dan failover

cluster global tidak diblokir. Untuk informasi selengkapnya, lihat Membuka blokir peralihan atau failover klaster global.

Untuk meminimalkan kehilangan data, sebaiknya lakukan hal berikut sebelum menggunakan fitur ini:

- Bawa aplikasi offline untuk mencegah penulisan dikirim ke cluster utama klaster global Amazon DocumentDB.
- Periksa waktu jeda untuk semua cluster sekunder Amazon DocumentDB. Memilih Wilayah sekunder dengan keterlambatan replikasi minimum dapat meminimalkan kehilangan data dari Wilayah primer yang mengalami kegagalan. Periksa waktu jeda untuk semua cluster sekunder Amazon DocumentDB di cluster global dengan melihat metrik di `GlobalClusterReplicationLag` Amazon. CloudWatch Metrik ini menunjukkan seberapa jauh di belakang (dalam milidetik) replikasi ke cluster sekunder ke cluster utama.

  Untuk informasi selengkapnya tentang CloudWatch metrik untuk Amazon Metrik Amazon DocumentDB DocumentDB, lihat.

Selama failover terkelola, cluster sekunder yang dipilih dipromosikan ke peran barunya sebagai primer. Namun, itu tidak mewarisi berbagai opsi konfigurasi cluster utama. Ketidakcocokan dalam konfigurasi dapat menyebabkan masalah performa, inkompatibilitas beban kerja, dan perilaku anomali lainnya. Untuk menghindari masalah seperti itu, kami sarankan Anda menyelesaikan perbedaan antara kluster global Amazon DocumentDB Anda untuk hal-hal berikut:

- Konfigurasikan grup parameter klaster Amazon DocumentDB untuk primer baru, jika perlu — Anda dapat mengonfigurasi grup parameter klaster Amazon DocumentDB secara independen untuk setiap cluster di cluster global Amazon DocumentDB Anda. Oleh karena itu, ketika Anda mempromosikan klaster sekunder untuk mengambil alih peran utama, grup parameter dari sekunder mungkin dikonfigurasi secara berbeda dari yang utama. Jika demikian, ubah grup parameter klaster sekunder yang dipromosikan agar sesuai dengan pengaturan klaster utama Anda. Untuk mempelajari caranya, lihat Memodifikasi grup parameter cluster Amazon DocumentDB.
- Konfigurasikan alat dan opsi pemantauan, seperti CloudWatch peristiwa dan alarm Amazon — Konfigurasikan klaster yang dipromosikan dengan kemampuan logging, alarm, dan sebagainya yang sama sesuai kebutuhan untuk klaster global. Seperti grup parameter, konfigurasi untuk fitur ini tidak diwariskan dari klaster primer selama proses failover berlangsung. Beberapa CloudWatch metrik, seperti replikasi lag, hanya tersedia untuk Wilayah sekunder. Karena itu, failover akan

mengubah cara Anda melihat metrik tersebut dan mengatur alarmnya, serta mengharuskan adanya perubahan pada dasbor yang ditentukan sebelumnya. Untuk informasi selengkapnya tentang klaster dan pemantauan Amazon DocumentDB, lihat. Memantau Amazon DocumentDB

Biasanya, cluster sekunder yang dipilih mengasumsikan peran utama dalam satu menit. Segera setelah simpul penulis Wilayah primer baru tersedia, Anda dapat menghubungkan aplikasi Anda ke simpul tersebut dan melanjutkan beban kerja Anda. Setelah Amazon DocumentDB mempromosikan cluster primer baru, secara otomatis membangun kembali semua klaster Region sekunder tambahan.

Karena klaster global Amazon DocumentDB menggunakan replikasi asinkron, kelambatan replikasi di setiap Wilayah sekunder dapat bervariasi. Amazon DocumentDB membangun kembali Wilayah sekunder ini untuk memiliki data point-in-time yang sama persis dengan cluster Region utama yang baru. Durasi penyelesaian tugas pembangunan ulang dapat memerlukan waktu beberapa menit hingga beberapa jam, bergantung pada ukuran volume penyimpanan dan jarak di antara Wilayah. Saat klaster Wilayah sekunder selesai dibuat ulang dari Wilayah primer yang baru, klaster ini menjadi tersedia untuk akses baca. Segera setelah penulis utama baru dipromosikan dan tersedia, klaster Region primer yang baru dapat menangani operasi baca dan tulis untuk cluster global Amazon DocumentDB.

Untuk mengembalikan topologi asli cluster global, Amazon DocumentDB memantau ketersediaan Wilayah primer lama. Segera setelah Wilayah itu sehat dan tersedia kembali, Amazon DocumentDB secara otomatis menambahkannya kembali ke cluster global sebagai Wilayah sekunder. Sebelum membuat volume penyimpanan baru di Wilayah primer lama, Amazon DocumentDB mencoba mengambil snapshot dari volume penyimpanan lama pada titik kegagalan. Hal ini dilakukan agar Anda dapat menggunakannya untuk memulihkan setiap data yang hilang. Jika operasi ini berhasil, Amazon DocumentDB menempatkan snapshot ini bernama “rds:docdb-unplanned-global-failover-name-of-old-primary-DB-cluster-timestamp” di bagian snapshot dari file. Konsol Manajemen AWS Anda juga dapat melihat snapshot ini tercantum dalam informasi yang dikembalikan oleh operasi `DescribeDBClusterSnapshots` API.

> **Note**
>
> Snapshot dari volume penyimpanan lama adalah snapshot sistem yang tunduk pada periode retensi pencadangan yang dikonfigurasi pada klaster primer yang lama. Untuk mempertahankan snapshot ini di luar periode retensi, Anda dapat menyalin snapshot untuk disimpan sebagai snapshot manual. Untuk mempelajari selengkapnya tentang cara menyalin snapshot, termasuk harga, lihat Menyalin snapshot cluster.

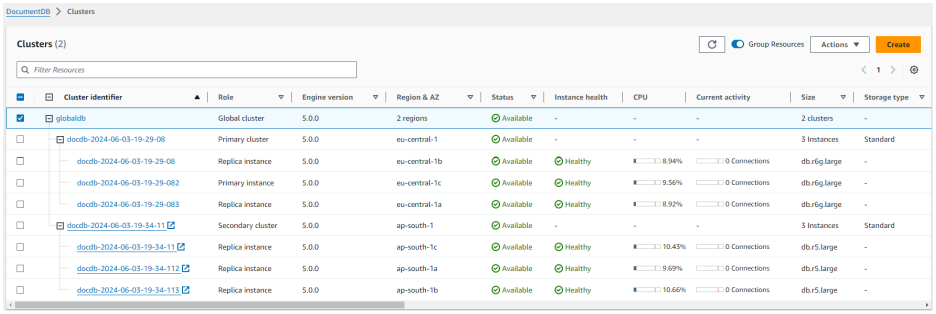
Setelah topologi asli dipulihkan, Anda dapat gagal mengembalikan klaster global Anda ke Wilayah utama asli dengan melakukan operasi peralihan ketika itu paling masuk akal untuk bisnis dan beban kerja Anda. Untuk melakukannya, ikuti langkah yang ada di Melakukan peralihan untuk klaster global Amazon DocumentDB.

Anda dapat gagal melalui kluster global Amazon DocumentDB menggunakan Konsol Manajemen AWS API Amazon DocumentDB, AWS CLI atau Amazon DocumentDB.

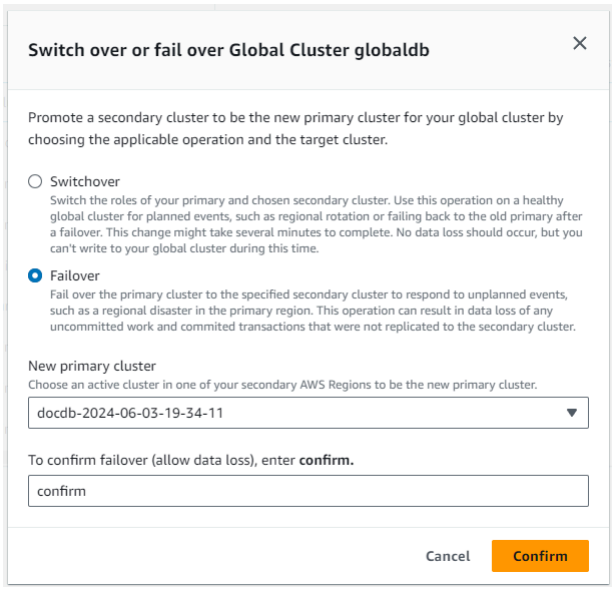
Using the Konsol Manajemen AWS

Lakukan failover terkelola di klaster global Amazon DocumentDB

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Pada panel navigasi, silakan pilih Klaster.
3. Temukan dan pilih klaster global Amazon DocumentDB yang ingin Anda gagal.



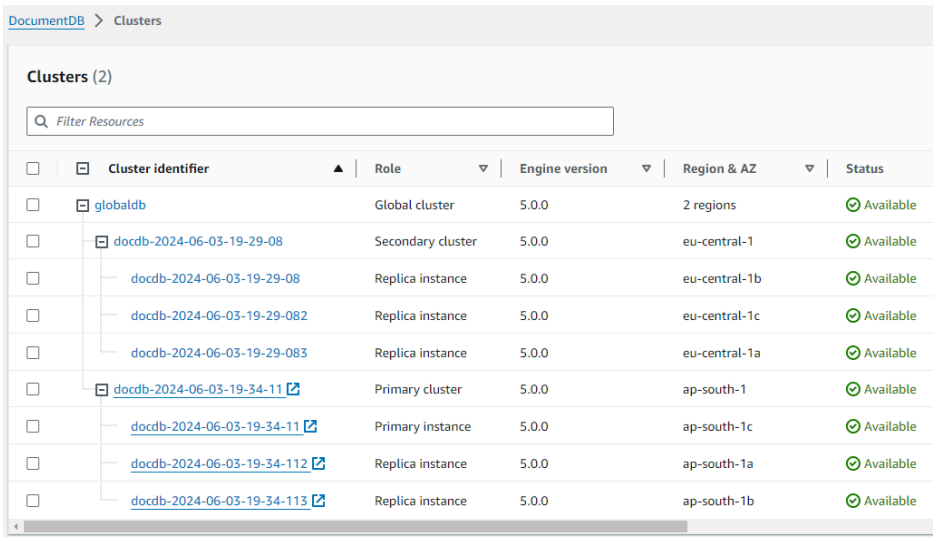
4. Pilih Switchover atau Failover dari menu Tindakan.
5. Pada kotak dialog yang muncul, pilih Failover, lalu pilih cluster sekunder dari Daftar drop-down bidang cluster primer baru.



6. Ketik “konfirmasi” di bidang terakhir. Kemudian pilih Konfirmasi.

   Status klaster utama berubah menjadi "Failing-over”. Kondisi ini akan memakan waktu sekitar satu menit. Selama waktu ini, status cluster primer baru menunjukkan "Memodifikasi... “. Setelah primer baru dipromosikan, itu akan menampilkan "Tersedia" dan akan dapat melayani transaksi baca dan tulis. Wilayah sekunder termasuk primer lama akan menampilkan "Menyinkronkan ulang... “sementara itu menyinkronkan ulang ke primer baru. Mirip dengan primer baru, itu hanya akan dapat melayani transaksi setelah status berubah menjadi "Tersedia”.

7. Ketika selesai, cluster primer asli menjadi cluster sekunder. Cluster sekunder yang dipilih menjadi cluster utama.



Using the AWS CLI

Lakukan failover terkelola di klaster global Amazon DocumentDB

Jalankan perintah `failover-global-cluster`CLI agar gagal melalui cluster global Amazon DocumentDB Anda. Dengan perintah, berikan nilai untuk opsi berikut:

- `--region`
- `--global-cluster-identifier`
- `--target-db-cluster-identifier`
- `--allow-data-loss`

Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi cluster Anda.

Untuk Linux, macOS, atau 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
```

Untuk 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
```

Melakukan failover manual untuk klaster global Amazon DocumentDB

Jika seluruh cluster dalam satu Wilayah AWS menjadi tidak tersedia, Anda dapat mempromosikan klaster lain di cluster global untuk memiliki read/write kemampuan.

Anda dapat mengaktifkan mekanisme failover cluster global secara manual jika cluster di cluster yang berbeda Wilayah AWS adalah pilihan yang lebih baik untuk menjadi cluster utama. Misalnya,

Anda dapat meningkatkan kapasitas salah satu klaster sekunder dan kemudian mempromosikannya menjadi klaster primer. Atau keseimbangan aktivitas di antara Wilayah AWS mungkin berubah, sehingga mengalihkan cluster utama ke cluster lain Wilayah AWS mungkin memberikan latensi yang lebih rendah untuk operasi penulisan.

Prosedur berikut menguraikan apa yang harus dilakukan untuk mempromosikan salah satu cluster sekunder di cluster global Amazon DocumentDB.

Untuk mempromosikan cluster sekunder:

1. Berhenti mengeluarkan pernyataan DHTML dan operasi penulisan lainnya ke cluster utama di Wilayah AWS with the outage.
2. Identifikasi cluster dari sekunder Wilayah AWS untuk digunakan sebagai cluster primer baru. Jika Anda memiliki dua (atau lebih) sekunder Wilayah AWS di cluster global Anda, pilih cluster sekunder yang memiliki waktu jeda paling sedikit.
3. Lepaskan klaster sekunder yang Anda pilih dari klaster global.

   Menghapus cluster sekunder dari cluster global segera menghentikan replikasi dari primer ke sekunder ini dan mempromosikannya ke cluster yang disediakan mandiri dengan kemampuan penuh. read/write Klaster sekunder lainnya yang terkait dengan klaster primer di wilayah dengan pemadaman masih tersedia dan dapat menerima panggilan dari aplikasi Anda. Klaster tersebut juga mengonsumsi sumber daya. Karena Anda membuat ulang klaster global, untuk menghindari otak-terbelah dan masalah lainnya, hapus klaster sekunder lainnya sebelum membuat klaster global baru dalam langkah-langkah berikut.

   Untuk langkah-langkah rinci untuk melepaskan, lihat Menghapus klaster dari klaster global Amazon DocumentDB.
4. Klaster ini menjadi klaster primer dari klaster global baru saat Anda mulai menambahkan Wilayah ke dalamnya, pada langkah berikutnya.
5. Tambahkan Wilayah AWS ke cluster. Saat Anda melakukannya, proses replikasi dari klaster primer ke klaster sekunder akan dimulai.
6. Tambahkan lebih banyak Wilayah AWS sesuai kebutuhan untuk membuat ulang topologi yang diperlukan untuk mendukung aplikasi Anda. Pastikan bahwa penulisan aplikasi dikirim ke klaster yang benar sebelum, selama, dan setelah membuat perubahan seperti ini, untuk menghindari inkonsistensi data di antara klaster dalam klaster global (masalah split-brain).
7. Saat pemadaman teratasi dan Anda siap menugaskan Wilayah AWS asli sebagai klaster primer lagi, lakukan langkah yang sama secara terbalik.

8. Hapus salah satu klaster sekunder dari klaster global. Ini akan memungkinkannya untuk melayani read/write lalu lintas.
9. Alihkan ulang semua lalu lintas tulis ke klaster primer di Wilayah AWS asli.
10. Tambahkan Wilayah AWS untuk mengatur satu atau lebih cluster sekunder yang sama Wilayah AWS seperti sebelumnya.

Cluster global Amazon DocumentDB dapat dikelola AWS menggunakan SDK, memungkinkan Anda membuat solusi untuk mengotomatiskan proses failover klaster global untuk kasus penggunaan Disaster Recovery dan Business Continuity Planning. Salah satu solusi tersebut tersedia untuk pelanggan kami di bawah lisensi Apache 2.0 dan dapat diakses dari repositori alat kami di sini. Solusi ini memanfaatkan Amazon Route 53 untuk manajemen titik akhir dan menyediakan AWS Lambda fungsi yang dapat dipicu berdasarkan peristiwa yang sesuai.

Melakukan peralihan untuk klaster global Amazon DocumentDB

Dengan menggunakan switchover, Anda dapat mengubah Wilayah cluster utama Anda secara rutin. Pendekatan ini ditujukan untuk skenario yang terkontrol, seperti pemeliharaan operasional dan prosedur operasional terencana lainnya.

Ada tiga kasus penggunaan umum untuk menggunakan switchover:

- Untuk persyaratan "rotasi regional" yang diberlakukan pada industri tertentu. Misalnya, peraturan layanan keuangan mungkin menginginkan sistem tier-0 untuk beralih ke Wilayah yang berbeda selama beberapa bulan untuk memastikan prosedur pemulihan bencana dilaksanakan secara teratur.
- Untuk aplikasi "follow-the-sun" multi-Wilayah. Misalnya, suatu bisnis mungkin ingin menyediakan penulisan dengan latensi lebih rendah di berbagai Wilayah berdasarkan jam kerja di zona waktu yang berbeda.
- Sebagai metode zero-data-loss untuk melakukan failback ke Wilayah primer asli setelah failover.

> **Note**
>
> Switchovers dirancang untuk digunakan pada cluster global Amazon DocumentDB yang sehat. Untuk pulih dari pemadaman yang tak terduga, ikuti prosedur yang sesuai di Melakukan failover manual untuk klaster global Amazon DocumentDB.
> Untuk melakukan switchover, semua wilayah sekunder harus menjalankan versi mesin yang sama persis dengan yang utama. Jika versi mesin wilayah tidak cocok, peralihan

akan diblokir. Periksa peningkatan yang tertunda dan terapkan untuk memastikan semua versi mesin wilayah cocok dan peralihan cluster global tidak diblokir. Untuk informasi selengkapnya, lihat Membuka blokir peralihan atau failover klaster global.

Selama peralihan, Amazon DocumentDB mengalihkan klaster utama Anda ke Wilayah sekunder pilihan Anda sambil mempertahankan topologi replikasi klaster global yang ada. Sebelum memulai proses peralihan, Amazon DocumentDB menunggu semua cluster Region sekunder disinkronkan sepenuhnya dengan kluster Region primer. Selanjutnya, klaster DB di Wilayah primer menjadi klaster hanya-baca, dan klaster sekunder yang dipilih akan mempromosikan salah satu simpul hanya-bacanya menjadi status penulis penuh. Mempromosikan simpul ini menjadi penulis memungkinkan klaster sekunder mengambil peran klaster primer. Karena semua cluster sekunder disinkronkan dengan primer pada awal proses, primer baru melanjutkan operasi untuk cluster global Amazon DocumentDB tanpa kehilangan data apa pun. Basis data Anda tidak tersedia untuk sementara selama klaster primer dan klaster sekunder yang dipilih mengambil peran barunya masing-masing.

Untuk mengoptimalkan ketersediaan aplikasi, sebaiknya lakukan hal berikut sebelum menggunakan fitur ini:

- Lakukan operasi ini selama jam nonpeak atau di lain waktu ketika menulis ke cluster primer minimal.
- Bawa aplikasi offline untuk mencegah penulisan dikirim ke cluster utama klaster global Amazon DocumentDB.
- Periksa waktu jeda untuk semua cluster sekunder Amazon DocumentDB di cluster global dengan melihat metrik di `GlobalClusterReplicationLag` Amazon. CloudWatch Metrik ini menunjukkan seberapa jauh di belakang (dalam milidetik) replikasi ke cluster sekunder ke cluster primer. Nilai ini berbanding lurus dengan waktu yang dibutuhkan Amazon DocumentDB untuk menyelesaikan peralihan. Karena itu, semakin besar nilai keterlambatan, semakin lama durasi switchover.

  Untuk informasi selengkapnya tentang CloudWatch metrik untuk Amazon Metrik Amazon DocumentDB DocumentDB, lihat.

Selama proses switchover, klaster DB sekunder yang dipilih akan dipromosikan ke peran barunya sebagai primer. Namun, klaster ini tidak mewarisi berbagai opsi konfigurasi klaster DB primer. Ketidakcocokan dalam konfigurasi dapat menyebabkan masalah performa, inkompatibilitas beban

kerja, dan perilaku anomali lainnya. Untuk menghindari masalah seperti itu, kami sarankan Anda menyelesaikan perbedaan antara kluster global Amazon DocumentDB Anda untuk hal-hal berikut:

- Konfigurasikan grup parameter cluster Amazon DocumentDB DB untuk primer baru, jika perlu — Anda dapat mengonfigurasi grup parameter klaster Amazon DocumentDB secara independen untuk setiap cluster di klaster global Amazon DocumentDB Anda. Hal ini berarti ketika Anda mempromosikan klaster DB sekunder untuk mengambil alih peran primer, grup parameter dari klaster sekunder mungkin memiliki konfigurasi yang berbeda dengan klaster primer. Jika demikian, ubah grup parameter klaster DB sekunder yang dipromosikan agar sesuai dengan pengaturan klaster primer Anda. Untuk mempelajari caranya, lihat Mengelola grup parameter cluster Amazon DocumentDB.
- Konfigurasikan alat dan opsi pemantauan, seperti Amazon CloudWatch Events dan alarm — Konfigurasikan klaster yang dipromosikan dengan kemampuan logging, alarm, dan sebagainya yang sama sesuai kebutuhan untuk klaster global. Seperti grup parameter, konfigurasi untuk fitur ini tidak diwariskan dari klaster primer selama proses switchover berlangsung. Beberapa CloudWatch metrik, seperti replikasi lag, hanya tersedia untuk Wilayah primer. Karena itu, switchover akan mengubah cara Anda melihat metrik tersebut dan mengatur alarmnya, serta mengharuskan adanya perubahan pada dasbor yang ditentukan sebelumnya. Untuk informasi selengkapnya, lihat Memantau Amazon DocumentDB.

> **Note**
>
> Biasanya, switchover peran dapat memerlukan waktu hingga beberapa menit.

Saat proses peralihan selesai, klaster Amazon DocumentDB yang dipromosikan dapat menangani operasi penulisan untuk klaster global.

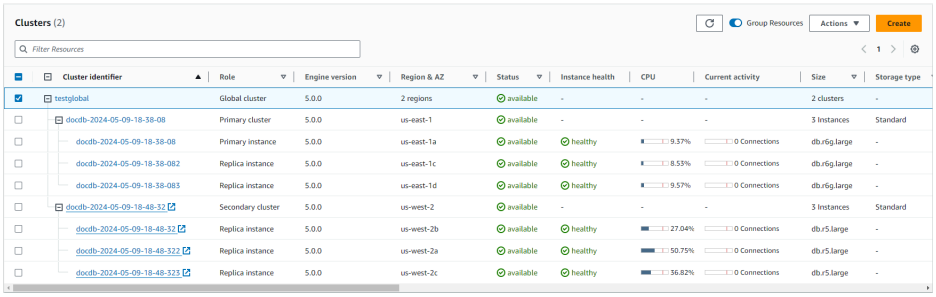
Anda dapat beralih ke klaster global Amazon DocumentDB menggunakan Konsol Manajemen AWS atau: AWS CLI

Using the Konsol Manajemen AWS

Lakukan peralihan di klaster global Amazon DocumentDB

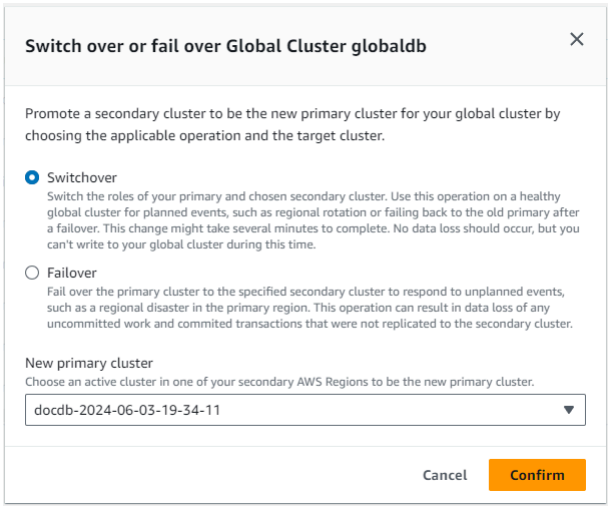
1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Pada panel navigasi, silakan pilih Klaster.

3. Temukan dan pilih klaster global Amazon DocumentDB yang ingin Anda alihkan.



4. Pilih Switchover atau Failover dari menu Tindakan.

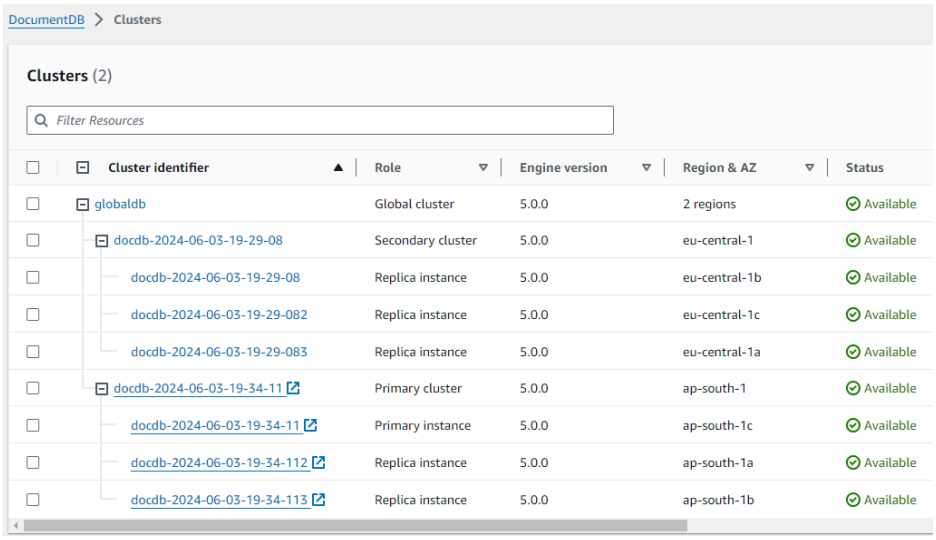
5. Pada kotak dialog yang muncul, pilih Switchover, lalu pilih cluster sekunder dari Daftar drop-down bidang cluster primer baru.



6. Pilih Konfirmasi.

   Status klaster utama berubah menjadi "Switching-over”. Kondisi ini akan memakan waktu sekitar tiga menit. Selama waktu ini, status semua cluster regional menunjukkan "Memodifikasi... “. Setelah wilayah disinkronkan dan primer baru dipromosikan, itu akan menampilkan "Tersedia" untuk semua bidang status dan akan dapat melayani transaksi.

7. Ketika selesai, cluster primer asli menjadi cluster sekunder. Cluster sekunder yang dipilih menjadi cluster utama.



Using the AWS CLI

Lakukan peralihan di klaster global Amazon DocumentDB

Jalankan perintah `switchover-global-cluster`CLI untuk beralih ke cluster global Amazon DocumentDB Anda. Dengan perintah, berikan nilai untuk opsi berikut:

- `--region`
- `--global-cluster-identifier`
- `--target-db-cluster-identifier`

Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi cluster Anda.

Untuk Linux, macOS, atau 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
```

Untuk 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
```

Membuka blokir peralihan atau failover klaster global

Switchover dan failover cluster global diblokir ketika tidak semua cluster regional di cluster global menggunakan versi mesin yang sama. Jika versi tidak cocok, Anda mungkin melihat kesalahan ini sebagai respons saat memanggil switchover atau failover: Cluster DB target yang ditentukan menjalankan versi engine dengan tingkat patch yang berbeda dari cluster DB sumber. Kami merekomendasikan secara rutin menerapkan versi mesin terbaru untuk memastikan Anda menjalankan pembaruan terbaru untuk menjaga klaster global Anda dalam keadaan sehat.

Untuk mengatasi kesalahan ini, perbarui semua wilayah sekunder terlebih dahulu, lalu wilayah utama ke versi engine yang sama dengan menerapkan item tindakan pemeliharaan yang tertunda. Untuk melihat item tindakan pemeliharaan yang tertunda, dan menerapkan perubahan yang diperlukan untuk memperbaiki masalah, lakukan instruksi di salah satu tab berikut:

Using the Konsol Manajemen AWS

Untuk membuka blokir peralihan atau failover klaster global, Anda harus menentukan apakah ada tindakan pemeliharaan yang tertunda untuk cluster Anda dan menerapkannya. Ikuti langkah-langkah ini untuk melihat dan menerapkan tindakan pemeliharaan:

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Pada panel navigasi, silakan pilih Klaster.
3. Dalam tabel Clusters, temukan klaster global Anda di kolom pengenal Cluster. Di bawah klaster global Anda, catat setiap cluster sekunder dan klaster utama untuk klaster global yang diberikan, dan lakukan langkah-langkah berikut untuk masing-masing klaster.
4. Untuk setiap cluster sekunder:

    a. Jika pembaruan tersedia untuk klaster Anda, itu ditunjukkan sebagai Tersedia, Diperlukan, atau Jendela Berikutnya di kolom Pemeliharaan.

    b. Untuk mengambil tindakan, pilih cluster untuk menampilkan detailnya, lalu pilih Maintenance & backup. Item Pemeliharaan Tertunda muncul.

    c. Di bawah Deskripsi, jika ini menunjukkan bahwa “Pembaruan pemeliharaan baru tersedia”, pilih dan kemudian pilih Terapkan sekarang.

5. Untuk klaster utama Anda:

   a. Jika pembaruan tersedia untuk klaster Anda, itu ditunjukkan sebagai Tersedia, Diperlukan, atau Jendela Berikutnya di kolom Pemeliharaan.

   b. Untuk mengambil tindakan, pilih cluster untuk menampilkan detailnya, lalu pilih Maintenance & backup. Item Pemeliharaan Tertunda muncul.

   c. Di bawah Deskripsi, jika ini menunjukkan bahwa “Pembaruan pemeliharaan baru tersedia”, pilih dan kemudian pilih Terapkan sekarang.

Using the AWS CLI

Untuk membuka blokir peralihan atau failover klaster global, Anda harus menentukan apakah ada tindakan pemeliharaan yang tertunda untuk klaster dan menerapkannya. Ikuti langkah-langkah berikut untuk melihat dan menerapkan tindakan pemeliharaan terlebih dahulu pada cluster sekunder kemudian pada klaster utama untuk klaster global Anda:

1. Jalankan yang berikut ini pada setiap cluster regional regional sekunder terlebih dahulu dan kemudian untuk klaster regional daerah primer.
2. Jalankan perintah `describe-pending-maintenance-actions`CLI dengan `--resource-identifier` opsi untuk menentukan apakah ada tindakan pemeliharaan yang tersedia untuk klaster regional Amazon DocumentDB Anda.

   Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi cluster Anda.

   Untuk Linux, macOS, atau Unix:

```
aws docdb describe-pending-maintenance-action \
   --resource-identifier arn:aws:rds:us-
east-1:001234567890:cluster:docdb-2025-03-27-19-21-15
```

   Untuk Windows:

```
aws docdb describe-pending-maintenance-action ^
   --resource-identifier arn:aws:rds:us-
east-1:001234567890:cluster:docdb-2025-03-27-19-21-15
```

   Hasilnya terlihat mirip dengan ini:

```
{
    "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. Jika tindakan pemeliharaan diperlukan, jalankan perintah [apply-pending-maintenance-action](#)CLI dengan opsi berikut:

   - `--resource-identifier`
   - `--apply-action`
   - `--opt-in-type`
   - `--region`

   Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi cluster Anda.

   Untuk Linux, macOS, atau 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
```

Untuk 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. Setelah tindakan pemeliharaan selesai, jalankan describe-pending-maintenance-actionsperintah lagi untuk memastikan bahwa tidak ada tindakan lain yang tertunda untuk cluster Anda.

   Hasil yang Anda inginkan adalah:

```
{
    "PendingMaintenanceActions": []
}
```

Using the Amazon DocumentDB API

Untuk membuka blokir peralihan atau failover klaster global, Anda harus menentukan apakah ada tindakan pemeliharaan yang tertunda untuk klaster dan menerapkannya. Gunakan API berikut untuk melihat dan menerapkan tindakan pemeliharaan:

1. Jalankan yang berikut ini pada setiap cluster regional regional sekunder terlebih dahulu dan kemudian untuk klaster regional daerah primer.
2. Panggil PendingMaintenanceActionAPI untuk menentukan apakah ada tindakan pemeliharaan yang tersedia untuk klaster global Amazon DocumentDB Anda.
3. Terapkan perubahan apa pun dengan memanggil ApplyPendingMaintenanceActionAPI.

Mengelola cluster Amazon DocumentDB

Untuk mengelola klaster Amazon DocumentDB, Anda harus memiliki kebijakan IAM dengan izin bidang kontrol Amazon DocumentDB yang sesuai. Izin ini memungkinkan Anda untuk membuat,

memodifikasi, dan menghapus klaster dan instans. Kebijakan `AmazonDocDBFullAccess` menyediakan semua izin yang diperlukan untuk mengelola klaster Amazon DocumentDB.

Topik berikut menunjukkan bagaimana caranya melakukan berbagai tugas ketika bekerja dengan klaster Amazon DocumentDB, termasuk menciptakan, menghapus, memodifikasi, menghubungkan ke, dan melihat klaster.

Topik

- Memahami cluster
- Pengaturan Amazon DocumentDB Cluster
- Konfigurasi penyimpanan cluster Amazon DocumentDB
- Menentukan status klaster
- Siklus hidup klaster Amazon DocumentDB
- Menskalakan cluster Amazon DocumentDB
- Mengkloning volume untuk cluster Amazon DocumentDB
- Memahami toleransi kesalahan klaster Amazon DocumentDB

Memahami cluster

Amazon DocumentDB memisahkan perhitungan dan penyimpanan, dan replikasi data offload dan backup ke volume klaster. Volume klaster menyediakan lapisan penyimpanan tahan lama, andal, dan sangat tersedia yang mereplikasi data enam cara di tiga Zona Ketersediaan. Replika memungkinkan ketersediaan data yang lebih tinggi dan membaca penskalaan. Setiap klaster dapat diskalakan ke atas hingga 15 replika.

Kata benda	Deskripsi	Operasi API (Kata Kerja)

Klaster	Terdiri atas satu atau beberapa instans dan volume penyimpanan klaster yang mengelola data untuk instans tersebut.	`create-db-cluster`
		`delete-db-cluster`
		`describe-db-clusters`
		`modify-db-cluster`

Instans	Membaca dan menulis data ke volume penyimpanan klaster	`create-db-instance`

Kata benda	Deskripsi	Operasi API (Kata Kerja)

	dilakukan melalui instans. Dalam sebuah klaster tertentu, ada dua jenis instans: primer dan replika. Sebuah klaster selalu memiliki satu instans utama dan dapat memiliki 0–15 replika.	`delete-db-instance`
		`describe-db-instances`
		`modify-db-instance`
		`describe-orderable-db-instance-options`
		`reboot-db-instance`

Volume klaster	Volume penyimpanan basis data virtual yang mencakup tiga Availability Zone, dengan setiap Availability Zone memiliki dua salinan data klaster.	N/A
Instans utama	Mendukung operasi baca dan tulis, dan melakukan semua modifikasi data pada volume klaster. Setiap klaster memiliki satu instans utama.	N/A
Instans replika	Mendukung hanya operasi baca. Setiap klaster Amazon DocumentDB dapat memiliki hingga 15 instans replika di samping instans utama. Beberapa replika mendistri busikan beban kerja baca. Dengan menemukan replika di Availability Zone terpisah, Anda juga dapat meningkat kan ketersediaan basis data.	N/A

Kata benda	Deskripsi	Operasi API (Kata Kerja)
Titik akhir klaster	Titik akhir untuk klaster Amazon DocumentDB yang menghubungkan ke instans utama saat ini untuk klaster. Setiap klaster Amazon DocumentDB memiliki titik akhir klaster dan satu instans utama.	N/A
Titik akhir pembaca	Titik akhir untuk klaster Amazon DocumentDB yang terhubung ke salah satu replika yang tersedia untuk klaster tersebut. Setiap klaster Amazon DocumentDB memiliki reader endpoint. Jika ada lebih dari satu replika, reader endpoint mengarahk an setiap permintaan koneksi ke salah satu replika Amazon DocumentDB.	N/A
Instans titik akhir	Titik akhir untuk instans dalam klaster Amazon DocumentD B yang terhubung ke instans tertentu. Setiap instans dalam sebuah klaster, apa pun jenis instansnya, memiliki titik akhir instans uniknya sendiri.	N/A

Pengaturan Amazon DocumentDB Cluster

Ketika Anda membuat atau memodifikasi klaster, penting untuk memahami parameter yang tetap dan yang dimodifikasi setelah klaster selesai dibuat. Tabel berikut mencantumkan semua pengaturan,

atau parameter, yang khusus dibuat untuk klaster. Seperti yang ditentukan dalam tabel, beberapa dapat dimodifikasi, yang lain tidak.

> **Note**
>
> Pengaturan ini jangan sampai tertukar dengan grup parameter klaster Amazon DocumentDB dan parameternya. Untuk informasi selengkapnya tentang grup parameter klaster, lihat Mengelola grup parameter cluster Amazon DocumentDB.

Parameter	Dapat diubah	Catatan
DBClusterIdentifier	Ya	Aturan penamaan: • Panjangnya adalah [1—63] huruf, angka, atau tanda hubung. • Karakter pertama harus berupa huruf. • Tidak dapat diakhiri dengan tanda hubung atau berisi dua tanda hubung berurutan. • Harus unik untuk semua cluster di Amazon RDS, Amazon Neptune, dan Amazon DocumentDB per, per Wilayah. Akun AWS
Engine	Tidak	Harus berupa `docdb`.
BackupRetentionPeriod	Ya	Harus antara [1-35] hari.
DBClusterParameterGroupName	Ya	Aturan penamaan: • Panjang adalah [1—255] karakter alfanumerik. • Karakter pertama harus berupa huruf. • Tidak dapat diakhiri dengan tanda hubung atau mengandung dua tanda hubung berturut-turut.

Parameter	Dapat diubah	Catatan
DBSubnetGroupName	Tidak	Setelah klaster selesai dibuat, Anda tidak dapat mengubah subnet klaster.
EngineVersion	Tidak	Nilai bisa `5.0.0` (default),`4.0.0`, atau`3.6.0`.
KmsKeyId	Tidak	Jika Anda memilih untuk mengenkripsi klaster Anda, Anda tidak dapat mengubah AWS KMS kunci yang Anda gunakan untuk mengenkripsi klaster Anda.
MasterUsername	Tidak	Setelah sebuah klaster telah dibuat, Anda tidak dapat mengubah `MasterUsername` . Aturan penamaan: • Panjangnya adalah [1—63] karakter alfanumerik. • Karakter pertamanya harus berupa huruf. • Tidak bisa menjadi kata yang disimpan oleh mesin basis data.

MasterUserPassword	Ya	Aturan:
		• Panjang adalah [8—100] karakter ASCII yang dapat dicetak.
		• Dapat menggunakan karakter ASCII yang dapat dicetak kecuali untuk berikut ini:
		  • **/** (garis miring ke depan)
		  • **"** (tanda kutip ganda)
		  • **@** (simbol pada)

Port	Ya	Nomor port berlaku untuk semua instans dalam klaster.
PreferredBackupWindow	Ya	

Parameter	Dapat diubah	Catatan
PreferredMaintenanceWindow	Ya	
StorageEncrypted	Tidak	Jika Anda memilih untuk mengenkripsi klaster Anda, klaster tersebut tidak dapat terenkripsi.
StorageType	Ya	Jenis penyimpanan untuk cluster DB: Standard (`standard`) atau I/O-Optimized (`iopt1`). Default: `standard` Parameter ini dapat dikonfigurasi dengan `CreateDBCluster` dan`ModifyDBCluster` . Untuk informasi selengkapnya, lihat Konfigurasi penyimpanan cluster Amazon DocumentDB.
Tags	Ya	
VpcSecurityGroupIds	Tidak	Setelah klaster telah dibuat, Anda tidak dapat mengubah VPC tempat klaster berada.

Konfigurasi penyimpanan cluster Amazon DocumentDB

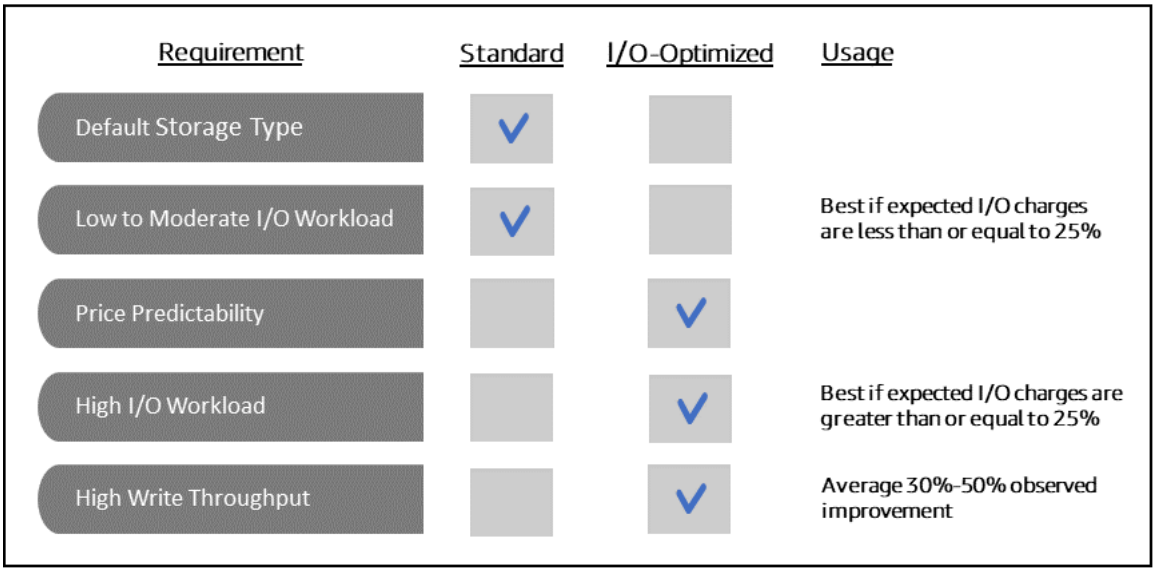
Mulai dari Amazon DocumentDB 5.0, cluster berbasis instance mendukung dua jenis konfigurasi penyimpanan:

- Penyimpanan standar Amazon DocumentDB: Dirancang untuk pelanggan dengan konsumsi rendah hingga sedang. I/O Jika Anda mengharapkan I/O biaya Anda kurang dari 25% dari total cluster Amazon DocumentDB Anda, pilihan ini mungkin ideal untuk Anda. Dengan konfigurasi penyimpanan standar Amazon DocumentDB, Anda ditagih berdasarkan I/O bayar per permintaan selain biaya instans dan penyimpanan. Ini berarti tagihan Anda mungkin bervariasi dari satu siklus ke siklus lainnya berdasarkan penggunaan. Konfigurasi disesuaikan untuk mengakomodasi I/O permintaan aplikasi Anda yang berfluktuasi.

- Penyimpanan Amazon I/O-optimized DocumentDB: Dirancang untuk pelanggan yang memprioritaskan prediktabilitas harga atau memiliki aplikasi intensif. I/O I/O-optimized Konfigurasi ini menawarkan peningkatan kinerja, peningkatan throughput, dan pengurangan latensi bagi pelanggan dengan beban kerja I/O intensif. Jika Anda mengharapkan I/O biaya Anda melebihi 25% dari total biaya cluster Amazon DocumentDB Anda, opsi ini menawarkan peningkatan kinerja harga. Dengan konfigurasi penyimpanan Amazon I/O-optimized DocumentDB, Anda tidak akan dikenakan biaya I/O berdasarkan operasi, memastikan biaya yang dapat diprediksi setiap siklus penagihan. Konfigurasi menstabilkan biaya sekaligus meningkatkan kinerja.

Anda dapat mengalihkan cluster database yang ada setiap 30 hari sekali ke penyimpanan Amazon DocumentDB I/O-optimized . Anda dapat beralih kembali ke penyimpanan standar Amazon DocumentDB kapan saja. Tanggal berikutnya untuk memodifikasi konfigurasi penyimpanan untuk I/O-optimized dapat dilacak dengan `describe-db-clusters` perintah menggunakan AWS CLI atau melalui Konsol Manajemen AWS di halaman konfigurasi cluster.

Anda dapat membuat kluster database baru termasuk konfigurasi Amazon I/O-optimized DocumentDB atau mengonversi kluster database yang ada dengan beberapa klik di, perubahan parameter tunggal di Konsol Manajemen AWSAWS CLI(), atau melalui SDK.AWS Command Line InterfaceAWS Tidak ada downtime atau reboot instance yang diperlukan selama atau setelah memodifikasi konfigurasi penyimpanan.

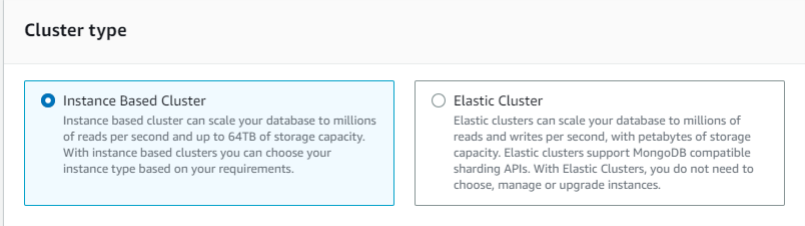


Membuat I/O-optimized cluster

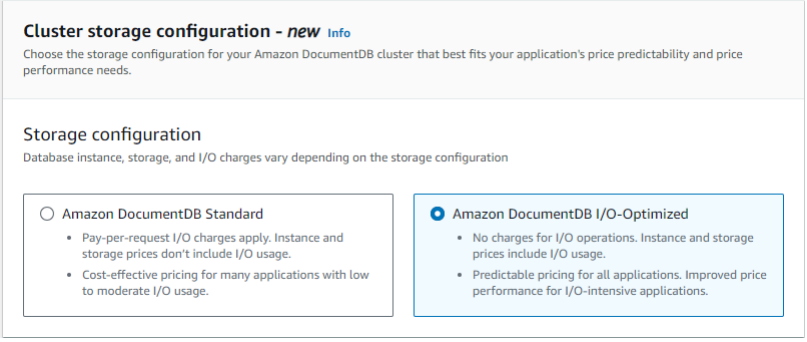
Using the Konsol Manajemen AWS

Untuk membuat atau memodifikasi I/O-optimized cluster menggunakan Konsol Manajemen AWS:

1. Di konsol manajemen Amazon DocumentDB, di bawah Clusters, pilih Create atau pilih cluster dan pilih Actions, lalu pilih Modify.
2. Jika Anda membuat klaster baru, pastikan Anda memilih Instance Based Cluster di bagian tipe Cluster (ini adalah opsi default).



3. Di bagian Konfigurasi, di bawah konfigurasi penyimpanan Cluster, pilih Amazon I/O-Optimized DocumentDB.



4. Selesaikan pembuatan atau modifikasi cluster Anda dan pilih Create cluster atau Modify cluster.

   Untuk proses Create cluster yang lengkap, lihatMembuat cluster dan instance primer menggunakan Konsol Manajemen AWS.

   Untuk proses klaster Modify yang lengkap, lihatMemodifikasi cluster Amazon DocumentDB.

Using the AWS CLI

Untuk membuat I/O-optimized cluster menggunakan AWS CLI:

Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi Anda sendiri.

Untuk Linux, macOS, atau 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
```

Untuk 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
```

Analisis biaya untuk menentukan konfigurasi penyimpanan

Dengan Amazon DocumentDB, Anda memiliki fleksibilitas untuk memilih konfigurasi penyimpanan untuk setiap cluster database yang Anda miliki. Untuk mengalokasikan cluster Anda dengan benar antara standar dan I/O-optimized, Anda dapat melacak biaya Amazon DocumentDB Anda berdasarkan klaster. Untuk melakukannya, Anda dapat menambahkan tag ke cluster yang ada, mengaktifkan penandaan alokasi biaya di AWS Manajemen Penagihan dan Biaya dasbor Anda, dan menganalisis biaya Anda untuk klaster tertentu di. AWS Cost Explorer Service Untuk informasi tentang analisis biaya, lihat blog kami Menggunakan tag alokasi biaya.

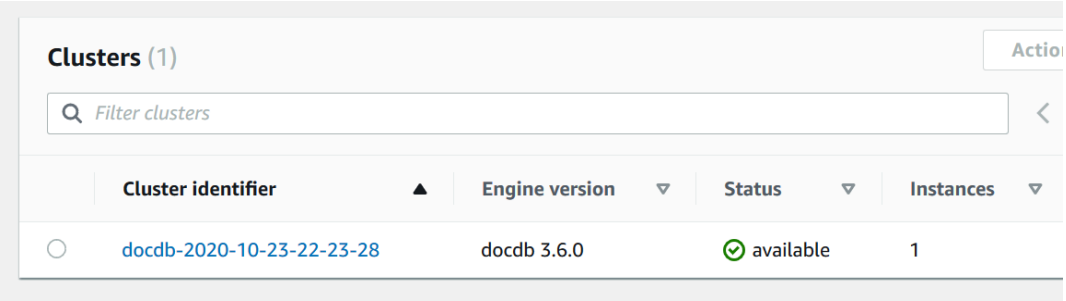
Menentukan status klaster

Anda dapat menentukan status cluster menggunakan Konsol Manajemen AWS atau AWS CLI.

Using the Konsol Manajemen AWS

Gunakan prosedur berikut untuk melihat status klaster Amazon DocumentDB Anda menggunakan Konsol Manajemen AWS

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Klaster.
3. Dalam kolom Pengidentifikasi klaster, temukan nama klaster yang Anda minati. Kemudian, untuk menemukan status klaster, bacalah di baris tersebut pada kolom Status, seperti yang ditunjukkan di bawah ini.



Using the AWS CLI

Gunakan operasi `describe-db-clusters` untuk melihat status klaster Amazon DocumentDB Anda menggunakan AWS CLI.

Kode berikut menemukan status klaster `sample-cluster`.

Untuk Linux, macOS, atau Unix:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster  \
    --query 'DBClusters[*].[DBClusterIdentifier,Status]'
```

Untuk Windows:

```
aws docdb describe-db-clusters ^
    --db-cluster-identifier sample-cluster  ^
    --query 'DBClusters[*].[DBClusterIdentifier,Status]'
```

Output dari operasi ini terlihat seperti berikut (format JSON).

```
[
    [
        "sample-cluster",
```

```
        "available"
    ]
]
```

Siklus hidup klaster Amazon DocumentDB

Siklus hidup dari Klaster Amazon DocumentDB termasuk menciptakan, menggambarkan, memodifikasi, dan menghapus klaster. Bagian ini menyediakan informasi tentang cara menyelesaikan proses ini.

Topik

- Membuat cluster Amazon DocumentDB
- Menggambarkan cluster Amazon DocumentDB
- Memodifikasi cluster Amazon DocumentDB
- Menentukan pemeliharaan yang tertunda
- Melakukan pembaruan tambalan ke versi mesin cluster
- Menghentikan dan memulai cluster Amazon DocumentDB
- Menghapus cluster Amazon DocumentDB

Membuat cluster Amazon DocumentDB

Sebuah klaster Amazon DocumentDB terdiri dari instans dan volume yang mewakili data untuk klaster. Volume klaster direplikasi dengan enam cara di tiga Availability Zone sebagai volume virtual tunggal. Klaster berisi instans utama dan, secara opsional, hingga 15 instans replika.

Bagian berikut menunjukkan cara membuat cluster Amazon DocumentDB menggunakan salah satu atau Konsol Manajemen AWS . AWS CLI Anda kemudian dapat menambahkan instans replika tambahan untuk klaster tersebut. Bila Anda menggunakan konsol untuk membuat klaster Amazon DocumentDB, instans utama secara otomatis dibuat untuk Anda pada saat yang sama. Jika Anda menggunakan AWS CLI untuk membuat klaster Amazon DocumentDB, setelah status klaster tersedia, Anda harus membuat instance utama untuk klaster tersebut.

Prasyarat

Berikut ini adalah prasyarat untuk membuat klaster Amazon DocumentDB.

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar untuk Akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.
2. Ikuti petunjuk online.

   Bagian dari prosedur pendaftaran melibatkan menerima panggilan telepon atau pesan teks dan memasukkan kode verifikasi pada keypad telepon.

   Saat Anda mendaftar untuk sebuah Akun AWS, sebuah Pengguna root akun AWSdibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya di akun. Sebagai praktik keamanan terbaik, tetapkan akses administratif ke pengguna, dan gunakan hanya pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

Prasyarat VPC

Anda hanya dapat membuat cluster Amazon DocumentDB di Amazon Virtual Private Cloud (Amazon VPC). VPC Amazon Anda harus memiliki setidaknya satu subnet di masing-masing setidaknya dua Availability Zone agar Anda dapat menggunakannya dengan cluster Amazon DocumentDB. Dengan mendistribusikan instance klaster di seluruh Availability Zone, Anda memastikan bahwa instance tersedia di klaster jika terjadi kegagalan Availability Zone.

Prasyarat subnet

Saat membuat klaster Amazon DocumentDB, Anda harus memilih VPC dan grup subnet yang sesuai dalam VPC itu untuk meluncurkan klaster Anda. Subnet menentukan Availability Zone dan cakupan IP dalam Availability Zone yang ingin Anda gunakan untuk meluncurkan instans. Untuk tujuan diskusi ini, istilah subnet dan Availability Zone digunakan secara bergantian. Grup subnet adalah set subnet yang telah diberi nama (atau Availability Zone). Apa yang grup subnet izinkan untuk Anda lakukan adalah menentukan Availability Zone yang ingin Anda gunakan untuk meluncurkan instans Amazon DocumentDB. Misalnya, dalam klaster dengan tiga instans, untuk ketersediaan tinggi direkomendasikan agar masing-masing instans tersebut disediakan di Availability Zone yang terpisah. Jadi, jika performa Availability Zone menurun, itu hanya memengaruhi satu instans.

Instans Amazon DocumentDB saat ini dapat disediakan dalam hingga tiga Availability Zone. Bahkan jika grup subnet memiliki lebih dari tiga subnet, Anda hanya dapat menggunakan tiga subnet tersebut untuk membuat klaster Amazon DocumentDB. Oleh karenanya, disarankan agar saat Anda membuat grup subnet, pilih hanya tiga subnet tempat Anda ingin menerapkan instans. Di US East (N. Virginia),

grup subnet Anda dapat memiliki enam subnet (atau Availability Zones). Namun, ketika sebuah klaster Amazon DocumentDB disediakan, Amazon DocumentDB memilih tiga dari Availability Zones yang digunakannya untuk menyediakan instans.

Sebagai contoh, misalkan bahwa ketika Anda membuat sebuah klaster, Amazon DocumentDB memilih Availability Zone {1A, 1B, dan 1C}. Jika Anda mencoba untuk membuat sebuah instans di Availability Zone {1D}, panggilan API gagal. Namun, jika Anda memilih untuk membuat sebuah instans tanpa menentukan Availability Zone tertentu, maka Amazon DocumentDB memilih Availability Zone atas nama Anda. Amazon DocumentDB menggunakan algoritme untuk menyeimbangkan beban instans di Availability Zone untuk membantu Anda mencapai ketersediaan tinggi. Sebagai contoh, jika tiga instans disediakan, secara default, mereka disediakan di tiga Availability Zone dan tidak disediakan secara bersamaan di Availability Zone tunggal.

Rekomendasi:

- Kecuali Anda memiliki alasan tertentu, selalu buat grup subnet dengan tiga subnet. Dengan melakukannya, Anda dibantu untuk memastikan bahwa klaster dengan tiga instans atau lebih dapat mencapai ketersediaan yang lebih tinggi karena instans disediakan di tiga Availability Zone.
- Selalu sebarkan instans di beberapa Availability Zone untuk mencapai ketersediaan yang tinggi. Jangan pernah menempatkan semua instans untuk klaster dalam Availability Zone tunggal.
- Karena peristiwa failover dapat terjadi setiap saat, Anda tidak boleh berasumsi bahwa instans utama atau instans replika selalu berada di Availability Zone tertentu.

Prasyarat tambahan

Berikut ini adalah beberapa prasyarat tambahan untuk menciptakan sebuah klaster Amazon DocumentDB:

- Jika Anda terhubung ke kredensial AWS menggunakan AWS Identity and Access Management (IAM), akun IAM Anda harus memiliki kebijakan IAM yang memberikan izin yang diperlukan untuk menjalankan operasi Amazon DocumentDB.

  Jika Anda menggunakan akun IAM untuk mengakses konsol Amazon DocumentDB, Anda harus terlebih dahulu masuk Konsol Manajemen AWS ke akun dengan IAM Anda. Lalu pergi ke konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
- Jika Anda ingin menyesuaikan parameter konfigurasi untuk klaster, Anda harus menentukan grup parameter klaster dan grup parameter dengan pengaturan parameter yang diperlukan. Untuk

informasi tentang membuat atau memodifikasi grup parameter klaster atau grup parameter, lihat Mengelola grup parameter cluster Amazon DocumentDB.

- Anda harus menentukan nomor TCP/IP port yang ingin Anda tentukan untuk cluster Anda. Firewall di beberapa perusahaan memblokir koneksi ke port default untuk Amazon DocumentDB. Jika firewall perusahaan Anda memblokir port default, pilih port lain untuk klaster Anda. Semua instans dalam klaster menggunakan port yang sama.

Membuat cluster dan instance primer menggunakan Konsol Manajemen AWS

Prosedur berikut menjelaskan cara menggunakan konsol untuk meluncurkan klaster Amazon DocumentDB dengan satu atau beberapa instans.

Buat cluster: menggunakan pengaturan default

Untuk membuat cluster dengan instance menggunakan pengaturan default menggunakan Konsol Manajemen AWS

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Jika Anda ingin membuat klaster di wilayah Wilayah AWS lain selain Wilayah AS Timur (Virginia N.), pilih Wilayah dari daftar di bagian kanan atas konsol.
3. Di panel navigasi, pilih Klaster, lalu pilih Buat.

   > **Tip**
   >
   > Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu (≡) di pojok kiri atas halaman.

4. Pada halaman Buat klaster Amazon DocumentDB, selesaikan panel Konfigurasi.

   a. Pengidentifikasi klaster—Terima nama yang disediakan oleh Amazon DocumentDB, atau masukkan nama untuk klaster Anda; misalnya, **`sample-cluster`**.

      Aturan penamaan klaster:

      - Panjangnya adalah [1—63] huruf, angka, atau tanda hubung.
      - Karakter pertama harus berupa huruf.

- Tidak dapat diakhiri dengan tanda hubung atau berisi dua tanda hubung berurutan.
- Harus unik untuk semua cluster di Amazon RDS, Neptunus, dan Amazon DocumentDB per, per Wilayah. Akun AWS

b. Versi mesin —Terima versi mesin default 5.0.0, atau opsional pilih 8.0.0, 4.0.0, atau 3.6.0.

c. Kelas instans—Terima default`db.r5.large`, atau pilih kelas instans yang Anda inginkan dari daftar.

d. Jumlah instans—Dalam daftar, pilih jumlah instans yang ingin Anda buat dengan klaster ini. Instans pertama adalah instans utama, dan semua instans lain adalah instans replika hanya baca. Anda dapat menambahkan dan menghapus instans nanti jika Anda perlu. Secara default, sebuah klaster Amazon DocumentDB melakukan peluncuran dengan tiga instans (satu primer dan dua replika).

5. Lengkapi bagian konfigurasi penyimpanan Cluster.

   Pilih Amazon DocumentDB Standard (default) atau Amazon DocumentDB. I/O-Optimized Untuk informasi selengkapnya, lihat Konfigurasi penyimpanan cluster Amazon DocumentDB.

6. Menyelesaikan panel Autentikasi.

   a. Username —Masukkan nama untuk pengguna utama. Untuk masuk ke cluster Anda, Anda harus menggunakan nama pengguna utama.

      Kendala penamaan pengguna utama:

      - Panjangnya adalah [1—63] karakter alfanumerik.
      - Karakter pertamanya harus berupa huruf.
      - Tidak dapat berupa kata yang dicadangkan oleh mesin basis data.

   b. Pilih salah satu opsi kata sandi berikut:

      - Dikelola di AWS Secrets Manager —Pilih opsi ini jika Anda AWS Secrets Manager ingin mengelola kata sandi pengguna utama secara otomatis.

        Jika Anda memilih opsi ini, konfigurasikan kunci KMS dengan membuat sendiri atau menggunakan kunci yang dibuat Secrets Manager.

      - Dikelola sendiri —Pilih opsi ini jika Anda ingin mengelola sendiri kata sandi pengguna utama Anda. Jika Anda memilih opsi ini, masukkan kata sandi untuk pengguna utama, lalu konfirmasikan. Untuk masuk ke cluster Anda, Anda harus menggunakan kata sandi untuk pengguna utama.

Aturan kata sandi:

- Panjangnya adalah [8-100] karakter ASCII yang dapat dicetak.
- Dapat menggunakan karakter ASCII apa pun yang dapat dicetak kecuali untuk yang berikut ini:
  - **/** (garis miring ke depan)
  - **"** (tanda kutip ganda)
  - **@** (simbol pada)

7. Di bagian bawah layar, pilih salah satu dari berikut ini:
   - Untuk membuat klaster sekarang, pilih Buat klaster.
   - Untuk membatalkan pembuatan klaster, pilih Batalkan.
   - Untuk lebih mengkonfigurasi klaster sebelum membuat, pilih Menampilkan konfigurasi tambahan, dan kemudian lanjutkan di Buat cluster: konfigurasi tambahan.

     Konfigurasi yang tercakup dalam bagian Konfigurasi tambahan adalah sebagai berikut:

     - Pengaturan jaringan—Default di sini akan menggunakan grup keamanan VPC `default`.
     - Opsi klaster—Default di sini akan menggunakan port 27017 dan grup parameter default.
     - Enkripsi—Default adalah untuk mengaktifkan enkripsi dengan menggunakan kunci `(default) aws/rds`.

       > **Important**
       >
       > Setelah dienkripsi, klaster tidak dapat dibuat tanpa enkripsi.

     - Backup—Default adalah untuk mempertahankan backup selama 1 hari dan membiarkan Amazon DocumentDB memilih jendela backup.
     - Ekspor log —Defaultnya adalah tidak mengekspor log audit ke CloudWatch Log.
     - Pemeliharaan—Defaultnya adalah membiarkan Amazon DocumentDB memilih jendela pemeliharaan.
     - Perlindungan penghapusan—Lindungi klaster Anda dari penghapusan yang tidak disengaja. Default untuk klaster yang dibuat dengan menggunakan konsol diaktifkan.

     Jika Anda menerima pengaturan default sekarang, Anda dapat mengubah sebagian besar pengaturan tersebut nanti dengan memodifikasi klaster.

8. Mengaktifkan sambungan masuk untuk grup keamanan klaster Anda.

   Jika Anda tidak mengubah pengaturan default untuk klaster Anda, Anda membuat sebuah klaster menggunakan grup keamanan default untuk VPC default di wilayah tertentu. Untuk terhubung ke Amazon DocumentDB, Anda harus mengaktifkan koneksi masuk pada port 27017 (atau port pilihan Anda) untuk grup keamanan klaster Anda.

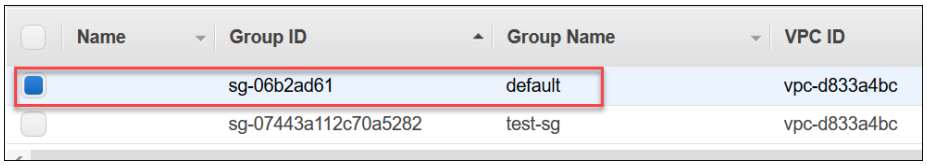
   Untuk menambahkan koneksi masuk ke grup keamanan klaster

   a. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon EC2 di. https://console.aws.amazon.com/ec2/

   b. Di bagian Sumber Daya dari jendela utama, pilih Grup keamanan.



   c. Dari daftar grup keamanan, temukan grup keamanan yang Anda gunakan saat membuat klaster Anda (kemungkinan besar grup keamanan default) dan pilih kotak di sebelah kiri nama grup keamanan.



   d. Dari menu Tindakan, pilih Edit aturan masuk lalu pilih atau masukkan batasan aturan.

      i. Jenis—Dari daftar tersebut, pilih protokol untuk membuka lalu lintas jaringan.

      ii. Protokol—Dari daftar, pilih jenis protokol.

      iii. Cakupan Port—Untuk aturan kustom, masukkan nomor port atau cakupan port. Pastikan bahwa nomor port atau cakupan terdiri atas port yang Anda tentukan ketika membuat klaster Anda (default: 27017).

      iv. Sumber—Menentukan lalu lintas yang dapat menjangkau instans Anda. Dari daftar, pilih sumber lalu lintas. Jika Anda memilih Kustom, tentukan satu alamat IP atau rentang alamat IP dalam notasi CIDR (mis., 203.0.113. 5/32).

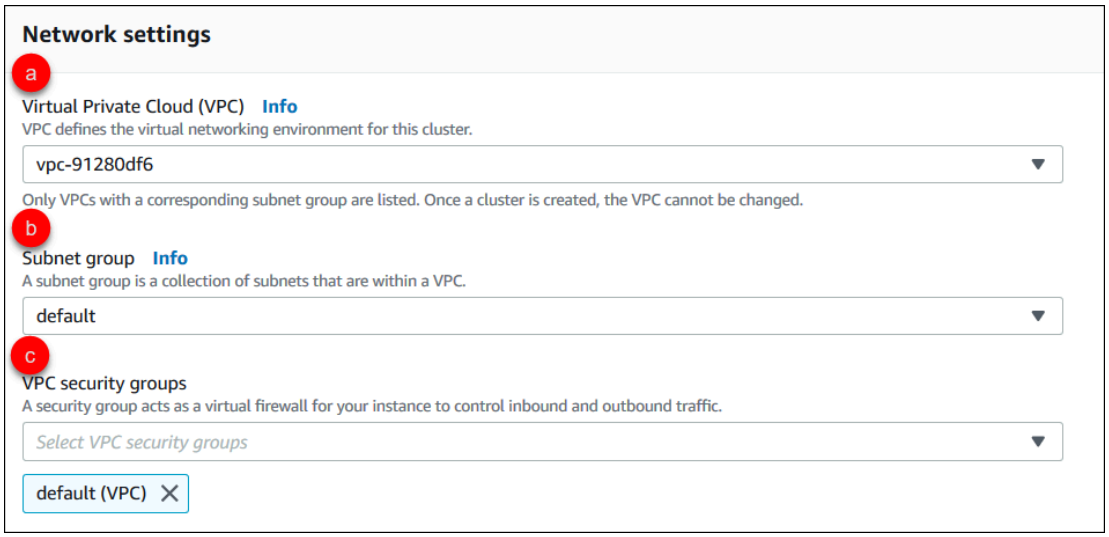
      v. Deskripsi—Masukkan deskripsi untuk aturan ini.

vi. Setelah selesai membuat aturan, pilih Simpan.

Buat cluster: konfigurasi tambahan

Jika Anda ingin menerima pengaturan default untuk klaster Anda, Anda dapat melewatkan langkah-langkah berikut dan memilih Buat klaster.

1. Menyelesaikan panel Pengaturan jaringan.



   a. Virtual Private Cloud (VPC)—Dalam daftar, pilih Amazon VPC tempat Anda ingin meluncurkan klaster ini.

   b. Grup subnet—Dalam daftar, pilih grup subnet yang ingin Anda gunakan untuk klaster ini.

   c. Grup keamanan VPC—Di daftar, pilih grup keamanan VPC untuk klaster ini.

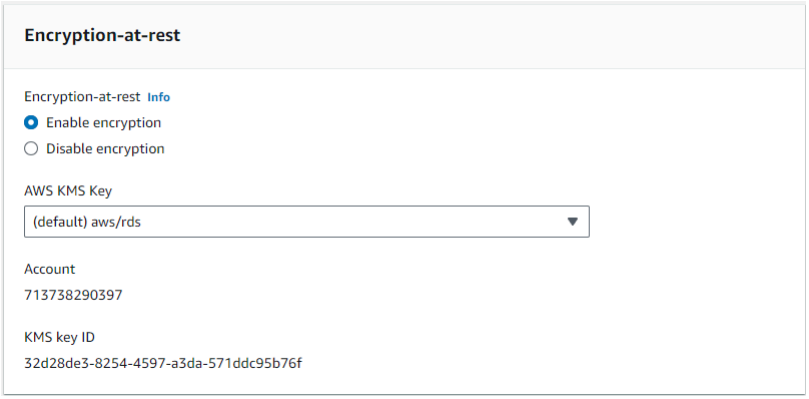
2. Lengkapi panel Opsi klaster.



   a. Port basis data —Gunakan panah atas dan bawah untuk mengatur TCP/IP port yang akan digunakan aplikasi untuk terhubung ke instans Anda.

b. Grup parameter klaster—Dalam daftar grup parameter, pilih grup parameter klaster untuk klaster ini.

3. Menyelesaikan panel Enkripsi.



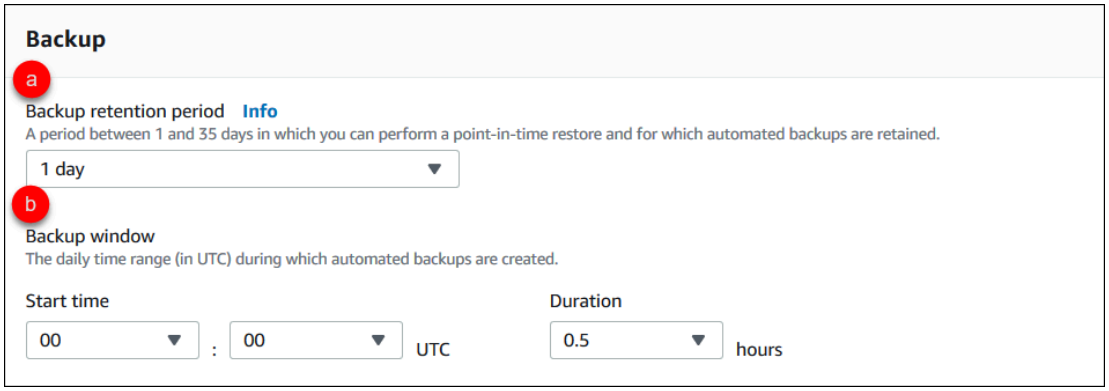
a. Encryption-at-rest—Pilih salah satu dari berikut ini:

- Aktifkan enkripsi—Default. Semua data at rest dienkripsi. Jika Anda memilih untuk mengenkripsi data, Anda tidak dapat membatalkan tindakan ini.
- Menonaktifkan enkripsi—Data Anda tidak dienkripsi.

b. AWS Kunci KMS —Ini hanya tersedia jika Anda mengenkripsi data Anda. Dalam daftar, pilih kunci yang ingin Anda gunakan untuk mengenkripsi data dalam klaster ini. Nilai default-nya `(default) aws/rds`.

Jika Anda memilih Masukkan ARN kunci, Anda harus memasukkan Amazon Resource Name (ARN) untuk kunci tersebut.

4. Menyelesaikan panel Backup.



a. Periode retensi cadangan—Dalam daftar, pilih jumlah hari untuk menyimpan backup otomatis klaster ini sebelum menghapusnya.

b. Jendela backup—Atur waktu harian dan durasi yang akan digunakan oleh Amazon DocumentDB untuk membuat backup dari klaster ini.

   i. Waktu mulai—Pada daftar pertama, pilih jam waktu mulai (UTC) untuk memulai pencadangan otomatis. Dalam daftar kedua, pilih menit dari jam pilihan Anda untuk memulai backup otomatis.

   ii. Durasi—Dalam daftar, pilih jumlah jam yang akan dialokasikan untuk membuat cadangan otomatis.

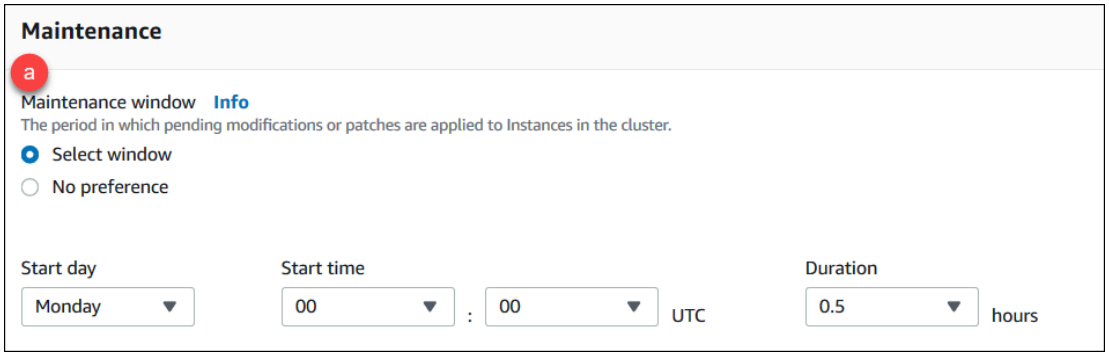
5. Lengkapi panel ekspor Log dengan memilih jenis log yang ingin Anda ekspor ke CloudWatch Log.



- Log audit —Pilih opsi ini untuk mengaktifkan mengekspor log audit ke Amazon CloudWatch Logs. Jika Anda memilih Log audit, Anda harus mengaktifkan `audit_logs` di grup parameter kustom klaster. Untuk informasi selengkapnya, lihat Mengaudit acara Amazon DocumentDB.
- Log profiler —Pilih opsi ini untuk mengaktifkan mengekspor log profiler operasi ke Amazon Logs. CloudWatch Jika Anda memilih Log profil, Anda juga harus mengubah parameter berikut dalam grup parameter kustom klaster:
  - `profiler`—Setel `keenabled`.
  - `profiler_threshold_ms`—Set ke nilai `[0-INT_MAX]` untuk mengatur ambang batas pada profil operasi.
  - `profiler_sampling_rate`—Set ke nilai `[0.0-1.0]` untuk mengatur persentase operasi lambat ke profil.

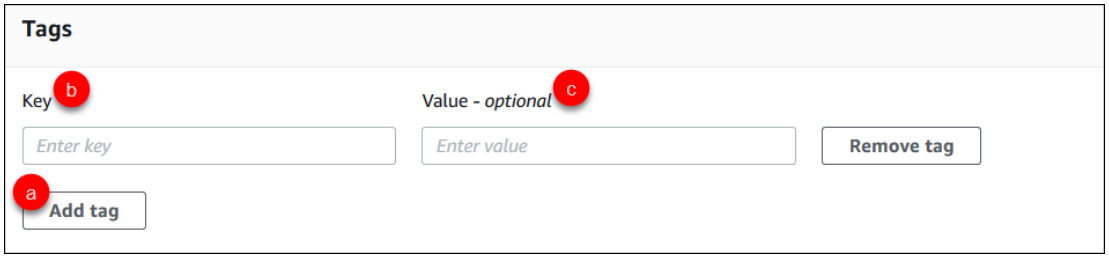
  Untuk informasi selengkapnya, lihat Membuat profil operasi Amazon DocumentDB.

6. Menyelesaikan panel Pemeliharaan.



- Pilih salah satu dari berikut
  - Pilih jendela—Anda dapat menentukan hari dalam seminggu, waktu mulai UTC, dan durasi untuk Amazon DocumentDB untuk melakukan pemeliharaan pada klaster Anda.
    a. Hari mulai—Dalam daftar, pilih hari dalam seminggu untuk memulai pemeliharaan klaster.
    b. Waktu mulai—Dalam daftar, pilih jam dan menit (UTC) untuk memulai pemeliharaan.
    c. Durasi—Dalam daftar, pilih berapa banyak waktu yang akan dialokasikan untuk pemeliharaan klaster. Jika pemeliharaan tidak dapat diselesaikan dalam waktu yang ditentukan, proses pemeliharaan akan berlanjut melewati waktu yang ditentukan sampai selesai.
  - Tidak Ada preferensi-Amazon DocumentDB memilih hari dalam seminggu, waktu mulai, dan durasi untuk melakukan pemeliharaan.

7. Jika Anda ingin menambahkan satu atau beberapa tag ke klaster ini, lengkapi panel Tag.



Untuk setiap tag yang ingin Anda tambahkan ke klaster, ulangi langkah-langkah berikut. Anda mungkin memiliki hingga 10 di sebuah klaster.

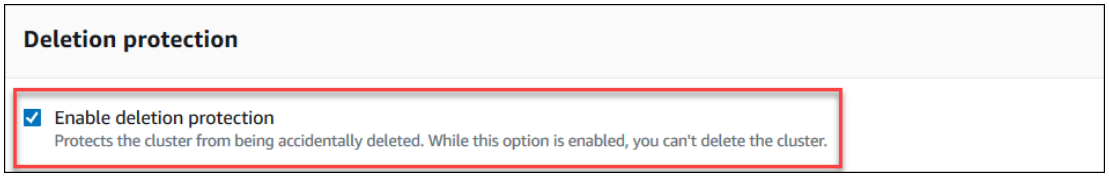
a. Pilih Tambahkan tag.

b. Ketik Kunci tag.

c. Secara opsional, ketik Nilai tag.

Untuk menghapus sebuah tag, pilih Hapus tag.

8. Perlindungan penghapusan diaktifkan secara default ketika Anda membuat klaster dengan menggunakan konsol. Untuk menonaktifkan perlindungan penghapusan, kosongkan Aktifkan perlindungan penghapusan. Ketika diaktifkan, perlindungan penghapusan mencegah agar klaster tidak terhapus. Untuk menghapus klaster yang dilindungi dari penghapusan, Anda harus terlebih dahulu memodifikasi klaster untuk menonaktifkan penghapusan perlindungan.



Untuk informasi selengkapnya tentang perlindungan penghapusan, lihat Menghapus cluster Amazon DocumentDB.

9. Untuk membuat klaster, pilih Buat klaster. Jika tidak, pilih Batalkan.

Membuat cluster menggunakan AWS CLI

Prosedur berikut menjelaskan cara menggunakan AWS CLI untuk meluncurkan cluster Amazon DocumentDB dan membuat replika Amazon DocumentDB.

Parameter

- **`--db-cluster-identifier`**—Diperlukan. Sebuah string huruf kecil yang mengidentifikasi klaster ini.

  Aturan Penamaan klaster:

  - Panjangnya adalah [1-63] huruf, angka, atau tanda hubung.
  - Karakter pertama harus berupa huruf.
  - Tidak dapat diakhiri dengan tanda hubung atau berisi dua tanda hubung berurutan.
  - Harus unik untuk semua cluster (di Amazon RDS, Amazon Neptune, dan Amazon DocumentDB) per akun, per Wilayah. AWS

- **`--engine`**—Diperlukan. Harus berupa **`docdb`**.

- **`--deletion-protection | --no-deletion-protection`**—Opsional. Ketika diaktifkan, perlindungan penghapusan mencegah agar klaster tidak terhapus. Saat Anda menggunakan AWS CLI, pengaturan defaultnya adalah menonaktifkan perlindungan penghapusan.

  Untuk informasi selengkapnya tentang perlindungan penghapusan, lihat Menghapus cluster Amazon DocumentDB.

- **`--storage-type standard | iopt1`**—Opsional. Default: **`standard`**. Konfigurasi penyimpanan cluster. Nilai yang valid adalah `standard` (Standar) atau `iopt1` (I/O-optimized).

- **`--master-username`**—Diperlukan. Nama pengguna yang digunakan untuk mengautentikasi pengguna.

  Aturan Penamaan Pengguna Utama:

  - Panjang adalah [1-63] karakter alfanumerik.
  - Karakter pertamanya harus berupa huruf.
  - Tidak dapat berupa kata yang dicadangkan oleh mesin basis data.

- **`--master-user-password`**—Opsional. Kata sandi pengguna yang digunakan untuk mengautentikasi pengguna.

  Aturan Kata Sandi Utama:

  - Panjang adalah [8-100] karakter ASCII yang dapat dicetak.
  - Dapat menggunakan karakter ASCII apa pun yang dapat dicetak kecuali untuk yang berikut ini:
    - **`/`** (garis miring ke depan)
    - **`"`** (tanda kutip ganda)
    - **`@`** (simbol pada)

- **`--manage-master-user-password`**—Opsional. Amazon DocumentDB menghasilkan kata sandi pengguna utama dan mengelolanya sepanjang siklus hidupnya di Secrets Manager.

Untuk parameter tambahan, lihat CreateDBCluster.

Untuk meluncurkan cluster Amazon DocumentDB menggunakan AWS CLI

Untuk membuat cluster Amazon DocumentDB, hubungi file. `create-db-cluster` AWS CLI AWS CLI Perintah berikut membuat klaster Amazon DocumentDB `sample-cluster` bernama dengan perlindungan penghapusan diaktifkan. Untuk informasi selengkapnya tentang perlindungan penghapusan, lihat Menghapus cluster Amazon DocumentDB.

Juga, `--engine-version` adalah parameter opsional yang default ke versi mesin utama terbaru. Versi mesin default saat ini adalah 5.0.0 (catatan: Amazon DocumentDB 8.0 tersedia tetapi harus ditentukan secara eksplisit). Ketika versi mesin utama baru dirilis, versi mesin default untuk `--engine-version` akan diperbarui untuk mencerminkan versi mesin utama terakhir. Akibatnya, untuk beban kerja produksi, dan terutama yang bergantung pada skrip, otomatisasi, atau CloudFormation templat, kami menyarankan Anda secara eksplisit menentukan `--engine-version` ke versi utama yang dimaksud.

> **Note**
>
> Jika `db-subnet-group-name` atau `vpc-security-group-id` tidak ditentukan, Amazon DocumentDB akan menggunakan grup subnet default dan grup keamanan Amazon VPC untuk wilayah tertentu.

Untuk Linux, macOS, atau 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
```

Untuk 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
```

Output dari operasi ini terlihat seperti berikut (format 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"
    }
}
```

Dibutuhkan beberapa menit untuk membuat klaster. Anda dapat menggunakan Konsol Manajemen AWS atau AWS CLI untuk memantau status cluster Anda. Untuk informasi selengkapnya, lihat Memantau status klaster Amazon DocumentDB.

> **Important**
>
> Saat Anda menggunakan AWS CLI untuk membuat klaster Amazon DocumentDB, tidak ada instance yang dibuat. Akibatnya, Anda harus secara eksplisit membuat instans primer dan semua instans replika yang Anda butuhkan. Anda dapat menggunakan konsol atau AWS CLI untuk membuat instance. Untuk informasi selengkapnya, lihat Menambahkan instance Amazon DocumentDB ke klaster.

Untuk informasi selengkapnya, lihat `CreateDBCluster` di Referensi API Amazon DocumentDB.

Menggambarkan cluster Amazon DocumentDB

Anda dapat menggunakan Amazon DocumentDB Management Console atau AWS CLI untuk melihat detail seperti titik akhir koneksi, grup keamanan, VPC, dan grup parameter yang berkaitan dengan cluster Amazon DocumentDB Anda.

Untuk informasi selengkapnya, lihat berikut ini:

- Memantau status klaster Amazon DocumentDB
- Menemukan titik akhir klaster

Using the Konsol Manajemen AWS

Gunakan prosedur berikut untuk melihat perincian klaster Amazon DocumentDB yang ditentukan dengan menggunakan konsol.

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Klaster.

> **Tip**
>
> Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu (≡) di pojok kiri atas halaman.

3. Dalam daftar klaster, pilih nama klaster yang ingin Anda lihat perinciannya. Informasi tentang klaster diatur ke dalam pengelompokan berikut:

   - Ringkasan — Informasi umum tentang klaster, termasuk versi mesin, status klaster, pemeliharaan tertunda, dan status grup parameter.
   - Konektivitas & Keamanan —Bagian Hubungkan berisi daftar titik akhir sambungan untuk membuat sambungan ke klaster ini dengan mongo shell atau dengan aplikasi. Bagian Grup Keamanan terdiri atas daftar grup keamanan yang terkait dengan klaster ini dan ID VPC serta deskripsinya.
   - Konfigurasi — Bagian Detail klaster berisi daftar perincian tentang klaster, termasuk Amazon Resource Name (ARN) klaster, titik akhir, dan grup parameter. Ini juga berisi daftar informasi backup klaster, perincian pemeliharaan, dan pengaturan keamanan dan jaringan. Bagian Instans klaster berisi daftar instans yang dimiliki klaster ini dengan masing-masing peran instans dan status grup parameter klaster.
   - Monitoring — Metrik Amazon CloudWatch Logs untuk klaster ini. Untuk informasi selengkapnya, lihat Memantau Amazon DocumentDB dengan CloudWatch.
   - Acara & tag — Bagian Peristiwa terbaru berisi daftar peristiwa terbaru untuk klaster ini. Amazon DocumentDB menyimpan catatan peristiwa yang terkait dengan klaster, instans, snapshot, grup keamanan, dan grup parameter klaster Anda. Informasi ini mencakup tanggal, waktu, dan pesan yang terkait dengan setiap peristiwa. Bagian Tag berisi daftar tag yang dilampirkan ke klaster ini.

Using the AWS CLI

Untuk melihat detail cluster Amazon DocumentDB Anda menggunakan, gunakan perintah seperti AWS CLI yang ditunjukkan `describe-db-clusters` pada contoh berikut. Untuk informasi selengkapnya, lihat `DescribeDBClusters`di Referensi API Manajemen Sumber Daya Amazon DocumentDB.

> **Note**
>
> Untuk fitur pengelolaan tertentu seperti pengelolaan siklus hidup klaster dan instans, Amazon DocumentDB memanfaatkan teknologi operasi yang dibagikan dengan Amazon RDS. Parameter filter `filterName=engine,Values=docdb` hanya mengembalikan klaster Amazon DocumentDB.

Example

Contoh 1: Daftar semua cluster Amazon DocumentDB

AWS CLI Kode berikut mencantumkan detail untuk semua cluster Amazon DocumentDB di suatu wilayah.

```
aws docdb describe-db-clusters --filter Name=engine,Values=docdb
```

Output dari operasi ini akan terlihat seperti berikut.

```
{
    "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

Contoh 2: Daftar semua detail untuk klaster Amazon DocumentDB tertentu

AWS CLI Kode berikut mencantumkan rincian untuk cluster`sample-cluster`.

Untuk Linux, macOS, atau Unix:

```
aws docdb describe-db-clusters \
   --filter Name=engine,Values=docdb \
   --db-cluster-identifier sample-cluster
```

Untuk Windows:

```
aws docdb describe-db-clusters ^
   --filter Name=engine,Values=docdb ^
   --db-cluster-identifier sample-cluster
```

Output dari operasi ini terlihat seperti berikut ini.

```
{
    "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

Contoh 3: Daftar detail spesifik untuk klaster Amazon DocumentDB

Untuk mencantumkan subset detail kluster menggunakan AWS CLI, tambahkan yang menentukan anggota klaster mana yang akan dicantumkan `describe-db-clusters` operasi. `--query` Parameter `--db-cluster-identifier` adalah pengenal untuk klaster tertentu yang perinciannya ingin Anda tampilkan. Untuk informasi selengkapnya tentang kueri, lihat Cara Filter Output dengan `--query` Opsi di AWS Command Line Interface Panduan Pengguna.

Contoh berikut mencantumkan instans dalam klaster Amazon DocumentDB.

Untuk Linux, macOS, atau Unix:

```
aws docdb describe-db-clusters \
    --filter Name=engine,Values=docdb \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].[DBClusterMembers]'
```

Untuk Windows:

```
aws docdb describe-db-clusters ^
    --filter Name=engine,Values=docdb ^
    --db-cluster-identifier sample-cluster ^
    --query 'DBClusters[*].[DBClusterMembers]'
```

Output dari operasi ini terlihat seperti berikut ini.

```
[
    [
        [
            {
                "DBInstanceIdentifier": "sample-instance-1",
                "IsClusterWriter": true,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
            },
            {
                "DBInstanceIdentifier": "sample-instance-2",
                "IsClusterWriter": false,
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1
            }
        ]
    ]
]
```

Memodifikasi cluster Amazon DocumentDB

Untuk mengubah suatu klaster, klaster tersebut harus dalam kondisi tersedia. Anda tidak dapat mengubah klaster yang dihentikan. Jika klaster berhenti, pertama mulai klaster, tunggu hingga klaster tersedia, lalu buat modifikasi yang diinginkan. Untuk informasi selengkapnya, lihat Menghentikan dan memulai cluster Amazon DocumentDB.

Using the Konsol Manajemen AWS

Gunakan prosedur berikut untuk memodifikasi klaster Amazon DocumentDB tertentu menggunakan konsol.

Cara memodifikasi klaster Amazon DocumentDB

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Klaster.

   > **Tip**
   >
   > Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu (≡) di pojok kiri atas halaman.

3. Tentukan klaster yang ingin Anda modifikasi dengan memilih tombol di sebelah kiri nama klaster.
4. Pilih Tindakan, lalu pilih Modifikasi.
5. Di Modifikasi klaster: panel <cluster-name>, buat perubahan yang Anda inginkan. Anda dapat membuat perubahan di bidang berikut:

   - Spesifikasi klaster —Nama klaster, grup keamanan, dan manajemen kredensialnya.
   - Konfigurasi penyimpanan cluster —Mode penyimpanan data cluster. Pilih antara Standar dan I/O-Optimized konfigurasi.
   - Opsi klaster—Port klaster dan grup parameter.
   - Backup—Periode penyimpanan backup klaster dan jendela backup.
   - Ekspor log—Aktifkan atau nonaktifkan ekspor audit atau log pembuat profil.
   - Pemeliharaan—Atur jendela pemeliharaan klaster.
   - Perlindungan penghapusan—Aktifkan atau nonaktifkan perlindungan penghapusan pada klaster. Perlindungan penghapusan diaktifkan secara default.
6. Setelah selesai, pilih Lanjutkan untuk melihat ringkasan perubahan Anda.
7. Jika Anda puas dengan perubahan Anda, Anda dapat memilih Modifikasi klaster untuk memodifikasi klaster Anda. Sebagai alternatif, Anda dapat memilih Kembali atau Batalkan untuk mengedit atau membatalkan perubahan Anda, masing-masing.

Diperlukan waktu beberapa menit agar perubahan Anda diterapkan. Anda dapat menggunakan klaster hanya ketika statusnya tersedia. Anda dapat memantau status klaster menggunakan

konsol atau AWS CLI. Untuk informasi selengkapnya, lihat Memantau status klaster Amazon DocumentDB.

Using the AWS CLI

Gunakan operasi `modify-db-cluster` untuk mengubah klaster yang ditentukan dengan menggunakan AWS CLI. Untuk informasi selengkapnya, lihat `ModifyDBCluster` di Referensi API Amazon DocumentDB.

Parameter

- **`--db-cluster-identifier`**—Diperlukan. Pengenal klaster Amazon DocumentDB yang akan Anda modifikasi.
- **`--backup-retention-period`**—Opsional. Jumlah hari untuk penyimpanan cadangan otomatis dipertahankan. Nilai yang valid adalah 1–35.
- **`--storage-type`**—Opsional. Konfigurasi penyimpanan cluster. Nilai yang valid adalah `standard` (Standar) atau `iopt1` (I/O-optimized).
- **`--db-cluster-parameter-group-name`**—Opsional. Nama grup parameter klaster yang digunakan untuk klaster.
- **`--manage-master-user-password`**—Opsional. Amazon DocumentDB menghasilkan kata sandi pengguna utama dan mengelolanya sepanjang siklus hidupnya di Secrets Manager.
- **`--rotate-master-user-password`**—Opsional. Secrets Manager menghasilkan versi rahasia baru untuk rahasia yang ada. Versi baru rahasia berisi kata sandi pengguna utama yang baru. Amazon DocumentDB mengubah kata sandi pengguna utama untuk cluster agar sesuai dengan kata sandi untuk versi rahasia baru.

  Anda harus menentukan **`--apply-immediately`** opsi ketika Anda memutar kata sandi utama.
- **`--master-user-password`**—Opsional. Kata sandi baru untuk pengguna basis data utama.

  Aturan kata sandi:

  - Panjang adalah [8—100] karakter ASCII yang dapat dicetak.
  - Dapat menggunakan karakter ASCII yang dapat dicetak kecuali untuk berikut ini:
    - **`/`** (garis miring ke depan)
    - **`"`** (tanda kutip ganda)
    - **`@`** (simbol pada)

- **`--new-db-cluster-identifier`**—Opsional. Pengenal klaster baru untuk klaster ketika mengubah nama klaster. Nilai ini disimpan sebagai string huruf kecil.

  Aturan penamaan:

  - Panjangnya adalah [1—63] huruf, angka, atau tanda hubung.
  - Karakter pertama harus berupa huruf.
  - Tidak dapat diakhiri dengan tanda hubung atau berisi dua tanda hubung berurutan.
  - Harus unik untuk semua cluster di Amazon RDS, Amazon Neptune, dan Amazon DocumentDB per, per wilayah. Akun AWS
- **`--preferred-backup-window`**—Opsional. Dalam rentang waktu harian selama pencadangan otomatis dibuat, dalam Waktu Terkoordinasi Universal (UTC).
  - Format: `hh24:mm-hh24:mm`
- **`--preferred-maintenance-window`**—Opsional. Rentang waktu mingguan untuk melakukan pemeliharaan sistem dapat terjadi, dalam UTC.
  - Format: `ddd:hh24:mm-ddd:hh24:mm`
  - Hari yang sah: `Sun`, `Mon`, `Tue`, `Wed`, `Thu`, `Fri`, dan `Sat`.
- **`--deletion-protection`** atau **`--no-deletion-protection`**—Opsional. Apakah perlindungan penghapusan harus diaktifkan pada klaster ini. Perlindungan penghapusan mencegah sebuah klaster dari yang sengaja dihapus sampai klaster dimodifikasi untuk menonaktifkan perlindungan penghapusan. Untuk informasi selengkapnya, lihat Menghapus cluster Amazon DocumentDB.
- **`--apply-immediately`** atau **`--no-apply-immediately`**—Gunakan `--apply-immediately` untuk segera membuat perubahan. Gunakan `--no-apply-immediately` untuk membuat perubahan selama jendela pemeliharaan berikutnya pada klaster Anda.

Example

Kode berikut mengubah periode penyimpanan cadangan untuk klaster `sample-cluster`.

Untuk Linux, macOS, atau Unix:

```
aws docdb modify-db-cluster \
       --db-cluster-identifier sample-cluster \
       --apply-immediately \
       --backup-retention-period 7
```

Untuk Windows:

```
aws docdb modify-db-cluster ^
       --db-cluster-identifier sample-cluster ^
       --apply-immediately ^
       --backup-retention-period 7
```

Output dari operasi ini terlihat seperti berikut ini.

```
{
    "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
    }
}
```

Diperlukan waktu beberapa menit agar perubahan Anda diterapkan. Anda dapat menggunakan klaster hanya ketika statusnya tersedia. Anda dapat memantau status klaster menggunakan konsol atau AWS CLI. Untuk informasi selengkapnya, lihat Memantau status klaster Amazon DocumentDB.

Menentukan pemeliharaan yang tertunda

Anda dapat menentukan apakah Anda memiliki versi mesin Amazon DocumentDB terbaru dengan menentukan apakah Anda memiliki pemeliharaan klaster tertunda.

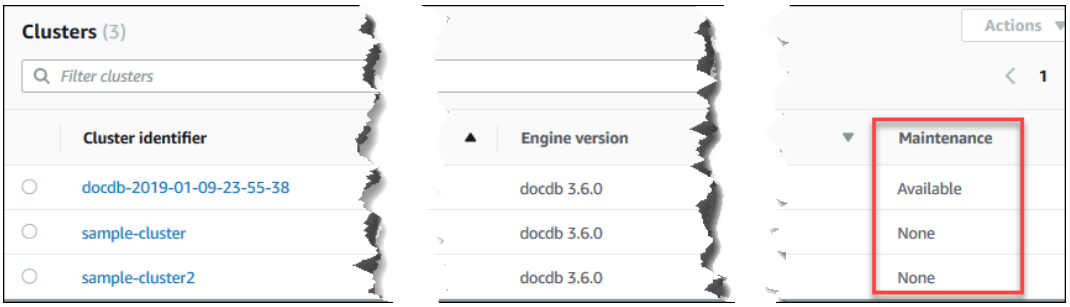
Using the Konsol Manajemen AWS

Anda dapat menggunakan Konsol Manajemen AWS untuk menentukan apakah klaster memiliki pemeliharaan yang tertunda.

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Klaster.

> **Tip**
>
> Jika panel navigasi tidak terlihat di sisi kiri layar Anda, pilih ikon menu (≡) di sudut kiri atas halaman.

3. Temukan kolom Pemeliharaan untuk menentukan apakah sebuah klaster memiliki pemeliharaan tertunda.



   Tidak ada menunjukkan bahwa klaster menjalankan versi mesin terbaru. Tersedia menunjukkan bahwa klaster memiliki pemeliharaan tertunda, yang mungkin berarti bahwa peningkatan mesin diperlukan.

4. Jika klaster Anda memiliki pemeliharaan tertunda, lanjutkan dengan langkah-langkah di Melakukan pembaruan tambalan ke versi mesin cluster.

Using the AWS CLI

Anda dapat menggunakan AWS CLI untuk menentukan apakah sebuah cluster memiliki versi mesin terbaru dengan menggunakan `describe-pending-maintenance-actions` operasi dengan parameter berikut.

Parameter

- **`--resource-identifier`**—Opsional. ARN untuk sumber daya (klaster). Jika parameter ini dihilangkan, tindakan pemeliharaan tertunda untuk semua klaster didaftarkan.
- **`--region`**—Opsional. Wilayah AWS tempat Anda ingin menjalankan operasi ini, misalnya, `us-east-1`.

Example

Untuk Linux, macOS, atau Unix:

```
aws docdb describe-pending-maintenance-actions \
   --resource-identifier arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster \
   --region us-east-1
```

Untuk Windows:

```
aws docdb describe-pending-maintenance-actions ^
   --resource-identifier arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster ^
   --region us-east-1
```

Output dari operasi ini terlihat seperti berikut ini.

```
{
    "PendingMaintenanceActions": [
        {
            "ResourceIdentifier": "arn:aws:rds:us-
east-1:123456789012:cluster:sample-cluster",
            "PendingMaintenanceActionDetails": [
                {
                    "Description": "New feature",
                    "Action": "db-upgrade",
                    "ForcedApplyDate": "2019-02-25T21:46:00Z",
                    "AutoAppliedAfterDate": "2019-02-25T07:41:00Z",
                    "CurrentApplyDate": "2019-02-25T07:41:00Z"
                }
            ]
        }
    ]
}
```

Jika klaster Anda memiliki pemeliharaan tertunda, lanjutkan dengan langkah-langkah di Melakukan pembaruan tambalan ke versi mesin cluster.

Melakukan pembaruan tambalan ke versi mesin cluster

Di bagian ini, kami akan menjelaskan cara menerapkan pembaruan tambalan menggunakan Konsol Manajemen AWS atau. AWS CLI Pembaruan patch adalah pembaruan dalam versi mesin yang sama (misalnya, memperbarui versi mesin 3.6 ke versi mesin 3.6 yang lebih baru). Anda dapat memperbaruinya segera atau selama jendela pemeliharaan berikutnya pada klaster Anda. Untuk menentukan apakah mesin Anda memerlukan pembaruan, lihat Menentukan pemeliharaan yang tertunda. Perhatikan bahwa saat Anda menerapkan pembaruan, klaster Anda akan mengalami beberapa waktu henti.

> **Note**
>
> Jika Anda mencoba untuk meng-upgrade dari versi mesin utama ke yang lain, seperti 3.6 ke 5.0 atau 5.0 ke 8.0, lihat salah satu atauPeningkatan versi utama Amazon DocumentDB di tempat. Memutakhirkan cluster Amazon DocumentDB Anda menggunakan AWS Database Migration Service

Ada dua persyaratan konfigurasi untuk mendapatkan pembaruan patch terbaru untuk sebuah versi mesin pada suatu klaster:

- Status klaster harus tersedia.
- Klaster harus menjalankan versi mesin sebelumnya.

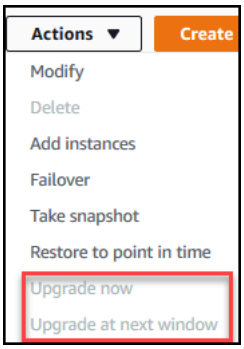
Using the Konsol Manajemen AWS

Prosedur berikut menerapkan pembaruan patch untuk versi mesin pada klaster dengan menggunakan konsol. Anda memiliki pilihan untuk memperbarui segera atau selama jendela pemeliharaan berikutnya pada klaster Anda.

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Pada panel navigasi, silakan pilih Klaster. Dalam daftar klaster, pilih tombol di sebelah kiri pada klaster yang ingin Anda tingkatkan. Status klaster harus tersedia.

> **Tip**
>
> Jika panel navigasi tidak terlihat di sisi kiri layar Anda, pilih ikon menu (≡) di sudut kiri atas halaman.

3. Dari menu Tindakan, pilih salah satu dari berikut ini. Opsi menu ini dapat dipilih hanya jika klaster yang Anda pilih tidak menjalankan versi mesin terbaru.



   - Tingkatkan sekarang—Segera memulai proses peningkatan. Klaster Anda akan offline untuk sementara waktu, sedangkan klaster ditingkatkan ke versi mesin terbaru.
   - Tingkatkan pada jendela berikutnya—Memulai proses peningkatan selama jendela pemeliharaan berikutnya pada klaster. Klaster Anda akan offline untuk sementara waktu, sedangkan klaster ditingkatkan ke versi mesin terbaru.

4. Jika jendela konfirmasi terbuka, pilih salah satu hal berikut:
   - Meningkatkan—Untuk meningkatkan klaster Anda ke versi mesin terbaru sesuai dengan jadwal yang dipilih pada langkah sebelumnya.
   - Batalkan—Untuk membatalkan peningkatan mesin klaster dan melanjutkan dengan versi mesin saat ini pada klaster.

Using the AWS CLI

Anda dapat menerapkan pembaruan tambalan ke cluster Anda menggunakan AWS CLI dan `apply-pending-maintenance-action` operasi dengan parameter berikut.

Parameter

- **`--resource-identifier`**—Diperlukan. ARN dari klaster Amazon DocumentDB yang akan Anda tingkatkan.

- **`--apply-action`**—Diperlukan. Nilai berikut diizinkan. Untuk meningkatkan versi mesin klaster Anda, gunakan `db-upgrade`.
  - **`db-upgrade`**
  - **`system-update`**
- **`--opt-in-type`**—Diperlukan. Nilai berikut diizinkan.
  - `immediate`—Terapkan tindakan pemeliharaan segera.
  - `next-maintenance`—Terapkan tindakan pemeliharaan selama jendela pemeliharaan berikutnya.
  - `undo-opt-in`—Membatalkan permintaan keikutsertaan `next-maintenance` yang ada.

Example

Patch contoh berikut memperbarui versi mesin `sample-cluster` ke versi 4.0.0.

Untuk Linux, macOS, atau Unix:

```
aws docdb apply-pending-maintenance-action \
   --resource-identifier arn:aws:rds:us-east-1:123456789012\:cluster:sample-cluster
 \
   --apply-action db-upgrade \
   --opt-in-type immediate
```

Untuk Windows:

```
aws docdb apply-pending-maintenance-action ^
   --resource-identifier arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster ^
   --apply-action db-upgrade ^
   --opt-in-type immediate
```

Output dari operasi ini terlihat seperti berikut ini.

```
{
    "ResourcePendingMaintenanceActions": {
        "ResourceIdentifier": "arn:aws:rds:us-
east-1:444455556666:cluster:docdb-2019-01-09-23-55-38",
        "PendingMaintenanceActionDetails": [
            {
                "CurrentApplyDate": "2019-02-20T20:57:06.904Z",
                "Description": "Bug fixes",
```

```
                "ForcedApplyDate": "2019-02-25T21:46:00Z",
                "OptInStatus": "immediate",
                "Action": "db-upgrade",
                "AutoAppliedAfterDate": "2019-02-25T07:41:00Z"
            }
        ]
    }
}
```

Menghentikan dan memulai cluster Amazon DocumentDB

Menghentikan dan memulai klaster Amazon DocumentDB membantu Anda mengelola biaya untuk pengembangan dan pengujian lingkungan. Alih-alih menciptakan dan menghapus klaster dan instans setiap kali Anda menggunakan Amazon DocumentDB, untuk sementara, Anda dapat menghentikan semua instans pada klaster ketika tidak diperlukan. Anda kemudian dapat memulainya lagi saat melanjutkan pengujian.

Topik

- Ikhtisar menghentikan dan memulai klaster
- Operasi yang dapat Anda lakukan pada cluster yang dihentikan

Ikhtisar menghentikan dan memulai klaster

Selama periode di mana Anda tidak memerlukan klaster Amazon DocumentDB, Anda dapat menghentikan semua instans dalam klaster itu sekaligus. Anda lalu dapat memulai klaster lagi kapan saja Anda memerlukannya. Memulai dan menghentikan menyederhanakan proses pengaturan dan perombakan pada klaster yang digunakan untuk pengembangan, pengujian, atau aktifitas serupa yang tidak memerlukan ketersediaan berkelanjutan. Anda dapat menghentikan dan memulai cluster menggunakan Konsol Manajemen AWS atau AWS CLI dengan satu tindakan, terlepas dari berapa banyak instance yang ada di cluster.

Sementara klaster Anda berhenti, volume penyimpanan klaster tetap tidak berubah. Anda hanya dikenakan biaya untuk penyimpanan, snapshot manual, dan penyimpanan cadangan otomatis dalam jendela penyimpanan yang Anda tentukan. Anda tidak dikenakan biaya untuk jam instans apa pun. Amazon DocumentDB secara otomatis memulai klaster Anda setelah tujuh hari sehingga tidak tertinggal pembaruan pemeliharaan yang diperlukan. Ketika klaster Anda dimulai setelah tujuh hari, Anda akan mulai dikenakan biaya untuk instans di klaster lagi. Sementara klaster Anda berhenti,

Anda tidak dapat mengajukan kueri untuk volume penyimpanan Anda karena kueri mengharuskan instans dalam keadaan tersedia.

Ketika sebuah klaster Amazon DocumentDB dihentikan, baik klaster maupun instans dapat dimodifikasi dengan cara apa pun. Ini termasuk menambahkan atau menghapus instans, atau menghapus klaster.

Using the Konsol Manajemen AWS

Prosedur berikut menunjukkan kepada Anda caranya menghentikan sebuah klaster dengan satu atau beberapa instans dalam keadaan tersedia, atau caranya memulai klaster yang telah dihentikan.

Untuk menghentikan atau memulai klaster Amazon DocumentDB

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Klaster.

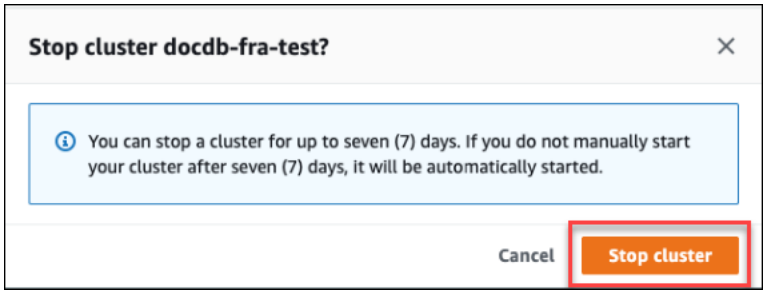
> **Tip**
>
> Jika panel navigasi tidak terlihat di sisi kiri layar Anda, pilih ikon menu (≡) di sudut kiri atas halaman.

3. Dalam daftar klaster, pilih tombol di sebelah kiri pada klaster yang ingin Anda mulai atau tingkatkan.
4. Pilih Tindakan, lalu pilih tindakan yang ingin Anda lakukan pada klaster.
   - Jika Anda ingin menghentikan klaster dan klaster tersedia:
     a. Pilih Berhenti.

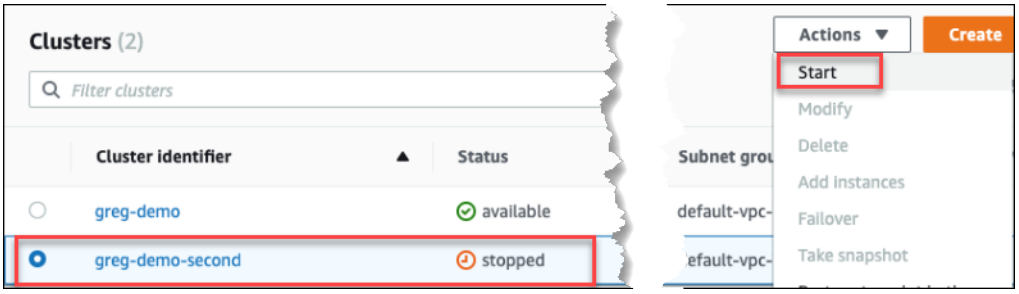


Untuk menghindari pengaktifan mekanisme failover, operasi stop menghentikan instans replika terlebih dahulu, lalu instans utama.

b. Pada dialog konfirmasi, konfirmasikan bahwa Anda ingin menghentikan klaster dengan memilih Hentikan klaster, atau untuk mempertahankan klaster agar terus berjalan, pilih Batalkan.



- Jika Anda ingin memulai klaster, dan klaster berhenti, pilih Mulai.



5. Pantau status klaster dan instansnya. Jika Anda memulai klaster, Anda dapat melanjutkan penggunaan klaster ketika klaster dan instansnya tersedia. Untuk informasi selengkapnya, lihat Menentukan status klaster.



Using the AWS CLI

Contoh kode berikut menunjukkan kepada Anda caranya menghentikan sebuah klaster dengan satu atau beberapa instans dalam keadaan tersedia, atau caranya memulai klaster yang telah dihentikan.

Untuk menghentikan cluster dengan satu atau lebih instance yang tersedia menggunakan AWS CLI, gunakan `stop-db-cluster` operasi. Untuk memulai klaster yang telah dihentikan, gunakan operasi `start-db-cluster`. Kedua operasi menggunakan parameter `--db-cluster-identifier`.

Parameter:

- **`--db-cluster-identifier`**—Diperlukan. Nama klaster untuk berhenti atau mulai.

Example— Untuk menghentikan cluster menggunakan AWS CLI

Kode berikut menghentikan klaster `sample-cluster`. Klaster harus memiliki satu atau beberapa instans dalam keadaan tersedia.

Untuk Linux, macOS, atau Unix:

```
aws docdb stop-db-cluster \
   --db-cluster-identifier sample-cluster
```

Untuk Windows:

```
aws docdb stop-db-cluster ^
   --db-cluster-identifier sample-cluster
```

Example— Untuk memulai cluster menggunakan AWS CLI

Kode berikut memulai klaster `sample-cluster`. Klaster saat ini harus dihentikan.

Untuk Linux, macOS, atau Unix:

```
aws docdb start-db-cluster \
   --db-cluster-identifier sample-cluster
```

Untuk Windows:

```
aws docdb start-db-cluster ^
   --db-cluster-identifier sample-cluster
```

Operasi yang dapat Anda lakukan pada cluster yang dihentikan

Saat klaster Amazon DocumentDB dihentikan, Anda dapat melakukan pemulihan titik waktu ke titik mana pun dengan jendela penyimpanan backup otomatis yang ditentukan. Untuk detail tentang melakukan pemulihan titik waktu, lihat Memulihkan ke titik waktu.

Anda tidak dapat memodifikasi konfigurasi klaster Amazon DocumentDB, atau salah satu instansnya saat klaster dihentikan. Anda juga tidak dapat menambahkan atau menghapus instans dari klaster, atau menghapus klaster tersebut jika masih memiliki instans terkait. Anda harus memulai klaster sebelum melakukan tindakan administratif tersebut.

Amazon DocumentDB menerapkan pemeliharaan terjadwal pada klaster yang dihentikan hanya setelah dimulai lagi. Setelah tujuh hari, Amazon DocumentDB secara otomatis memulai klaster yang telah dihentikan sehingga tidak jauh tertinggal dalam status pemeliharaan. Ketika klaster dinyalakan kembali, Anda akan mulai kembali dikenakan biaya untuk instans dalam klaster.

Sementara klaster dihentikan, Amazon DocumentDB tidak melakukan backup otomatis atau memperpanjang periode retensi cadangan.

Menghapus cluster Amazon DocumentDB

Anda dapat menghapus cluster Amazon DocumentDB menggunakan Konsol Manajemen AWS atau file. AWS CLI Untuk menghapus sebuah klaster, klaster harus dalam kondisi tersedia dan tidak harus memiliki instans apa pun yang terkait dengannya. Jika klaster berhenti, pertama mulai klaster, tunggu sampai klaster tersedia, kemudian hapus klaster tersebut. Untuk informasi selengkapnya, lihat Menghentikan dan memulai cluster Amazon DocumentDB.

Perlindungan penghapusan

Untuk melindungi klaster Anda dari penghapusan tidak disengaja, Anda dapat mengaktifkan perlindungan penghapusan. Perlindungan penghapusan diaktifkan secara default ketika Anda membuat klaster dengan menggunakan konsol. Namun, perlindungan penghapusan dinonaktifkan secara default jika Anda membuat klaster menggunakan AWS CLI.

Amazon DocumentDB memberlakukan perlindungan penghapusan untuk klaster, tidak soal apakah Anda melakukan operasi hapus dengan menggunakan konsol atau AWS CLI. Jika perlindungan penghapusan diaktifkan, Anda tidak dapat menghapus klaster. Untuk menghapus klaster yang mengaktifkan perlindungan penghapusan, Anda harus terlebih dahulu mengubah klaster dan menonaktifkan perlindungan penghapusan.

Saat menggunakan konsol dengan perlindungan penghapusan yang diaktifkan di sebuah klaster, Anda tidak dapat menghapus instans terakhir pada klaster karena hal itu juga akan menghapus klaster. Anda dapat menghapus instans terakhir dari klaster dengan perlindungan penghapusan menggunakan AWS CLI. Namun, klaster itu sendiri masih ada, dan data Anda disimpan. Anda dapat mengakses data dengan membuat instans baru untuk klaster. Untuk informasi selengkapnya tentang mengaktifkan dan menonaktifkan perlindungan penghapusan, lihat:

- Membuat cluster Amazon DocumentDB
- Memodifikasi cluster Amazon DocumentDB

Using the Konsol Manajemen AWS

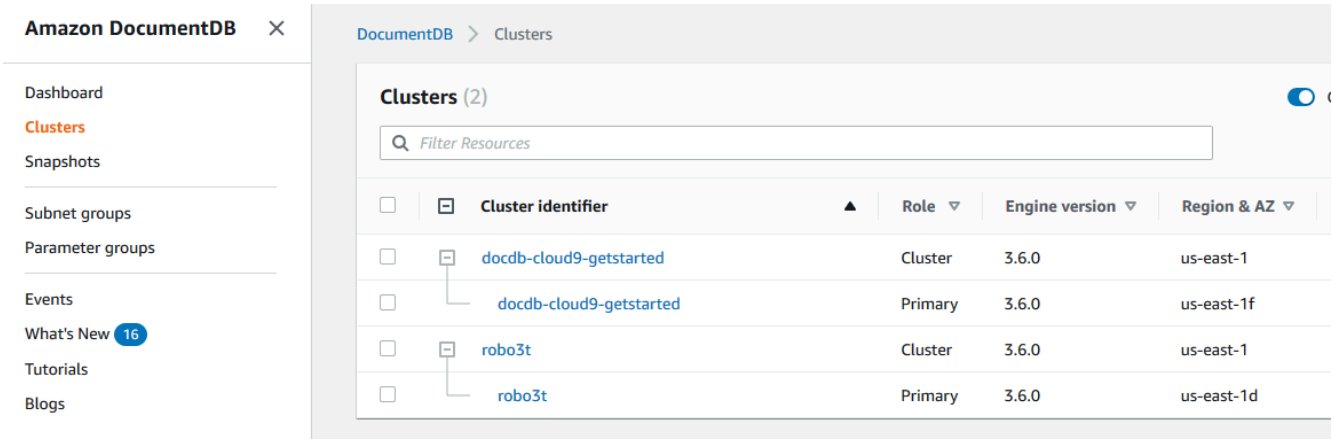
Untuk menghapus cluster menggunakan Konsol Manajemen AWS, perlindungan penghapusan harus dinonaktifkan.

Untuk menentukan apakah sebuah klaster memiliki perlindungan penghapusan yang diaktifkan:

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Klaster.

   > **Tip**
   >
   > Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu (≡) di pojok kiri atas halaman.

3. Perhatikan bahwa di kotak navigasi Cluster, kolom pengenal Cluster menunjukkan cluster dan instance. Contoh tercantum di bawah cluster, mirip dengan screenshot berikut.



4. Pilih nama klaster, dan pilih tab Konfigurasi. Di bagian Detail klaster, temukan Perlindungan penghapusan. Jika perlindungan penghapusan diaktifkan, modifikasi klaster untuk menonaktifkan perlindungan penghapusan. Untuk informasi tentang memodifikasi klaster, lihat Memodifikasi cluster Amazon DocumentDB.

   Setelah Perlindungan penghapusan dinonaktifkan, Anda siap untuk menghapus klaster.

Untuk menghapus klaster:

1. Pada panel navigasi, silakan pilih Klaster.
2. Tentukan apakah cluster memiliki instance dengan memeriksa kolom pengidentifikasi Cluster untuk contoh yang tercantum di bawahnya. Sebelum dapat menghapus klaster, Anda harus menghapus semua instans. Untuk informasi selengkapnya, lihat Menghapus instans Amazon DocumentDB.
3. Tergantung pada apakah klaster Anda memiliki instans, lakukan salah satu langkah berikut.
   - Jika klaster tidak memiliki instans, pilih tombol di sebelah kiri nama klaster dan pilih Tindakan. Dari menu pilihan menurun, pilih Hapus. Lengkapi kotak dialog Hapus <cluster-name>, lalu pilih Hapus.
   - Jika klaster memiliki satu atau beberapa instans, lakukan hal berikut:
     a. Pada panel navigasi, silakan pilih Klaster.
     b. Hapus setiap instance cluster dengan memilih kotak centang di sebelah kiri nama cluster. Pilih Tindakan, lalu pilih Hapus. Lengkapi kotak dialog Hapus <cluster-name>, lalu pilih Hapus.

Ketika Anda menghapus contoh terakhir, cluster juga akan dihapus. Untuk informasi selengkapnya tentang menghapus instance, lihat. Menghapus instans Amazon DocumentDB

Dibutuhkan beberapa menit untuk menghapus klaster. Untuk memantau status klaster, lihat Memantau status klaster Amazon DocumentDB.

Using the AWS CLI

Anda tidak dapat menghapus klaster yang memiliki instans yang berhubungan dengan itu. Untuk menentukan instans mana yang terkait dengan klaster Anda, jalankan perintah `describe-db-clusters` dan hapus semua instans klaster. Kemudian, jika diperlukan, nonaktifkan perlindungan penghapusan pada klaster Anda, dan terakhir, hapus klaster.

1. Pertama, hapus semua instance cluster.

   Untuk menentukan instans mana yang perlu Anda hapus, jalankan perintah berikut.

   ```
   aws docdb describe-db-clusters \
       --db-cluster-identifier sample-cluster \
       --query 'DBClusters[*].
   [DBClusterIdentifier,DBClusterMembers[*].DBInstanceIdentifier]'
   ```

   Output dari operasi ini terlihat seperti berikut (format JSON).

   ```
   [
       [
           "sample-cluster",
           [
               "sample-instance-1",
               "sample-instance-2"
           ]
       ]
   ]
   ```

   Jika klaster yang Anda ingin hapus memiliki suatu instans, hapuslah dengan cara yang ditunjukkan di bawah ini.

   ```
   aws docdb delete-db-instance \
   ```

```
    --db-instance-identifier sample-instance
```

2. Kedua, nonaktifkan perlindungan penghapusan.

   Menggunakan AWS CLI untuk menghapus semua instance cluster tidak menghapus cluster. Anda juga harus menghapus klaster, tetapi Anda dapat melakukan ini hanya jika perlindungan penghapusan dinonaktifkan.

   Untuk menentukan apakah klaster memiliki perlindungan penghapusan yang diaktifkan, jalankan perintah berikut.

   > **Tip**
   >
   > Untuk melihat status perlindungan penghapusan di seluruh klaster Amazon DocumentDB Anda, hilangkan parameter `--db-cluster-identifier`.

   ```
   aws docdb describe-db-clusters \
       --db-cluster-identifier sample-cluster \
       --query 'DBClusters[*].[DBClusterIdentifier,DeletionProtection]'
   ```

   Output dari operasi ini terlihat seperti berikut ini.

   ```
   [
       [
           "sample-cluster",
           "true"
       ]
   ]
   ```

   Jika klaster memiliki perlindungan penghapusan yang diaktifkan, modifikasi klaster dan nonaktifkan perlindungan penghapusan. Untuk menonaktifkan perlindungan penghapusan pada klaster, jalankan perintah berikut.

   ```
   aws docdb modify-db-cluster \
       --db-cluster-identifier sample-cluster \
       --no-deletion-protection \
       --apply-immediately
   ```

3. Akhirnya, hapus cluster.

   Setelah perlindungan penghapusan dinonaktifkan, Anda siap untuk menghapus klaster. Untuk menghapus klaster, gunakan operasi `delete-db-cluster` dengan parameter berikut.

   - **`--db-cluster-identifier`**—Diperlukan. Pengenal klaster yang ingin Anda hapus.
   - **`--final-db-snapshot-identifier`**—Opsional. Jika Anda ingin snapshot akhir, Anda harus menyertakan parameter ini dengan nama untuk snapshot akhir. Anda harus menyertakan `--final-db-snapshot-identifier` atau `--skip-final-snapshot`.

     Aturan penamaan:

     - Panjangnya adalah [1—63] huruf, angka, atau tanda hubung.
     - Karakter pertama harus berupa huruf.
     - Tidak dapat diakhiri dengan tanda hubung atau berisi dua tanda hubung berurutan.
     - Harus unik untuk semua cluster di Amazon RDS, Amazon Neptune, dan Amazon DocumentDB per, per Wilayah. Akun AWS
   - **`--skip-final-snapshot`**—Opsional. Gunakan parameter ini hanya jika Anda tidak ingin mengambil snapshot akhir sebelum menghapus klaster Anda. Pengaturan default adalah untuk mengambil snapshot akhir. Anda harus menyertakan `--final-db-snapshot-identifier` atau `--skip-final-snapshot`.

   AWS CLI Kode berikut menghapus cluster `sample-cluster` dengan snapshot akhir. Operasi gagal jika ada setiap instans yang terkait dengan klaster atau jika perlindungan penghapusan diaktifkan.

   Example

   Untuk Linux, macOS, atau Unix:

   ```
   aws docdb delete-db-cluster \
       --db-cluster-identifier sample-cluster \
       --final-db-snapshot-identifier sample-cluster-final-snapshot
   ```

   Untuk Windows:

```
aws docdb delete-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --final-db-snapshot-identifier sample-cluster-final-snapshot
```

Example

AWS CLI Kode berikut menghapus cluster `sample-cluster` tanpa mengambil snapshot akhir.

Untuk Linux, macOS, atau Unix:

```
aws docdb delete-db-cluster \
    --db-cluster-identifier sample-cluster \
    --skip-final-snapshot
```

Untuk Windows:

```
aws docdb delete-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --skip-final-snapshot
```

Output dari operasi `delete-db-cluster` adalah klaster Anda akan Anda hapus.

Dibutuhkan beberapa menit untuk menghapus klaster. Untuk memantau status klaster, lihat Memantau status klaster.

Menskalakan cluster Amazon DocumentDB

Amazon DocumentDB memungkinkan Anda untuk menskalakan penyimpanan dan melakukan komputasi dalam klaster Anda berdasarkan kebutuhan Anda. Bagian ini menjelaskan bagaimana Anda dapat menggunakan penskalaan untuk penyimpanan, penskalaan untuk instans, dan penskalaan untuk pembacaan guna mengelola performa dan penskalaan untuk klaster dan instans Amazon DocumentDB.

Topik

- Penskalaan penyimpanan
- Penskalaan instans

- Penskalaan baca
- Tulis penskalaan

Penskalaan penyimpanan

Penyimpanan Amazon DocumentDB secara otomatis diskalakan dengan data dalam volume klaster Anda. Seiring pertumbuhan data Anda, penyimpanan volume cluster Anda tumbuh dalam peningkatan 10 GiB, hingga 128 TiB.

Penskalaan instans

Anda dapat menskalakan klaster Amazon DocumentDB sesuai kebutuhan dengan memodifikasi kelas instans untuk setiap instans dalam klaster. Amazon DocumentDB mendukung beberapa kelas instans yang dioptimalkan untuk Amazon DocumentDB.

Untuk informasi selengkapnya, lihat Memodifikasi instance Amazon DocumentDB.

Penskalaan baca

Anda dapat mencapai penskalaan baca untuk klaster Amazon DocumentDB dengan membuat hingga 15 replika Amazon DocumentDB dalam klaster. Setiap replika Amazon DocumentDB menghasilkan data yang sama dari volume klaster dengan sedikit penundaan replika—biasanya kurang dari 100 milidetik setelah instans utama menulis pembaruan. Saat lalu lintas baca meningkat, Anda dapat membuat replika Amazon DocumentDB tambahan dan terhubung langsung ke sana untuk mendistribusikan beban baca pada klaster Anda. Replika Amazon DocumentDB tidak harus dari kelas instans DB yang sama dengan instans utama.

Untuk informasi selengkapnya, lihat Menambahkan instance Amazon DocumentDB ke klaster.

Untuk membaca penskalaan dengan Amazon DocumentDB, Anda sebaiknya membuat sambungan ke klaster Anda sebagai set replika dan mendistribusikan pembacaan ke instans replika menggunakan kemampuan preferensi baca bawaan driver Anda. Untuk informasi selengkapnya, lihat Menghubungkan ke Amazon DocumentDB sebagai set replika.

Tulis penskalaan

Anda dapat menskalakan kapasitas tulis pada klaster Amazon DocumentDB Anda dengan meningkatkan ukuran instans utama klaster Anda. Bagian ini menyediakan dua metode untuk menskalakan instans utama klaster Anda berdasarkan kebutuhan. Opsi pertama berupaya meminimalkan dampak aplikasi tetapi membutuhkan lebih banyak langkah untuk menyelesaikannya.

Opsi kedua mengoptimalkan kesederhanaan karena memiliki langkah yang lebih sedikit, tetapi dilengkapi dengan tradeoff yang memiliki dampak yang lebih potensial untuk aplikasi Anda.

Bergantung pada aplikasi Anda, Anda bisa memilih pendekatan apa di bawah ini yang terbaik untuk Anda. Untuk informasi selengkapnya tentang ukuran dan biaya instans yang tersedia, lihat halaman Harga Amazon DocumentDB.

1. Optimalkan ketersediaan dan performa tinggi — Jika Anda terhubung ke klaster Anda di mode set replika (disarankan), Anda dapat menggunakan proses berikut untuk meminimalkan dampak aplikasi Anda saat menskalakan instans primer Anda. Metode ini meminimalkan dampak karena mempertahankan klaster Anda pada atau di atas ketersediaan tinggi Anda, dan target penskalaan baca ditambahkan ke klaster sebagai instans, alih-alih diperbarui pada tempatnya.

   a. Tambahkan satu atau beberapa replika dari jenis instans yang lebih besar ke klaster Anda (lihat ???). Semua replika sebaiknya memiliki jenis instans yang sama dengan atau lebih besar dari yang utama. Hal ini untuk menghindari pengurangan yang tidak disengaja dalam performa tulis akibat kegagalan terhadap jenis instans yang lebih kecil. Untuk sebagian besar pelanggan, ini berarti penggandaan jumlah instans sementara di klaster mereka, kemudian penghapusan replika yang lebih kecil setelah penskalaan selesai.

   b. Atur tingkat failover pada semua replika baru ke prioritas nol, hal ini memastikan replika jenis instans yang lebih kecil memiliki prioritas failover tertinggi. Untuk informasi selengkapnya, lihat ???.

   c. Memulai failover manual, yang akan mempromosikan salah satu replika baru menjadi instans utama. Untuk informasi selengkapnya, lihat ???.

      > **Note**
      >
      > Ini akan menimbulkan ~30 detik waktu henti untuk klaster Anda. Rencanakan sesuai.

   d. Hapus semua replika dengan jenis instans yang lebih kecil daripada intans utama baru Anda dari klaster.

   e. Kembalikan tingkat failover semua intans ke prioritas yang sama (biasanya, ini berarti pengaturan kembali ke 1).

   Sebagai contoh, anggaplah Anda memiliki klaster yang saat ini berisi tiga instans `r5.large` (satu primer dan dua replika), dan Anda ingin melakukan penskalaan ke tipe instans

`r5.xlarge`. Untuk melakukannya, pertama Anda akan menambahkan tiga instans replika `r5.xlarge` ke klaster Anda, lalu tetapkan tingkat failover replika `r5.xlarge` baru ke nol. Selanjutnya, Anda akan memulai failover manual (diketahui bahwa aplikasi Anda akan mengalami ~ 30 detik waktu henti). Setelah failover selesai, Anda akan menghapus ketiga instans `r5.large` dari klaster Anda, sehingga klaster tersebut diskalakan ke instans `r5.xlarge`.

Untuk membantu mengoptimalkan biaya, instans Amazon DocumentDB ditagih dalam kenaikan satu detik, dengan biaya minimum sepuluh menit setelah perubahan status yang dapat ditagih seperti membuat, memodifikasi, atau menghapus instans. Untuk informasi selengkapnya lihat Optimalisasi biaya dalam dokumentasi praktik terbaik.

2. Optimalisasi untuk kesederhanaan — Pendekatan ini dioptimalkan untuk menyederhanakan prosesnya. Cara ini tidak memperluas dan menyempitkan klaster, tetapi mungkin mengurangi kapasitas baca Anda untuk sementara.

   Ada kemungkinan bahwa mengubah kelas instans replika akan menghasilkan instans yang tidak melayani permintaan untuk jangka waktu singkat, dari beberapa detik hingga kurang dari 30 detik. Jika Anda terhubung ke klaster Anda di mode set replika (direkomendasikan), maka ini akan mengurangi kapasitas baca Anda pada satu replika (misalnya, untuk 66% kapasitas dalam klaster 3-node, atau 75% kapasitas dalam klaster 4-node, dll) selama operasi penskalaan.

   a. Skalakan salah satu contoh instans di klaster Anda. Untuk informasi selengkapnya, lihat Mengelola kelas instance.

   b. Tunggu sampai instans tersedia (lihatMemantau status instans Amazon DocumentDB).

      > **Note**
      >
      > Ini akan menimbulkan ~30 detik waktu henti untuk klaster Anda. Rencanakan sesuai.

   c. Lanjutkan menjalankan langkah 1 dan 2 sampai semua instans replika telah diskalakan, satu per satu.

   d. Memulai failover manual. Ini akan mempromosikan salah satu replika menjadi instans utama. Untuk informasi selengkapnya, lihat Failover Amazon DocumentDB.

> **Note**
>
> Tindakan ini akan menyebabkan waktu henti hingga 30 detik untuk klaster Anda, tetapi seringkali membutuhkan waktu lebih singkat. Rencanakan sesuai.

e. Skalakan instans primer sebelumnya (sekarang replika).

Mengkloning volume untuk cluster Amazon DocumentDB

Dengan menggunakan kloning Amazon DocumentDB, Anda dapat membuat cluster baru yang menggunakan volume cluster Amazon DocumentDB yang sama dan memiliki data yang sama dengan aslinya. Prosesnya dirancang agar cepat dan hemat biaya. Klaster baru dengan volume data terkaitnya dikenal sebagai Klon. Membuat klon lebih cepat dan lebih hemat ruang daripada menyalin data secara fisik menggunakan teknik lain, seperti memulihkan snapshot.

Amazon DocumentDB mendukung pembuatan klon yang disediakan Amazon DocumentDB dari klaster Amazon DocumentDB yang disediakan. Saat Anda membuat klon menggunakan konfigurasi penyebaran yang berbeda dari sumbernya, klon dibuat menggunakan versi terbaru dari mesin Amazon DocumentDB sumber.

Saat Anda membuat klon dari kluster Amazon DocumentDB Anda, klon dibuat AWS di akun Anda—akun yang sama yang memiliki cluster Amazon DocumentDB sumber.

Topik

- Ikhtisar kloning Amazon DocumentDB
- Keterbatasan kloning Amazon DocumentDB
- Cara kerja kloning Amazon DocumentDB
- Membuat klon Amazon DocumentDB

Ikhtisar kloning Amazon DocumentDB

Amazon DocumentDB menggunakan protokol copy-on-write untuk membuat klon. Mekanisme ini menggunakan ruang tambahan minimal untuk membuat klon awal. Saat klon pertama kali dibuat, Amazon DocumentDB menyimpan satu salinan data yang digunakan oleh cluster DB sumber dan cluster Amazon DocumentDB baru (kloning). Penyimpanan tambahan dialokasikan hanya ketika perubahan dilakukan pada data (pada volume penyimpanan Amazon DocumentDB) oleh

cluster Amazon DocumentDB sumber atau klon klon Amazon DocumentDB. Untuk mempelajari selengkapnya tentang protokol copy-on-write, lihat Cara kerja kloning Amazon DocumentDB.

Kloning Amazon DocumentDB sangat berguna untuk menyiapkan lingkungan pengujian dengan cepat menggunakan data produksi Anda, tanpa risiko kerusakan data. Anda dapat menggunakan klon untuk berbagai jenis aplikasi, seperti berikut:

- Bereksperimen dengan potensi perubahan (misalnya, perubahan skema dan perubahan grup parameter) untuk menilai semua dampak.
- Menjalankan operasi sarat beban kerja, seperti mengekspor data atau menjalankan kueri analitis.
- Membuat salinan dari klaster DB produksi Anda untuk pengembangan, pengujian, atau tujuan lainnya.

Anda dapat membuat lebih dari satu klon dari cluster Amazon DocumentDB yang sama. Anda juga dapat membuat beberapa klon dari klon lain.

Setelah membuat klon Amazon DocumentDB, Anda dapat mengonfigurasi instans Amazon DocumentDB secara berbeda dari cluster Amazon DocumentDB sumber. Misalnya, Anda mungkin tidak memerlukan klon untuk tujuan pengembangan agar memenuhi persyaratan ketersediaan tinggi yang sama dengan klaster Amazon DocumentDB produksi sumber. Dalam hal ini, Anda dapat mengonfigurasi klon dengan satu instans Amazon DocumentDB daripada beberapa instans DB yang digunakan oleh cluster Amazon DocumentDB.

Setelah selesai menggunakan klon untuk pengujian, pengembangan, atau tujuan lainnya, Anda dapat menghapusnya.

Keterbatasan kloning Amazon DocumentDB

Amazon DocumentDB; kloning saat ini memiliki batasan sebagai berikut:

- Anda dapat membuat klon sebanyak yang Anda inginkan, hingga jumlah maksimum klaster DB yang diizinkan di Wilayah AWS. Namun, setelah Anda membuat 15 klon, klon berikutnya adalah salinan lengkap. Operasi kloning bertindak seperti pemulihan point-in-time.
- Anda tidak dapat membuat klon di AWS Wilayah yang berbeda dari cluster Amazon DocumentDB sumber.
- Anda tidak dapat membuat klon dari cluster Amazon DocumentDB yang tidak memiliki instans DB. Anda hanya dapat mengkloning cluster Amazon DocumentDB yang memiliki setidaknya satu instans DB.

- Anda dapat membuat klon di cloud pribadi virtual (VPC) yang berbeda dari klaster Amazon DocumentDB. Jika Anda melakukannya, subnet VPC harus dipetakan ke Zona Ketersediaan yang sama.

Cara kerja kloning Amazon DocumentDB

Kloning Amazon DocumentDB bekerja pada lapisan penyimpanan cluster Amazon DocumentDB. Ini menggunakan protokol copy-on-write yang cepat dan hemat ruang dalam hal media tahan lama yang mendasarinya yang mendukung volume penyimpanan Amazon DocumentDB. Anda dapat mempelajari lebih lanjut tentang volume klaster Amazon DocumentDB di. Mengelola cluster Amazon DocumentDB

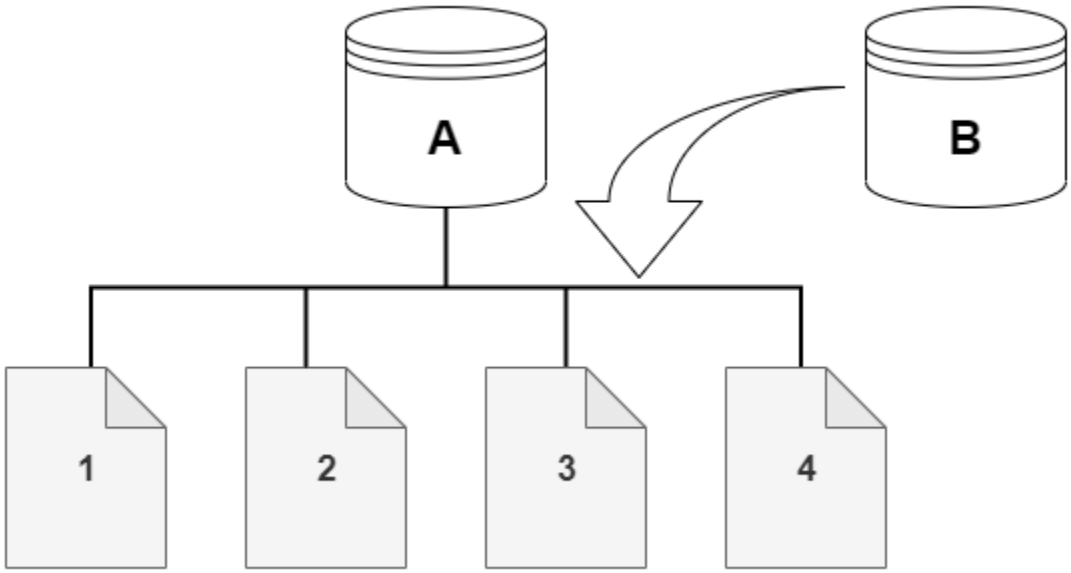
Topik

- Memahami protokol copy-on-write
- Menghapus volume klaster sumber

Memahami protokol copy-on-write

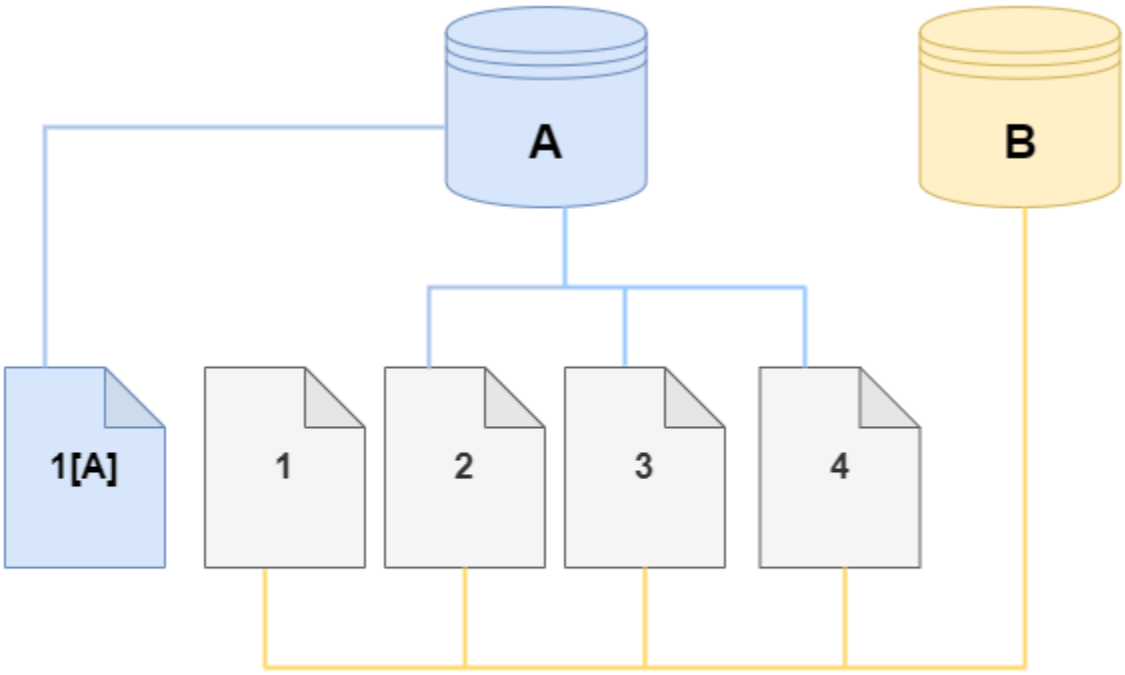
Cluster Amazon DocumentDB menyimpan data di halaman dalam volume penyimpanan Amazon DocumentDB yang mendasarinya.

Misalnya, dalam diagram berikut Anda dapat menemukan cluster Amazon DocumentDB (A) yang memiliki empat halaman data, 1, 2, 3, dan 4. Bayangkan bahwa klon, B, dibuat dari cluster Amazon DocumentDB. Saat klon dibuat, tidak ada data yang disalin. Sebaliknya, klon menunjuk ke kumpulan halaman yang sama dengan cluster Amazon DocumentDB sumber.

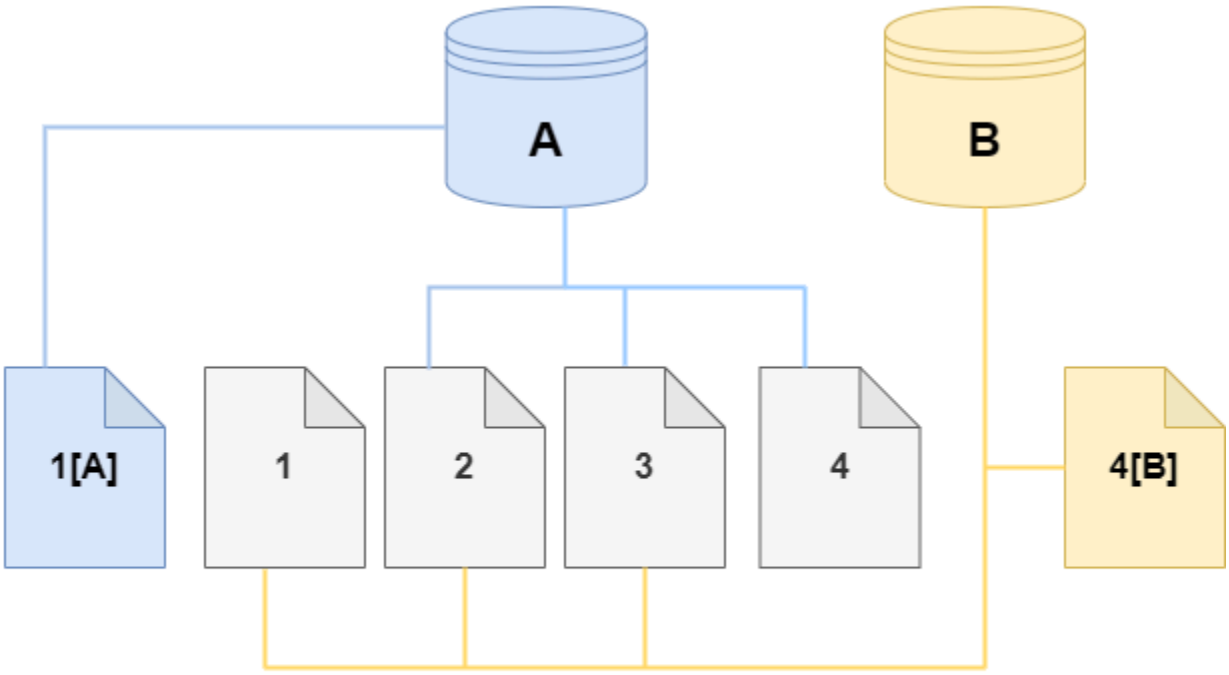


Saat klon dibuat, tidak ada penyimpanan tambahan yang biasanya diperlukan. Protokol copy-on-write menggunakan segmen yang sama pada media penyimpanan fisik sebagai segmen sumber. Penyimpanan tambahan hanya diperlukan jika kapasitas segmen sumber tidak cukup untuk seluruh segmen klon. Jika demikian, segmen sumber disalin ke perangkat fisik lain.

Dalam diagram berikut, Anda dapat menemukan contoh protokol copy-on-write dalam praktiknya menggunakan klaster A yang sama dan klonnya, B, seperti yang ditunjukkan di atas. Katakanlah Anda membuat perubahan pada klaster Amazon DocumentDB (A) yang menghasilkan perubahan pada data yang disimpan di halaman 1. Alih-alih menulis ke halaman asli 1, Amazon DocumentDB membuat halaman baru 1 [A]. Volume cluster Amazon DocumentDB untuk cluster (A) sekarang menunjuk ke halaman 1 [A], 2, 3, dan 4, sedangkan klon (B) masih mereferensikan halaman asli.



Pada klon, perubahan dibuat pada halaman 4 di volume penyimpanan. Alih-alih menulis ke halaman asli 4, Amazon DocumentDB membuat halaman baru, 4 [B]. Klon sekarang menunjuk ke halaman 1, 2, 3, dan halaman 4[B], sementara klaster (A) terus menunjuk ke 1[A], 2, 3, dan 4.



Karena semakin banyak perubahan terjadi dari waktu ke waktu di volume cluster Amazon DocumentDB sumber dan klon, lebih banyak penyimpanan diperlukan untuk menangkap dan menyimpan perubahan.

Menghapus volume klaster sumber

Saat Anda menghapus volume klaster sumber yang memiliki satu atau beberapa klon yang berkaitan dengannya, klon ini tidak akan terpengaruh. Klon terus menunjuk ke halaman yang sebelumnya dimiliki oleh volume klaster sumber.

Membuat klon Amazon DocumentDB

Anda dapat membuat klon di AWS akun yang sama dengan cluster Amazon DocumentDB sumber. Untuk melakukannya, Anda dapat menggunakan Konsol Manajemen AWS atau AWS CLI dan prosedur berikut.

Dengan menggunakan kloning Amazon DocumentDB, Anda dapat membuat klon klon Amazon DocumentDB yang disediakan dari klon Amazon DocumentDB yang disediakan.

Using the Konsol Manajemen AWS

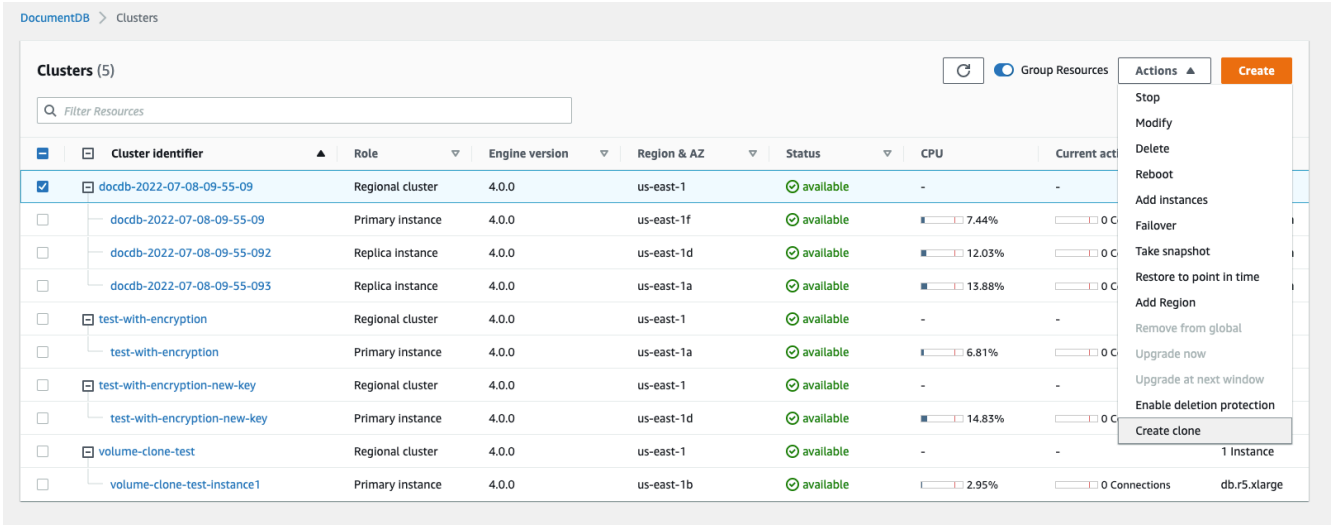
Prosedur berikut menjelaskan cara mengkloning cluster Amazon DocumentDB menggunakan. Konsol Manajemen AWS

Membuat klon menggunakan Konsol Manajemen AWS hasil di cluster Amazon DocumentDB dengan satu instance Amazon DocumentDB.

Instruksi ini berlaku untuk cluster DB yang dimiliki oleh AWS akun yang sama yang membuat klon. Cluster DB harus dimiliki oleh AWS akun yang sama karena kloning lintas akun tidak didukung di Amazon DocumentDB.

Untuk membuat tiruan dari cluster DB yang dimiliki oleh Anda AWS akun menggunakan Konsol Manajemen AWS

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Pada panel navigasi, silakan pilih Klaster.
3. Pilih klaster Amazon DocumentDB Anda dari daftar, dan untuk Tindakan, pilih Buat klon.

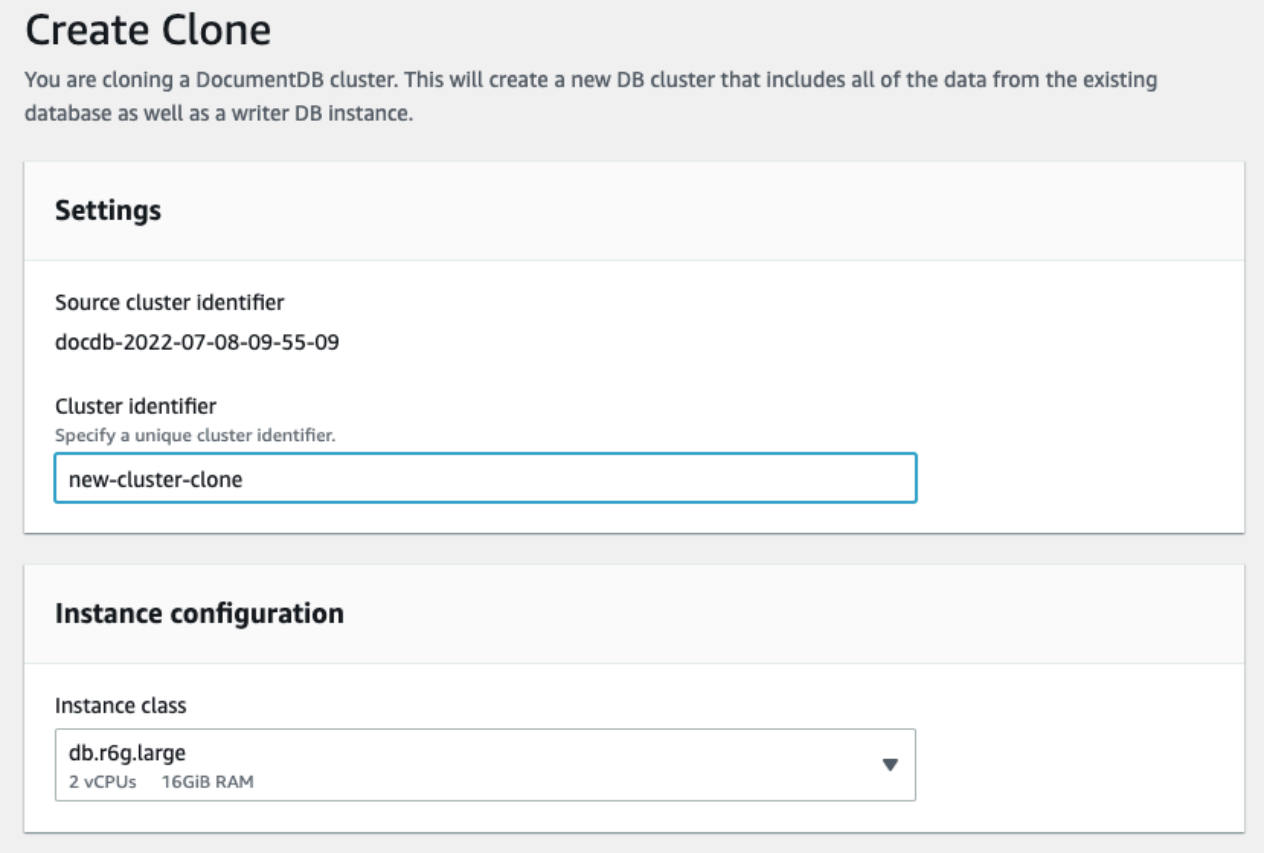


Halaman Create clone terbuka, di mana Anda dapat mengonfigurasi pengenal Cluster dan kelas Instance, dan opsi lain untuk klon klon Amazon DocumentDB.

4. Di bagian Pengaturan, lakukan hal berikut:

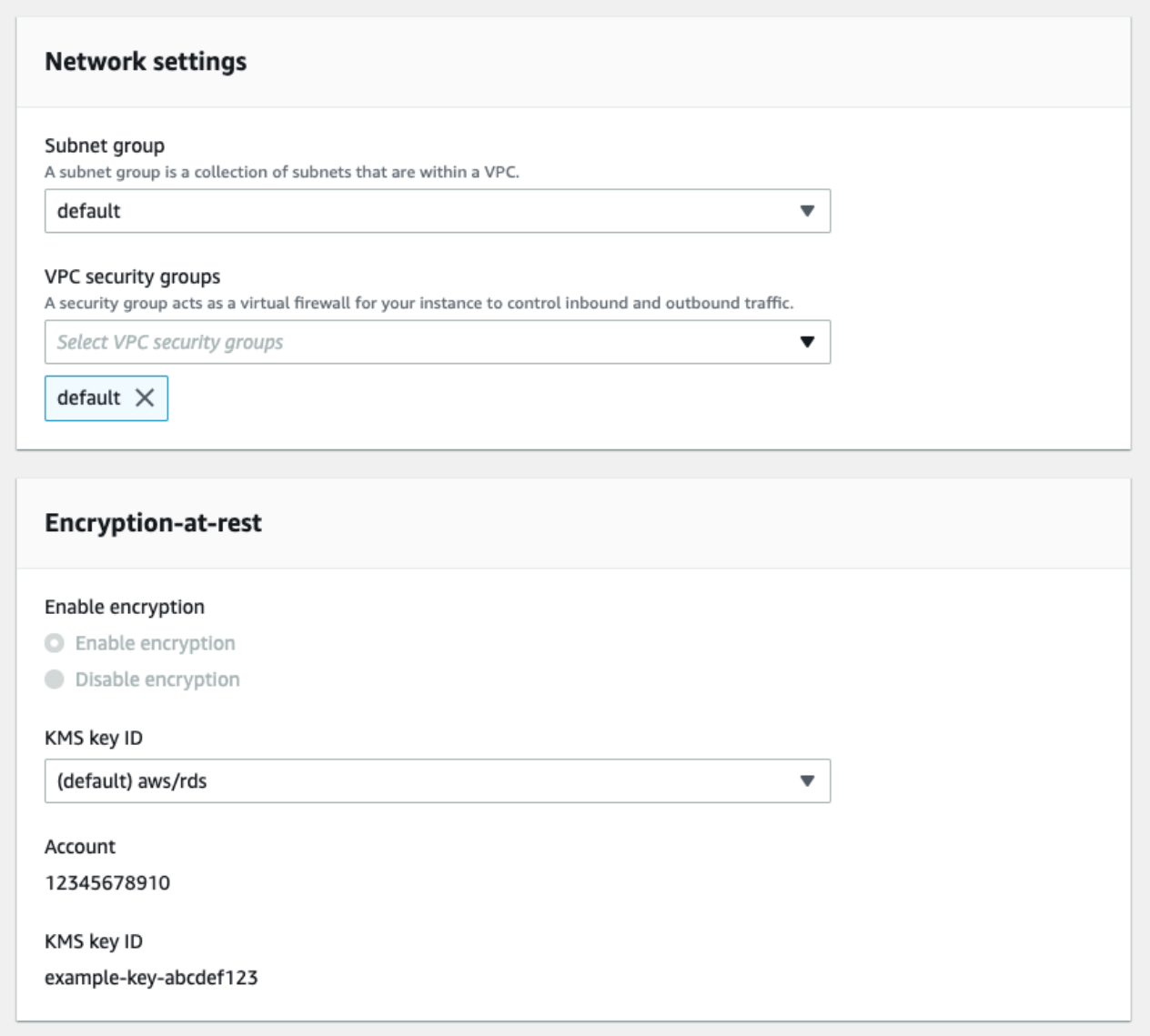
   a. Untuk pengidentifikasi Cluster, masukkan nama yang ingin Anda berikan ke klaster Amazon DocumentDB kloning Anda.

b. Untuk konfigurasi Instance, pilih class Instance yang sesuai untuk klaster Amazon DocumentDB kloning Anda.

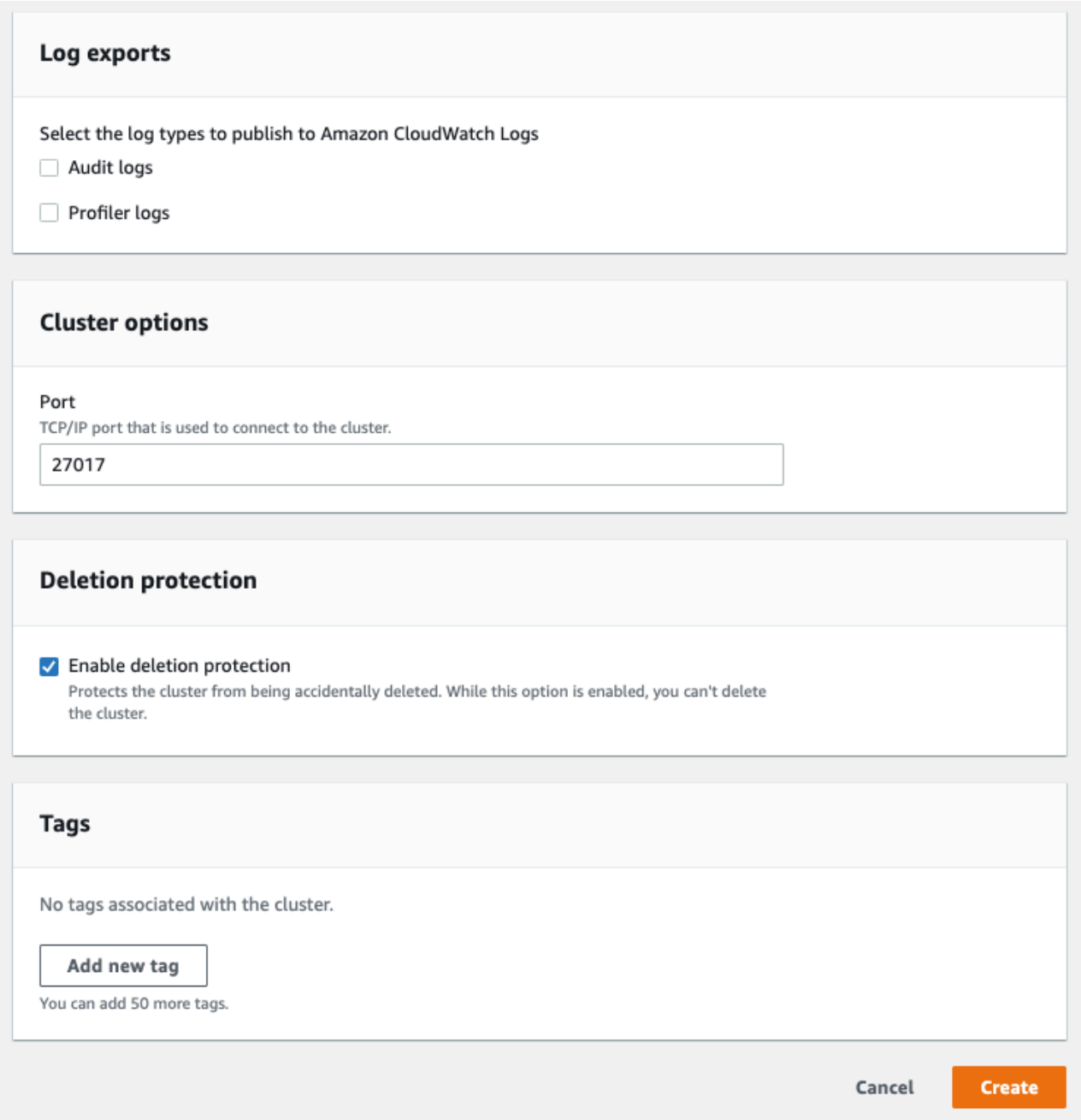


c. Untuk pengaturan Jaringan, pilih grup Subnet untuk kasus penggunaan Anda dan grup keamanan VPC terkait.

d. Sebab Encryption-at-rest, jika cluster sumber (cluster yang sedang dikloning) mengaktifkan enkripsi, klaster kloning juga harus mengaktifkan enkripsi. Jika skenario ini benar, maka opsi Enable encryption berwarna abu-abu (dinonaktifkan) tetapi dengan pilihan Enable encryption yang dipilih. Sebaliknya, jika cluster sumber tidak mengaktifkan enkripsi, opsi Enable encryption tersedia dan Anda dapat memilih untuk mengaktifkan atau menonaktifkan enkripsi.



e. Selesaikan konfigurasi klon cluster baru dengan memilih jenis log yang akan diekspor (opsional), memasukkan port tertentu yang digunakan untuk terhubung ke cluster, dan mengaktifkan perlindungan dari penghapusan cluster secara tidak sengaja (diaktifkan secara default).



f. Selesai memasukkan semua pengaturan untuk klon klon Amazon DocumentDB Anda. Untuk mempelajari selengkapnya tentang setelan cluster dan instans Amazon DocumentDB, lihat. Mengelola cluster Amazon DocumentDB

5. Pilih Buat klon untuk meluncurkan klon Amazon DocumentDB dari klaster Amazon DocumentDB pilihan Anda.

Saat klon dibuat, klon tersebut terdaftar dengan kluster Amazon DocumentDB Anda yang lain di bagian Database konsol dan menampilkan statusnya saat ini. Klon Anda siap digunakan ketika statusnya Tersedia.

Using the AWS CLI

Menggunakan AWS CLI untuk mengkloning cluster Amazon DocumentDB Anda melibatkan beberapa langkah.

`restore-db-cluster-to-point-in-time` AWS CLI Perintah yang Anda gunakan menghasilkan klaster Amazon DocumentDB kosong dengan 0 instance Amazon DocumentDB. Artinya, perintah hanya mengembalikan cluster Amazon DocumentDB, bukan instance DB untuk cluster itu. Anda melakukannya secara terpisah setelah klon tersedia. Dua langkah dalam proses ini adalah sebagai berikut:

1. Buat klon dengan menggunakan perintah CLI restore-db-cluster-to-point-in-time. Parameter yang Anda gunakan dengan perintah ini mengontrol jenis kapasitas dan detail lain dari klaster Amazon DocumentDB kosong (klon) yang sedang dibuat.
2. Buat instance Amazon DocumentDB untuk klon dengan menggunakan perintah CLI create-db-instance untuk membuat ulang instans Amazon DocumentDB di klaster Amazon DocumentDB yang dipulihkan.

Perintah berikut mengasumsikan bahwa AWS CLI diatur dengan AWS Wilayah Anda sebagai default. Dengan pendekatan ini, Anda tidak perlu meneruskan nama `--region` di setiap perintah. Untuk informasi selengkapnya, lihat Mengonfigurasi AWS CLI. Anda juga dapat menentukan `--region` di setiap perintah CLI selanjutnya.

Membuat klon

Parameter tertentu yang Anda teruskan ke perintah CLI `restore-db-cluster-to-point-in-time` akan bervariasi. Apa yang Anda lewati tergantung pada jenis klon yang ingin Anda buat.

Gunakan prosedur berikut untuk membuat klon Amazon DocumentDB yang disediakan dari klaster Amazon DocumentDB yang disediakan.

Untuk membuat tiruan dari mode mesin yang sama dengan sumber cluster Amazon DocumentDB

- Gunakan perintah CLI `restore-db-cluster-to-point-in-time` dan tentukan nilai untuk parameter berikut:

- `--db-cluster-identifier` – Pilih nama yang bermakna untuk klon anda. Anda menamai klon ketika Anda menggunakan perintah CLI restore-db-cluster-to-point-in-time.
- `--restore-type` – Gunakan `copy-on-write` untuk membuat klon dari klaster DB sumber. Tanpa parameter ini, `restore-db-cluster-to-point-in-time` mengembalikan cluster Amazon DocumentDB daripada membuat klon. Default untuk `restore-type` adalah`full-copy`.
- `--source-db-cluster-identifier`— Gunakan nama cluster Amazon DocumentDB sumber yang ingin Anda kloning.
- `--use-latest-restorable-time` – Nilai ini menunjuk ke data volume terbaru yang dapat dipulihkan untuk klon. Parameter ini diperlukan untuk`restore-type copy-on-write`, bagaimanapun, Anda tidak dapat menggunakan `restore-to-time parameter` dengan itu.

Contoh berikut membuat klon bernama `my-clone` dari klaster bernama `my-source-cluster`.

Untuk Linux, macOS, atau 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
```

Untuk 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
```

Perintah tersebut mengembalikan objek JSON yang berisi rincian dari klon. Periksa untuk memastikan bahwa klaster DB yang Anda kloning tersedia sebelum mencoba membuat instans DB untuk klon Anda. Untuk informasi selengkapnya, lihat Memeriksa status dan mendapatkan detail klon di bawah ini:

Memeriksa status dan mendapatkan detail klon

Anda dapat menggunakan perintah berikut untuk memeriksa status klaster DB kosong yang baru dibuat.

```
$ aws docdb describe-db-clusters --db-cluster-identifier my-clone --query '*[].
[Status]' --output text
```

Atau Anda dapat memperoleh status dan nilai-nilai lain yang Anda butuhkan untuk membuat instance DB untuk klon Anda dengan menggunakan AWS CLI kueri berikut:

Untuk Linux, macOS, atau Unix:

```
aws docdb describe-db-clusters --db-cluster-identifier my-clone \
  --query '*[].{Status:Status,Engine:Engine,EngineVersion:EngineVersion}'
```

Untuk Windows:

```
aws docdb describe-db-clusters --db-cluster-identifier my-clone ^
  --query "*[].{Status:Status,Engine:Engine,EngineVersion:EngineVersion}"
```

Kueri ini menghasilkan output yang serupa dengan berikut ini:

```
[
  {
        "Status": "available",
        "Engine": "docdb",
        "EngineVersion": "4.0.0",
    }
]
```

Membuat instance Amazon DocumentDB untuk klon Anda

Panggil perintah CLI create-db-instance untuk membuat instans DB untuk klon Anda.

`--db-instance-class`Parameter ini digunakan hanya untuk klaster Amazon DocumentDB yang disediakan.

Untuk Linux, macOS, atau 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
```

Untuk Windows:

```
aws docdb create-db-instance ^
    --db-instance-identifier my-new-db ^
    --db-cluster-identifier my-clone ^
    --db-instance-class  db.r5.4xlarge ^
    --engine docdb
```

Parameter yang digunakan untuk kloning

Tabel berikut merangkum berbagai parameter yang digunakan `restore-db-cluster-to-point-in-time` untuk mengkloning cluster Amazon DocumentDB.

Parameter	Deskripsi
--source-db-cluster-identifier	Gunakan nama cluster Amazon DocumentDB sumber yang ingin Anda kloning.
--db-cluster-identifier	Gunakan parameter ini untuk memilih nama yang bermakna untuk klon anda. Anda menamai klon Anda dengan perintah `restore-db-cluster-to-point-in-time` . Kemudian, Anda meneruskan nama ini ke perintah `create-db-instance` .
--restore-type	Tentukan `copy-on-write` sebagai `--restore-type` untuk membuat klon dari cluster DB sumber daripada memulihkan sumber Amazon DocumentDB cluster.
--use-latest-restorable-time	Nilai ini menunjuk ke data volume terbaru yang dapat dipulihkan untuk klon.

Memahami toleransi kesalahan klaster Amazon DocumentDB

Klaster Amazon DocumentDB menurut desainnya toleran terhadap kesalahan. Setiap volume klaster mencakup beberapa Availability Zone dalam satu Wilayah AWS, dan setiap Availability Zone berisi salinan data volume cluster. Dengan fungsionalitas ini, klaster Anda dapat menoleransi kegagalan Availability Zone tanpa kehilangan data dan hanya gangguan layanan singkat.

Jika instans utama dalam sebuah klaster gagal, Amazon DocumentDB secara otomatis melakukan failover ke instans utama baru dengan salah satu dari dua cara berikut:

- Dengan mempromosikan replika Amazon DocumentDB yang ada ke instans utama baru yang dipilih berdasarkan pengaturan Tingkat Promosi dari setiap replika, lalu membuat pengganti untuk instans primer sebelumnya. Failover untuk instans replika biasanya membutuhkan waktu kurang dari 30 detik. Operasi baca dan tulis mungkin mengalami gangguan singkat selama periode ini. Untuk meningkatkan ketersediaan klaster Anda, kami sarankan Anda membuat setidaknya satu replika Amazon DocumentDB atau lebih di dua Availability Zone yang berbeda.
- Dengan membuat instans primer baru. Ini hanya terjadi jika Anda tidak memiliki instans replika di klaster Anda dan penyelesaiannya dapat memakan waktu beberapa menit.

Jika klaster memiliki satu atau beberapa replika Amazon DocumentDB, replika Amazon DocumentDB akan dipromosikan ke instans primer selama peristiwa kegagalan. Peristiwa failover mengakibatkan interupsi singkat, selama operasi baca dan tulis gagal dengan pengecualian. Namun, layanan biasanya dipulihkan dalam waktu kurang dari 120 detik, dan sering kali kurang dari 60 detik. Untuk meningkatkan ketersediaan klaster Anda, kami sarankan Anda membuat setidaknya satu replika Amazon DocumentDB atau lebih di dua Availability Zone yang berbeda.

Anda dapat membuat urutan tertentu agar replika Amazon DocumentDB Anda dipromosikan ke instans primer setelah kegagalan dengan menetapkan prioritas pada setiap replika. Prioritas berkisar dari 0 untuk prioritas tertinggi hingga 15 untuk prioritas terendah. Jika instans primer gagal, replika Amazon DocumentDB dengan prioritas tertinggi akan dipromosikan ke instans primer baru. Anda dapat mengubah prioritas replika Amazon DocumentDB kapan saja. Memodifikasi prioritas tidak memicu failover. Anda dapat menggunakan operasi `modify-db-instance` dengan parameter `--promotion-tier`. Untuk informasi lebih lanjut tentang menyesuaikan prioritas failover instans, lihat Failover Amazon DocumentDB.

Lebih dari satu replika Amazon DocumentDB dapat berbagi prioritas yang sama, dan menghasilkan tingkat promosi. Jika dua replika Amazon DocumentDB atau lebih berbagi prioritas yang sama, maka replika dengan ukuran terbesar akan dipromosikan menjadi yang primer. Jika dua replika Amazon DocumentDB atau lebih memiliki prioritas dan ukuran yang sama, replika arbitrer di tingkat promosi yang sama akan dipromosikan.

Jika klaster tidak berisi replika Amazon DocumentDB, instans primer akan dibuat ulang selama peristiwa kegagalan. Peristiwa failover mengakibatkan gangguan di mana selama operasi baca dan tulis gagal dengan pengecualian. Layanan dipulihkan ketika instans primer baru dibuat, yang

biasanya memakan waktu kurang dari 10 menit. Mempromosikan replika Amazon DocumentDB ke instans utama jauh lebih cepat daripada membuat instans utama baru.

Mengelola instans Amazon DocumentDB

Topik berikut memberikan informasi untuk membantu Anda mengelola instans Amazon DocumentDB Anda. Topik-topik tersebut menyertakan detail tentang kelas dan status instans, serta cara membuat, menghapus, dan memodifikasi instans.

Topik

- Menentukan status instans
- Siklus hidup instance Amazon DocumentDB
- Mengelola kelas instance
- NVMe-backed contoh

Menentukan status instans

Untuk melihat status instans yang valid, artinya, dan cara menentukan status instans Anda, lihat Memantau status instans Amazon DocumentDB.

Siklus hidup instance Amazon DocumentDB

Siklus hidup dari instans Amazon DocumentDB termasuk menciptakan, memodifikasi, memelihara dan meningkatkan, melakukan backup dan pemulihan, mem-boot ulang, dan menghapus instans. Bagian ini menyediakan informasi tentang cara menyelesaikan proses ini.

Topik

- Menambahkan instance Amazon DocumentDB ke klaster
- Menjelaskan instans Amazon DocumentDB
- Memodifikasi instance Amazon DocumentDB
- Mem-boot ulang instans Amazon DocumentDB
- Menghapus instans Amazon DocumentDB

Anda dapat membuat instance Amazon DocumentDB baru menggunakan Konsol Manajemen AWS atau. AWS CLI Untuk menambahkan instans ke klaster, klaster harus dalam status tersedia. Anda

tidak dapat menambahkan instans ke klaster yang dihentikan. Jika klaster berhenti, pertama mulai klaster, tunggu sampai klaster tersedia, kemudian tambahkan instans. Untuk informasi selengkapnya, lihat Menghentikan dan memulai cluster Amazon DocumentDB.

> **Note**
>
> Jika Anda membuat klaster Amazon DocumentDB menggunakan konsol, sebuah instans secara otomatis dibuat untuk Anda pada saat yang sama. Jika Anda ingin membuat instans tambahan, gunakan salah satu prosedur berikut.

Menambahkan instance Amazon DocumentDB ke klaster

Using the Konsol Manajemen AWS

Gunakan prosedur berikut untuk membuat instans untuk klaster Anda menggunakan konsol Amazon DocumentDB.

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Klaster.

   > **Tip**
   >
   > Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu (≡) di pojok kiri atas halaman.

3. Untuk memilih klaster yang ingin Anda tambahi instans, pilih tombol di sebelah kiri nama klaster.
4. Pilih Tindakan, dan kemudian pilih Tambahkan instans.
5. Di halaman Tambahkan instans ke: **<cluster-name>**, ulangi langkah berikut untuk setiap instans yang ingin Anda tambahkan ke klaster. Anda dapat memiliki hingga 15.

   a. Pengidentifikasi instans — Anda dapat memasukkan pengidentifikasi unik untuk instans ini atau mengizinkan Amazon DocumentDB menyediakan pengidentifikasi instans berdasarkan pengidentifikasi klaster.

      Kendala penamaan instans:

- Panjangnya adalah [1—63] huruf, angka, atau tanda hubung.
- Karakter pertama harus berupa huruf.
- Tidak dapat diakhiri dengan tanda hubung atau berisi dua tanda hubung berurutan.
- Harus unik untuk semua instans di Amazon RDS, Neptunus, dan Amazon DocumentDB per, per Wilayah. Akun AWS

b. Kelas instans — Dari daftar tarik-turun, pilih tipe instans yang Anda inginkan untuk instans ini.

c. Tingkat promosi — Dari daftar tarik-turun, pilih tingkat promosi untuk instans Anda atau pilih Tidak ada preferensi guna mengizinkan Amazon DocumentDB untuk mengatur tingkat promosi untuk instans Anda. Angka yang lebih rendah berarti prioritas yang lebih tinggi. Untuk informasi selengkapnya, lihat Mengontrol target failover.

d. Untuk menambahkan lebih banyak instans, pilih Tambahkan instans tambahan dan ulangi langkah a, b, dan c.

6. Selesaikan operasi.

- Untuk menambahkan instans ke klaster Anda, pilih Buat.
- Untuk membatalkan operasi, pilih Batal.

Dibutuhkan beberapa menit untuk membuat sebuah instans. Anda dapat menggunakan konsol atau AWS CLI untuk melihat status instans. Untuk informasi selengkapnya, lihat Memantau status instans.

Using the AWS CLI

Gunakan `create-db-instance` AWS CLI operasi dengan parameter berikut untuk membuat instance utama untuk cluster Anda.

- **`--db-instance-class`** — Diperlukan. Kapasitas komputasi dan memori instans, misalnya, `db.m4.large`. Tidak semua kelas instans tersedia di semua Wilayah AWS.
- **`--db-instance-identifier`** — Diperlukan. Sebuah string yang mengidentifikasi instans.

  Kendala Penamaan Instans:

  - Panjangnya adalah [1—63] huruf, angka, atau tanda hubung.
  - Karakter pertama harus berupa huruf.
  - Tidak dapat diakhiri dengan tanda hubung atau berisi dua tanda hubung berurutan.

  - Harus unik untuk semua instans di Amazon RDS, Neptunus, dan Amazon DocumentDB per, per Wilayah. Akun AWS
- **`--engine`** — Diperlukan. Harus berupa `docdb`.
- **`--availability-zone`** — Opsional. Availability Zone tempat Anda ingin membuat instans ini. Gunakan parameter ini untuk menemukan instans Anda di Availability Zone yang berbeda untuk meningkatkan toleransi kesalahan. Untuk informasi selengkapnya, lihat Amazon DocumentDB Ketersediaan dan replikasi tinggi.
- **`--promotion-tier`** — Opsional. Tingkat prioritas failover untuk instans ini. Harus antara 0 dan 15 dengan angka yang lebih rendah menjadi prioritas yang lebih tinggi. Untuk informasi selengkapnya, lihat Mengontrol target failover.

1. Pertama, tentukan Availability Zone tempat Anda dapat membuat instance.

   Jika Anda ingin menentukan Availability Zone sebelum membuat instans, jalankan perintah berikut untuk menentukan Availability Zone mana yang tersedia untuk klaster Amazon DocumentDB Anda.

   Untuk Linux, macOS, atau Unix:

   ```
   aws docdb describe-db-clusters \
         --query 'DBClusters[*].[DBClusterIdentifier,AvailabilityZones[*]]'
   ```

   Untuk Windows:

   ```
   aws docdb describe-db-clusters ^
         --query 'DBClusters[*].[DBClusterIdentifier,AvailabilityZones[*]]'
   ```

   Output dari operasi ini terlihat seperti berikut.

   ```
   [
       [
           "sample-cluster",
           [
               "us-east-1c",
               "us-east-1b",
               "us-east-1a"
           ]
       ]
   ```

```
]
```

2. Kedua, tentukan kelas instance apa yang dapat Anda buat di Wilayah Anda.

   Untuk menentukan kelas instans mana yang tersedia untuk Anda di Wilayah Anda, jalankan perintah berikut. Dari output, pilih kelas instans untuk instans yang ingin Anda tambahkan ke klaster Amazon DocumentDB Anda.

   Untuk Linux, macOS, atau Unix:

   ```
   aws docdb describe-orderable-db-instance-options \
           --engine docdb \
           --query 'OrderableDBInstanceOptions[*].DBInstanceClass'
   ```

   Untuk Windows:

   ```
   aws docdb describe-orderable-db-instance-options ^
           --engine docdb ^
           --query 'OrderableDBInstanceOptions[*].DBInstanceClass'
   ```

   Output dari operasi ini terlihat seperti berikut.

   ```
   [
       "db.r5.16xlarge",
       "db.r5.2xlarge",
       "db.r5.4xlarge",
       "db.r5.8xlarge",
       "db.r5.large",
       "db.r5.xlarge"
   ]
   ```

3. Terakhir, tambahkan instance ke cluster Amazon DocumentDB Anda.

   Untuk menambahkan instans ke klaster Amazon DocumentDB Anda, jalankan perintah berikut..

   Untuk Linux, macOS, atau 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
```

Untuk 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
```

Output dari operasi ini terlihat seperti berikut.

```
{
    "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"
    }
}
```

Dibutuhkan beberapa menit untuk membuat instans. Anda dapat menggunakan konsol atau AWS CLI untuk melihat status instans. Untuk informasi selengkapnya, lihat Memantau status instans Amazon DocumentDB.

Menjelaskan instans Amazon DocumentDB

Anda dapat menggunakan Konsol Manajemen Amazon DocumentDB atau AWS CLI untuk melihat detail seperti titik akhir koneksi, VPC grup keamanan, otoritas sertifikat, dan grup parameter yang berkaitan dengan instans Amazon DocumentDB Anda.

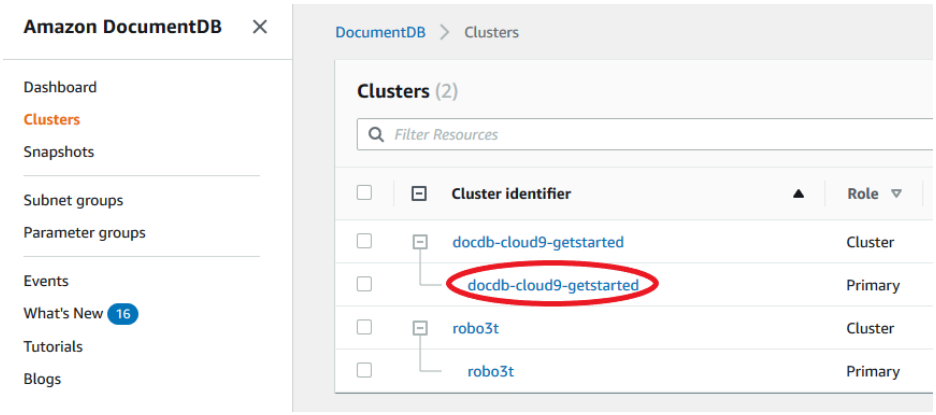
Using the Konsol Manajemen AWS

Untuk melihat detail instans Anda menggunakan Konsol Manajemen AWS, ikuti langkah-langkah di bawah ini.

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Klaster.

> **Tip**
>
> Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu (≡) di pojok kiri atas halaman.

3. Di kotak navigasi Klaster, Anda akan melihat kolom Pengidentifikasi klaster. Instance Anda tercantum di bawah cluster, mirip dengan tangkapan layar berikut.



4. Dalam daftar instans, pilih nama instans yang ingin Anda lihat detailnya. Informasi tentang instans diatur ke dalam pengelompokan berikut:

   - Ringkasan — Informasi umum tentang instans, termasuk versi mesin, kelas, status, dan setiap pemeliharaan tertunda.
   - Konektivitas & Keamanan — Bagian Hubungkan berisi daftar titik akhir koneksi untuk membuat sambungan ke instans ini dengan mongo shell atau dengan aplikasi. Bagian Grup Keamanan terdiri atas daftar grup keamanan yang terkait dengan instans ini dan ID VPC serta deskripsinya.
   - Konfigurasi — Bagian Detail berisi daftar konfigurasi dan status instans, termasuk Amazon Resource Name (ARN), titik akhir, peran, kelas, dan otoritas sertifikat instans. Ini juga berisi daftar pengaturan keamanan dan jaringan, serta informasi backup instans. Bagian Detail klaster berisi daftar detail klaster tempat instans ini berada. Bagian Instans klaster berisi daftar semua instans yang dimiliki klaster Anda dengan masing-masing peran instans dan status grup parameter klaster.

> **Note**
>
> Anda dapat memodifikasi klaster yang terkait dengan instans Anda dengan memilih Modifikasi di samping header Detail klaster. Untuk informasi selengkapnya, lihat Memodifikasi cluster Amazon DocumentDB.

- Monitoring —Metrik CloudWatch Log untuk contoh ini. Untuk informasi selengkapnya, lihat Memantau Amazon DocumentDB dengan CloudWatch.
- Peristiwa & tag — Bagian Peristiwa terbaru berisi daftar peristiwa terbaru untuk instans ini. Amazon DocumentDB menyimpan catatan peristiwa yang terkait dengan klaster, instans, snapshot, grup keamanan, dan grup parameter klaster Anda. Informasi ini mencakup tanggal, waktu, dan pesan yang terkait dengan setiap peristiwa. Bagian Tag berisi daftar tag yang dilampirkan ke klaster ini. Untuk informasi selengkapnya, lihat Menandai sumber daya Amazon DocumentDB.

Using the AWS CLI

Untuk melihat detail instance Amazon DocumentDB Anda menggunakan, gunakan perintah seperti AWS CLI yang ditunjukkan `describe-db-clusters` pada contoh berikut. Untuk informasi selengkapnya, lihat `DescribeDBInstances`di Referensi API Manajemen Sumber Daya Amazon DocumentDB.

> **Note**
>
> Untuk fitur pengelolaan tertentu seperti pengelolaan siklus hidup klaster dan instans, Amazon DocumentDB memanfaatkan teknologi operasi yang dibagikan dengan Amazon RDS. Parameter filter `filterName=engine,Values=docdb` hanya mengembalikan klaster Amazon DocumentDB.

1. Daftar semua instance Amazon DocumentDB.

   AWS CLI Kode berikut mencantumkan detail untuk semua instance Amazon DocumentDB di suatu wilayah.

   Untuk Linux, macOS, atau Unix:

```
aws docdb describe-db-instances \
    --filter Name=engine,Values=docdb
```

Untuk Windows:

```
aws docdb describe-db-instances \
    --filter Name=engine,Values=docdb
```

2. Cantumkan semua detail untuk instance Amazon DocumentDB tertentu

Kode berikut mengeluarkan daftar detail untuk `sample-cluster-instance`. Termasuk parameter `--db-instance-identifier` dengan nama instans membatasi output ke informasi pada instans tertentu.

Untuk Linux, macOS, atau Unix:

```
aws docdb describe-db-instances \
    --db-instance-identifier sample-cluster-instance
```

Untuk Windows:

```
aws docdb describe-db-instances \
    --db-instance-identifier sample-cluster-instance
```

Output dari operasi ini terlihat seperti berikut.

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


Memodifikasi instance Amazon DocumentDB

Anda dapat memodifikasi instans Amazon DocumentDB menggunakan instans atau Konsol Manajemen AWS file. AWS CLI Untuk memodifikasi sebuah instans, instans harus dalam status tersedia. Anda tidak dapat memodifikasi instans yang dihentikan. Jika klaster berhenti, pertama mulai klaster, tunggu hingga instans tersedia, lalu buat modifikasi yang diinginkan. Untuk informasi selengkapnya, lihat Menghentikan dan memulai cluster Amazon DocumentDB.

Using the Konsol Manajemen AWS

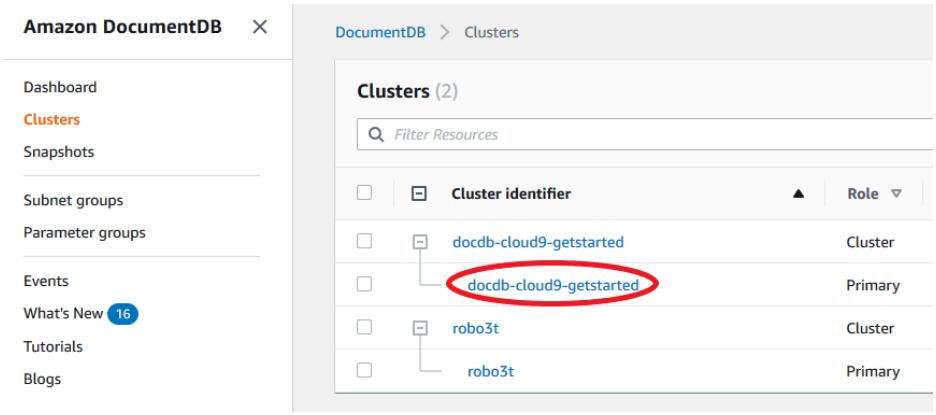
Untuk memodifikasi instans Amazon DocumentDB tertentu menggunakan konsol, selesaikan langkah berikut.

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb

2. Di panel navigasi, pilih Klaster.

> **Tip**
>
> Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu (≡) di pojok kiri atas halaman.

3. Di kotak navigasi Klaster, Anda akan melihat kolom Pengidentifikasi klaster. Instance Anda tercantum di bawah cluster, mirip dengan tangkapan layar berikut.



4. Centang kotak di sebelah kiri instans yang ingin Anda modifikasi.
5. Pilih Tindakan, dan kemudian pilih Modifikasi.
6. Di panel Modifikasi instans: <instance-name>, buat perubahan yang Anda inginkan. Anda dapat membuat perubahan berikut:

   - Spesifikasi instans — Pengidentifikasi dan kelas instans. Kendala penamaan pengidentifikasi instans:
     - Instance identifier — Masukkan nama yang unik untuk semua instans yang dimiliki oleh Anda Akun AWS di wilayah saat ini. Pengidentifikasi instans harus berisi [1-63] karakter alfanumerik atau tanda hubung, memiliki huruf sebagai karakter pertama, dan tidak dapat diakhiri dengan tanda hubung atau berisi dua tanda hubung berturut-turut.
     - Kelas instans — Dari menu tarik-turun, pilih kelas instans untuk instans Amazon DocumentDB Anda. Untuk informasi selengkapnya, lihat Mengelola kelas instance.
   - Otoritas sertifikat — Sertifikat server untuk instans ini. Untuk informasi selengkapnya, lihat Memperbarui sertifikat Amazon DocumentDB TLS Anda.
   - Failover — Selama failover, instans dengan tingkat promosi tertinggi akan dipromosikan ke primer. Untuk informasi selengkapnya, lihat Failover Amazon DocumentDB.

- Pemeliharaan — Jendela pemeliharaan di mana modifikasi atau patch yang tertunda diterapkan ke instans di klaster.

7. Setelah selesai, pilih Lanjutkan untuk melihat ringkasan perubahan Anda.
8. Setelah memverifikasi perubahan Anda, Anda dapat menerapkannya segera atau selama jendela pemeliharaan berikutnya di bawah Penjadwalan modifikasi. Pilih Modifikasi instans untuk menyimpan perubahan Anda. Atau, Anda dapat memilih Batal untuk membuang perubahan Anda.

Diperlukan beberapa menit agar perubahan Anda diterapkan. Anda dapat menggunakan instans hanya ketika statusnya tersedia. Anda dapat memantau status instans menggunakan konsol atau AWS CLI. Untuk informasi selengkapnya, lihat Memantau status instans Amazon DocumentDB.

Using the AWS CLI

Untuk memodifikasi instans Amazon DocumentDB tertentu menggunakan AWS CLI, gunakan dengan parameter `modify-db-instance` berikut. Untuk informasi selengkapnya, lihat ModifyDBInstance. Kode berikut memodifikasi kelas instans menjadi `db.r5.large` untuk instans `sample-instance`.

Parameter

- **`--db-instance-identifier`** — Diperlukan. Pengidentifikasi untuk instans yang akan dimodifikasi.
- **`--db-instance-class`** — Opsional. Kapasitas komputasi dan memori baru dari instans; misalnya, `db.r5.large`. Tidak semua kelas instans tersedia di semua Wilayah AWS. Jika Anda memodifikasi kelas instans, pemadaman terjadi selama perubahan. Perubahan diterapkan selama jendela pemeliharaan berikutnya, kecuali `ApplyImmediately` ditentukan sebagai true untuk permintaan ini.
- **`--apply-immediately`** atau **`--no-apply-immediately`** — Opsional. Menentukan apakah modifikasi ini harus diterapkan segera atau menunggu hingga jendela pemeliharaan berikutnya. Jika parameter ini dihilangkan, modifikasi dilakukan selama jendela pemeliharaan berikutnya.

Example

Untuk Linux, macOS, atau Unix:

```
aws docdb modify-db-instance \
```

```
       --db-instance-identifier sample-instance \
       --db-instance-class db.r5.large \
       --apply-immediately
```

Untuk Windows:

```
aws docdb modify-db-instance ^
       --db-instance-identifier sample-instance ^
       --db-instance-class db.r5.large ^
       --apply-immediately
```

Output dari operasi ini terlihat seperti berikut.

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

Diperlukan beberapa menit agar modifikasi Anda diterapkan. Anda dapat menggunakan instans hanya ketika statusnya tersedia. Anda dapat memantau status instans menggunakan Konsol Manajemen AWS atau AWS CLI. Untuk informasi selengkapnya, lihat Memantau status instans Amazon DocumentDB.

Mem-boot ulang instans Amazon DocumentDB

Terkadang, Anda mungkin perlu mem-boot ulang instans Amazon DocumentDB Anda, biasanya karena alasan pemeliharaan. Jika Anda melakukan perubahan tertentu, seperti mengubah grup parameter klaster yang terkait dengan klaster, Anda harus mem-boot ulang instans di klaster agar perubahan diterapkan. Anda dapat me-reboot instance tertentu menggunakan Konsol Manajemen AWS atau AWS CLI.

Mem-boot ulang instans akan memulai ulang layanan mesin basis data. Mem-boot ulang menyebabkan pemadaman sementara, selama status instans diatur ke `rebooting`. Peristiwa Amazon DocumentDB dibuat saat boot ulang selesai.

Mem-boot ulang instans tidak mengakibatkan failover. Untuk melakukan failover cluster Amazon DocumentDB, Konsol Manajemen AWS gunakan atau operasinya. AWS CLI `failover-db-cluster` Untuk informasi selengkapnya, lihat Failover Amazon DocumentDB.

Anda tidak dapat mem-boot ulang instans Anda jika tidak berada dalam status tersedia. Basis data Anda dapat tidak tersedia karena beberapa alasan, seperti modifikasi yang diminta sebelumnya, atau tindakan jendela pemeliharaan. Untuk informasi lebih lanjut tentang status instans, lihat Memantau status instans Amazon DocumentDB.

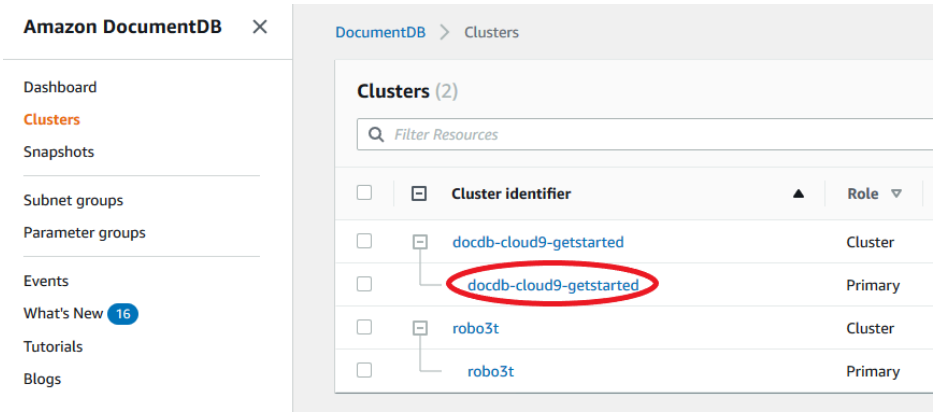
Using the Konsol Manajemen AWS

Prosedur berikut mem-boot ulang instans yang Anda tentukan menggunakan konsol.

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Klaster.

   > **Tip**
   >
   > Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu (≡) di pojok kiri atas halaman.

3. Di kotak navigasi Klaster, Anda akan melihat kolom Pengidentifikasi klaster. Instance Anda tercantum di bawah cluster, mirip dengan tangkapan layar berikut.



4. Centang kotak di sebelah kiri instans yang ingin Anda boot ulang.
5. Pilih Tindakan, pilih Boot ulang, lalu pilih Boot ulang untuk mengonfirmasi boot ulang Anda.

Diperlukan waktu beberapa menit agar instans di-boot ulang. Anda dapat menggunakan instans hanya ketika statusnya tersedia. Anda dapat memantau status instans menggunakan konsol atau AWS CLI. Untuk informasi selengkapnya, lihat Memantau status instans Amazon DocumentDB.

Using the AWS CLI

Untuk mem-boot ulang instans Amazon DocumentDB, gunakan operasi `reboot-db-instance` dengan parameter `--db-instance-identifier`. Parameter ini menentukan pengidentifikasi untuk instans yang akan di-boot ulang.

Kode berikut mem-boot ulang instans `sample-instance`.

Example

Untuk Linux, macOS, atau Unix:

```
aws docdb reboot-db-instance \
       --db-instance-identifier sample-instance
```

Untuk Windows:

```
aws docdb reboot-db-instance ^
       --db-instance-identifier sample-instance
```

Output dari operasi ini terlihat seperti berikut.

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

Diperlukan waktu beberapa menit agar instans di-boot ulang. Anda dapat menggunakan instans hanya ketika statusnya tersedia. Anda dapat memantau status instans menggunakan konsol atau AWS CLI. Untuk informasi selengkapnya, lihat Memantau status instans Amazon DocumentDB.

Menghapus instans Amazon DocumentDB

Anda dapat menghapus instans Amazon DocumentDB Anda menggunakan salah satu atau Konsol Manajemen AWS . AWS CLI Untuk menghapus sebuah instans, instans harus dalam status tersedia. Anda tidak dapat menghapus instans yang dihentikan. Jika klaster Amazon DocumentDB yang berisi instans Anda berhenti, pertama mulai klaster, tunggu instans menjadi tersedia, dan kemudian hapus instans tersebut. Untuk informasi selengkapnya, lihat Menghentikan dan memulai cluster Amazon DocumentDB.

> **Note**
>
> Amazon DocumentDB menyimpan semua data Anda dalam volume klaster. Data tetap ada dalam volume klaster tersebut, meskipun Anda menghapus semua instans dari klaster Anda. Jika Anda perlu mengakses data lagi, Anda dapat menambahkan instans ke klaster kapan saja dan melanjutkan dari bagian terakhir yang Anda tinggalkan.

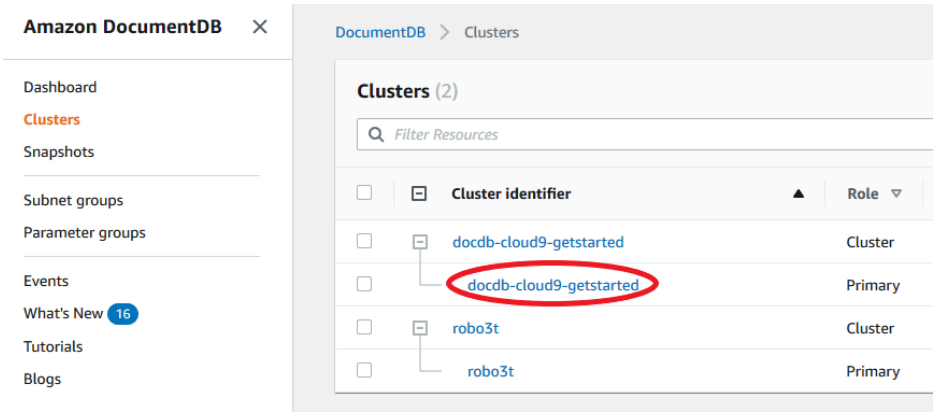
Using the Konsol Manajemen AWS

Prosedur berikut menghapus instans Amazon DocumentDB yang ditentukan dengan menggunakan konsol.

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Klaster.

> **Tip**
>
> Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu (≡) di pojok kiri atas halaman.

3. Di kotak navigasi Klaster, Anda akan melihat kolom Pengidentifikasi klaster. Instance Anda tercantum di bawah cluster, mirip dengan tangkapan layar berikut.



4. Centang kotak di sebelah kiri instans yang ingin Anda hapus.
5. Pilih Tindakan, lalu pilih Hapus.

    1. Jika Anda menghapus instans terakhir di klaster Anda:
        - Buat snapshot cluster akhir? — Pilih Ya jika Anda ingin membuat snapshot akhir sebelum klaster dihapus. Jika tidak, pilih Tidak.
        - Nama snapshot akhir — Jika Anda memilih untuk membuat snapshot akhir, masukkan pengidentifikasi snapshot klaster dari snapshot klaster baru yang dibuat.
        - Hapus contoh? <instance-name> — Masukkan frasa hapus seluruh klaster ke dalam bidang untuk mengonfirmasi penghapusan.
    2. Jika Anda tidak menghapus instans terakhir di klaster Anda:

- Hapus contoh? <instance-name> — Masukkan frasa hapus saya ke dalam bidang untuk mengonfirmasi penghapusan.

6. Pilih Hapus untuk menghapus instans.

Diperlukan beberapa menit untuk menghapus instans. Untuk memantau status instans, lihat Memantau status instans Amazon DocumentDB.

Using the AWS CLI

Prosedur berikut menghapus instans Amazon DocumentDB dengan menggunakan AWS CLI.

1. Pertama, tentukan berapa banyak instance yang ada di cluster Amazon DocumentDB Anda:

   Untuk menentukan berapa banyak instans di klaster Anda, jalankan perintah `describe-db-clusters`, sebagai berikut.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].
[DBClusterIdentifier,DBClusterMembers[*].DBInstanceIdentifier]'
```

   Output dari operasi ini terlihat seperti berikut.

```
[
    [
        "sample-cluster",
        [
            "sample-instance-1",
            "sample-instance-2"
        ]
    ]
]
```

2. Jika ada lebih dari satu instance di cluster Amazon DocumentDB Anda:

   Untuk menghapus instans Amazon DocumentDB tertentu, gunakan perintah `delete-db-instance` dengan parameter `--db-instance-identifier`, seperti yang ditunjukkan di bawah ini. Diperlukan beberapa menit untuk menghapus instans. Untuk memantau status instans, lihat Memantau status instans Amazon DocumentDB.

```
aws docdb delete-db-instance \
       --db-instance-identifier sample-instance-2
```

Output dari operasi ini akan terlihat seperti berikut.

```
{
    "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. Jika instance yang ingin Anda hapus adalah instance terakhir di cluster Amazon DocumentDB Anda:

   Jika Anda menghapus instans terakhir di klaster Amazon DocumentDB, Anda juga menghapus klaster tersebut serta snapshot otomatis dan backup terus menerus yang terkait dengan klaster tersebut.

   Untuk menghapus instans terakhir di klaster Anda, Anda dapat menghapus klaster dan secara opsional membuat snapshot akhir. Untuk informasi selengkapnya, lihat Menghapus cluster Amazon DocumentDB.

Perlindungan penghapusan

Menghapus instans terakhir dari klaster Amazon DocumentDB juga akan menghapus klaster tersebut, serta snapshot otomatis dan backup terus menerus yang terkait dengan klaster tersebut. Amazon DocumentDB memberlakukan perlindungan penghapusan untuk klaster apakah Anda

melakukan operasi penghapusan menggunakan atau. Konsol Manajemen AWS AWS CLI Jika perlindungan penghapusan diaktifkan, Anda tidak dapat menghapus klaster.

Untuk menghapus klaster yang mengaktifkan perlindungan penghapusan, Anda harus terlebih dahulu mengubah klaster dan menonaktifkan perlindungan penghapusan. Lihat informasi yang lebih lengkap di Menghapus cluster Amazon DocumentDB.

Mengelola kelas instance

Kelas instans menentukan komputasi dan kapasitas memori dari instans Amazon DocumentDB (dengan kompatibilitas MongoDB). Kelas instans yang Anda butuhkan tergantung pada kebutuhan daya dan memori pemrosesan Anda.

Amazon DocumentDB mendukung keluarga kelas instans R4, R5, R6G, R8G, T3, dan T4G. Kelas-kelas ini adalah kelas instans generasi saat ini yang dioptimalkan untuk aplikasi yang memakan memori. Untuk spesifikasi pada kelas-kelas ini, lihat Spesifikasi kelas instans.

Topik

- Menentukan kelas instance
- Mengubah kelas instance
- Kelas instance yang didukung berdasarkan wilayah
- Spesifikasi kelas instans

Menentukan kelas instance

Untuk menentukan kelas sebuah instance, Anda dapat menggunakan Konsol Manajemen AWS atau `describe-db-instances` AWS CLI operasi.

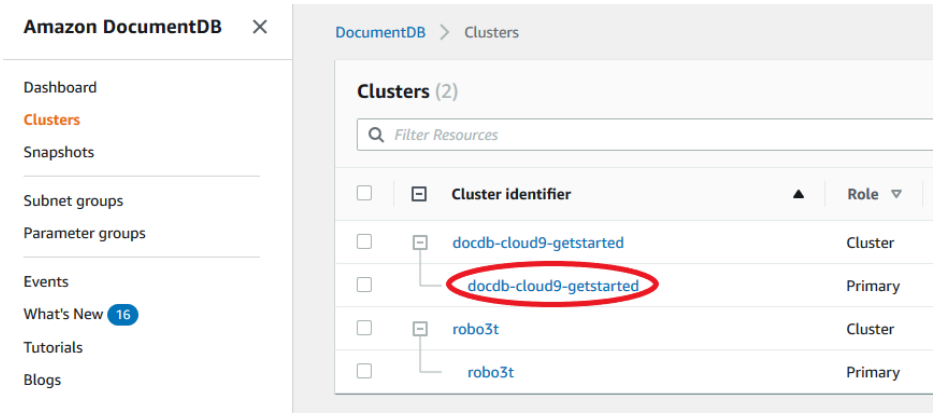
Using the Konsol Manajemen AWS

Untuk menentukan kelas instans bagi instans klaster Anda, selesaikan langkah-langkah berikut di konsol.

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Klaster untuk menemukan instans yang Anda minati.

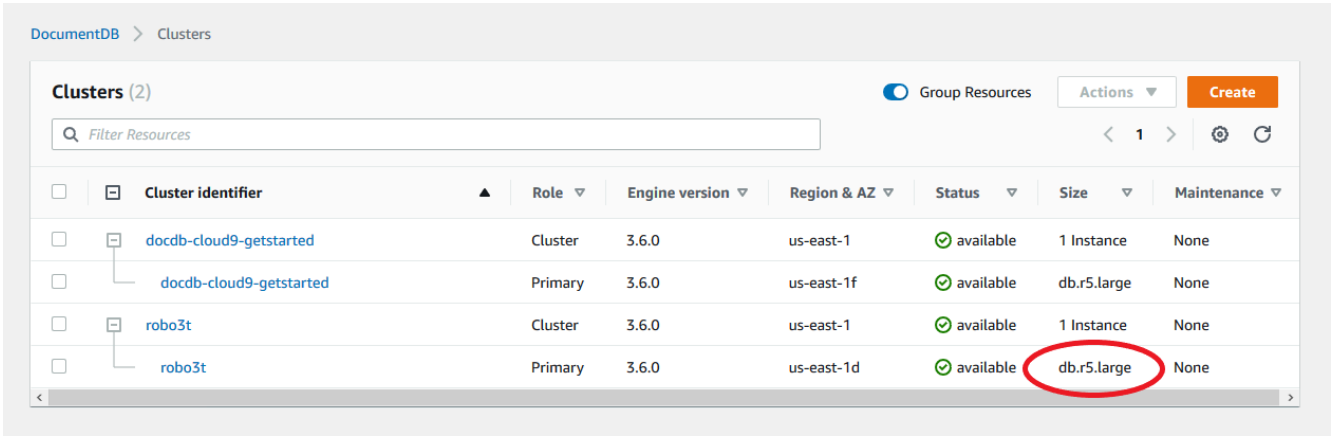
> **Tip**
>
> Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu (≡) di pojok kiri atas halaman.

3. Di kotak navigasi Klaster, Anda akan melihat kolom Pengidentifikasi klaster. Instance Anda tercantum di bawah cluster, mirip dengan tangkapan layar berikut.



4. Dalam daftar instans, perluas klaster untuk menemukan instans Anda minati. Temukan instans yang Anda inginkan. Kemudian, lihat pada kolom Ukuran dari baris instans untuk melihat kelas instansnya.

   Pada gambar berikut, kelas instans untuk instans `robo3t` adalah `db.r5.4xlarge`.



Using the AWS CLI

Untuk menentukan kelas instance menggunakan AWS CLI, gunakan `describe-db-instances` operasi dengan parameter berikut.

- **`--db-instance-identifier`**— Opsional. Menentukan instans yang ingin Anda temukan kelas instansnya. Jika parameter ini dihilangkan, `describe-db-instances` mengembalikan deskripsi hingga 100 instans Anda.
- **`--query`** — Opsional. Menentukan anggota instans untuk disertakan dalam hasil. Jika parameter ini dihilangkan, semua anggota instans dikembalikan.

Example

Contoh berikut menemukan nama dan kelas instans untuk instans `sample-instance-1`.

Untuk Linux, macOS, atau Unix:

```
aws docdb describe-db-instances \
    --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]' \
    --db-instance-identifier sample-instance-1
```

Untuk Windows:

```
aws docdb describe-db-instances ^
    --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]' ^
    --db-instance-identifier sample-instance-1
```

Output dari operasi ini terlihat seperti berikut.

```
[
    [
        "sample-instance-1",
        "db.r5.large"
    ]
```

Example

Contoh berikut menemukan nama dan kelas instans hingga 100 instans Amazon DocumentDB.

Untuk Linux, macOS, atau Unix:

```
aws docdb describe-db-instances \
    --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]' \
    --filter Name=engine,Values=docdb
```

Untuk Windows:

```
aws docdb describe-db-instances ^
    --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]' ^
    --filter Name=engine,Values=docdb
```

Output dari operasi ini terlihat seperti berikut.

```
[
    [
        "sample-instance-1",
        "db.r5.large"
    ],
    [
        "sample-instance-2",
        "db.r5.large"
    ],
    [
        "sample-instance-3",
        "db.r5.4xlarge"
    ],
    [
        "sample-instance-4",
        "db.r5.4xlarge"
    ]
]
```

Untuk informasi selengkapnya, lihat Menjelaskan instans Amazon DocumentDB.

Mengubah kelas instance

Anda dapat mengubah kelas instance instance Anda menggunakan Konsol Manajemen AWS atau AWS CLI. Untuk informasi selengkapnya, lihat Memodifikasi instance Amazon DocumentDB.

Kelas instance yang didukung berdasarkan wilayah

Amazon DocumentDB mendukung kelas instans berikut:

- R8G—Generasi terbaru dari instans yang dioptimalkan memori yang didukung oleh prosesor Arm-based AWS Graviton4 yang memberikan kinerja hingga 30% lebih baik dibandingkan instans R6G.
- R6G- Memory-optimized Instans yang didukung oleh prosesor Arm-based AWS Graviton2 yang memberikan kinerja hingga 30% lebih baik dibandingkan instans R5 dengan biaya 5% lebih murah.

- R6GD- Instans Memory-optimized R6G dengan penyimpanan Solid-State Drive (SSD) berbasis memori non-volatile (NVMe) lokal untuk data sementara.
- R5— Memory-optimized Instans yang memberikan kinerja hingga 100% lebih baik dibandingkan instans R4 dengan biaya instans yang sama.
- R4 — Generasi sebelumnya dari instans dengan memori yang dioptimalkan.
- T4G— Jenis instans tujuan umum burstable dengan biaya Latest-generation rendah yang ditenagai oleh prosesor Arm-based AWS Graviton2 yang memberikan tingkat kinerja CPU dasar, memberikan kinerja harga hingga 35% lebih baik dibandingkan instans T3 dan ideal untuk menjalankan aplikasi dengan penggunaan CPU moderat yang mengalami lonjakan penggunaan sementara.
- T3—Jenis instans tujuan umum burstable dengan biaya rendah yang memberikan tingkat kinerja CPU dasar dengan kemampuan untuk meningkatkan penggunaan CPU kapan saja selama diperlukan.

Untuk spesifikasi mendetail tentang kelas instans, lihat Spesifikasi kelas instans.

> **Note**
>
> I/O-Optimized penyimpanan hanya tersedia pada engine versi 5.0 dan 8.0 (cluster berbasis instance).

Sebuah kelas instans tertentu mungkin atau mungkin tidak didukung di Wilayah tertentu. Tabel berikut menentukan kelas instans yang didukung oleh Amazon DocumentDB di setiap Wilayah.

Kelas instans yang didukung berdasarkan Wilayah

Region	R8G	R6GD	R6G	R5	R4	T4G	T3	Nirserver
AS Timur (Ohio)	Didukung	Didukung	Didukung	Didukung	Didukung	Didukung	Didukung	Didukung
US East (N. Virginia)	Didukung	Didukung	Didukung	Didukung	Didukung	Didukung	Didukung	Didukung
US West (Oregon)	Didukung	Didukung	Didukung	Didukung	Didukung	Didukung	Didukung	Didukung

Region	R8G	R6GD	R6G	R5	R4	T4G	T3	Nirserver
Africa (Cape Town)			Didukung	Didukung		Didukung	Didukung	Didukung
Amerika Selatan (Sao Paulo)		Didukung	Didukung	Didukung		Didukung	Didukung	Didukung
Asia Pasifik (Hong Kong)			Didukung	Didukung		Didukung	Didukung	Didukung
Asia Pasifik (Hyderabad)			Didukung	Didukung		Didukung	Didukung	Didukung
Asia Pasifik (Malaysia)			Didukung			Didukung	Didukung	
Asia Pasifik (Mumbai)	Didukung	Didukung	Didukung	Didukung		Didukung	Didukung	Didukung
Asia Pasifik (Osaka)		Didukung	Didukung	Didukung		Didukung	Didukung	
Asia Pasifik (Seoul)	Didukung	Didukung	Didukung	Didukung		Didukung	Didukung	Didukung
Asia Pacific (Sydney)	Didukung	Didukung	Didukung	Didukung		Didukung	Didukung	Didukung
Asia Pasifik (Jakarta)	Didukung	Didukung	Didukung	Didukung		Didukung	Didukung	
Asia Pacific (Melbourne)			Didukung	Didukung		Didukung	Didukung	
Asia Pacific (Singapore)	Didukung	Didukung	Didukung	Didukung		Didukung	Didukung	Didukung
Asia Pasifik (Thailand)			Didukung			Didukung	Didukung	

Region	R8G	R6GD	R6G	R5	R4	T4G	T3	Nirserver
Asia Pacific (Tokyo)	Didukung	Didukung	Didukung	Didukung		Didukung	Didukung	Didukung
Kanada (Pusat)	Didukung	Didukung	Didukung	Didukung		Didukung	Didukung	Didukung
Kanada Barat (Calgary)			Didukung	Didukung		Didukung	Didukung	
Eropa (Frankfurt)	Didukung	Didukung	Didukung	Didukung		Didukung	Didukung	Didukung
Europe (Zurich)		Didukung	Didukung	Didukung		Didukung	Didukung	
Eropa (Irlandia)	Didukung	Didukung	Didukung	Didukung	Didukung	Didukung	Didukung	Didukung
Eropa (London)		Didukung	Didukung	Didukung		Didukung	Didukung	Didukung
Europe (Milan)			Didukung	Didukung		Didukung	Didukung	Didukung
Eropa (Paris)		Didukung	Didukung	Didukung		Didukung	Didukung	Didukung
Eropa (Spanyol)	Didukung	Didukung	Didukung	Didukung		Didukung	Didukung	Didukung
Eropa (Stockholm)	Didukung	Didukung	Didukung	Didukung		Didukung	Didukung	
Meksiko (Tengah)			Didukung			Didukung	Didukung	
Timur Tengah (UAE)			Didukung	Didukung		Didukung	Didukung	Didukung
Tiongkok (Beijing)		Didukung	Didukung	Didukung		Didukung	Didukung	Didukung
Tiongkok (Ningxia)			Didukung	Didukung		Didukung	Didukung	Didukung
Israel (Tel Aviv)			Didukung	Didukung		Didukung	Didukung	

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

Spesifikasi kelas instans

Tabel berikut memberikan rincian kelas instans Amazon DocumentDB, termasuk jenis instance yang didukung di setiap kelas. Anda dapat menemukan penjelasan untuk setiap kolom tabel di bawah tabel.

Kelas instans	vCPU[1]	Memori (GiB) 2	Cache berjenjang NVMe SSD (GiB) 3	Maks. temp. penyimpanan (GiB) 4	Bandwidth dasar/burst (Gbps) 5	Mesin Pendukung 6
R8G - Kelas Memory-Optimized Instance Generasi Saat Ini berdasarkan Graviton4 R8G adalah keluarga instans terbaru (Graviton4), hanya tersedia pada versi mesin 5.0 dan 8.0. Ukuran db.r8g.48xlarge belum tersedia.						
`db.r8g.large`	2	16	-	30	0,937/12,5	5.0.0 dan 8.0.0
`db.r8g.xlarge`	4	32	-	60	1.875/12,5	5.0.0 dan 8.0.0
`db.r8g.2xlarge`	8	64	-	121	3,75/15,0	5.0.0 dan 8.0.0
`db.r8g.4xlarge`	16	128	-	243	7,5/15,0	5.0.0 dan 8.0.0
`db.r8g.8xlarge`	32	256	-	488	15	5.0.0 dan 8.0.0

Kelas instans	vCPU[1]	Memori (GiB) 2	Cache berjenjang NVMe SSD (GiB) 3	Maks. temp. penyimpanan (GiB) 4	Bandwidth dasar/burst (Gbps) 5	Mesin Pendukung 6
db.r8g.12xlarge	48	384	-	732	22	5.0.0 dan 8.0.0
db.r8g.16xlarge	64	512	-	987	30	5.0.0 dan 8.0.0
db.r8g.24xlarge	96	768	-	1484	40	5.0.0 dan 8.0.0
R6G - Kelas Memory-Optimized Instance Generasi Saat Ini berdasarkan Graviton2						
db.r6g.large	2	16	-	32	0,75/10	4.0.0, 5.0.0, dan 8.0.0
db.r6g.xlarge	4	32	-	63	1,25/10	4.0.0, 5.0.0, dan 8.0.0
db.r6g.2xlarge	8	64	-	126	2.5/10	4.0.0, 5.0.0, dan 8.0.0
db.r6g.4xlarge	16	128	-	252	5.0/10	4.0.0, 5.0.0, dan 8.0.0
db.r6g.8xlarge	32	256	-	504	12	4.0.0, 5.0.0, dan 8.0.0
db.r6g.12xlarge	48	384	-	756	20	4.0.0, 5.0.0, dan 8.0.0

Kelas instans	vCPU[1]	Memori (GiB) 2	Cache berjenjang NVMe SSD (GiB) 3	Maks. temp. penyimpanan (GiB) 4	Bandwidth dasar/burst (Gbps) 5	Mesin Pendukung 6
db.r6g.16xlarge	64	512	-	1008	25	4.0.0, 5.0.0, dan 8.0.0
R6GD - Kelas NVMe-backed Instance Generasi Saat Ini berdasarkan Graviton2						
db.r6gd.xlarge	4	32	173	64	1,25/10	5.0.0 dan 8.0.0
db.r6gd.2xlarge	8	64	346	128	2.5/10	5.0.0 dan 8.0.0
db.r6gd.4xlarge	16	128	694	256	5.0/10	5.0.0 dan 8.0.0
db.r6gd.8xlarge	32	256	1388	512	12	5.0.0 dan 8.0.0
db.r6gd.12xlarge	48	384	2082	768	20	5.0.0 dan 8.0.0
db.r6gd.16xlarge	64	512	2776	1024	25	5.0.0 dan 8.0.0
R5 - Kelas Memory-Optimized Instance Generasi Sebelumnya						
db.r5.large	2	16	-	31	0,75/10	3.6.0, 4.0.0, 5.0.0, dan 8.0.0

Kelas instans	vCPU[1]	Memori (GiB) 2	Cache berjenjang NVMe SSD (GiB) 3	Maks. temp. penyimpanan (GiB) 4	Bandwidth dasar/burst (Gbps) 5	Mesin Pendukung 6
db.r5.xlarge	4	32	-	62	1,25/10	3.6.0, 4.0.0, 5.0.0, dan 8.0.0
db.r5.2xlarge	8	64	-	124	2.5/10	3.6.0, 4.0.0, 5.0.0, dan 8.0.0
db.r5.4xlarge	16	128	-	249	5.0/10	3.6.0, 4.0.0, 5.0.0, dan 8.0.0
db.r5.8xlarge	32	256	-	504	10	3.6.0, 4.0.0, 5.0.0, dan 8.0.0
db.r5.12xlarge	48	384	-	748	12	3.6.0, 4.0.0, 5.0.0, dan 8.0.0
db.r5.16xlarge	64	512	-	1008	20	3.6.0, 4.0.0, 5.0.0, dan 8.0.0

Kelas instans	vCPU[1]	Memori (GiB) 2	Cache berjenjang NVMe SSD (GiB) 3	Maks. temp. penyimpanan (GiB) 4	Bandwidth dasar/burst (Gbps) 5	Mesin Pendukung 6
db.r5.24xlarge	96	768	-	1500	25	3.6.0, 4.0.0, 5.0.0, dan 8.0.0
R4 - Kelas Memory-Optimized Instance Generasi Sebelumnya Instans R4 hanya didukung pada engine versi 3.6. Perhatikan bahwa Amazon DocumentDB 3.6 mencapai akhir dukungan standar pada 30 Maret 2026. Extended Support (berbayar) tersedia hingga Maret 2029.						
db.r4.large	2	15.25	-	30	0,75/10	3.6.0 saja
db.r4.xlarge	4	30,5	-	60	1,25/10	3.6.0 saja
db.r4.2xlarge	8	61	-	120	2.5/10	3.6.0 saja
db.r4.4xlarge	16	122	-	240	5.0 /10	3.6.0 saja
db.r4.8xlarge	32	244	-	480	10	3.6.0 saja
db.r4.16xlarge	64	488	-	960	25	3.6.0 saja
T4G — Kelas Instans Kinerja Burstable Generasi Terbaru berdasarkan Graviton2 T-series instance berjalan dalam mode burst CPU Tidak Terbatas. Penggunaan burst di luar baseline ditagih ekstra. Tidak didukung untuk Global Cluster.						

Kelas instans	vCPU[1]	Memori (GiB) 2	Cache berjenjang NVMe SSD (GiB) 3	Maks. temp. penyimpanan (GiB) 4	Bandwidth dasar/burst (Gbps) 5	Mesin Pendukung 6
`db.t4g.medium`	2	4	-	8.13	0,256/5	4.0.0, 5.0.0, dan 8.0.0
T3 — Kelas Instans Kinerja Burstable Generasi Sebelumnya						
`db.t3.medium`	2	4	-	7,5	0,256/5	3.6.0, 4.0.0, 5.0.0, dan 8.0.0

Kelas instans	vCPU[1]	Memori (GiB) 2	Cache berjenjang NVMe SSD (GiB) 3	Maks. temp. penyimpanan (GiB) 4	Bandwidth dasar/burst (Gbps) 5	Mesin Pendukung 6

1. vCPU — Jumlah unit pemrosesan pusat (CPU) virtual. CPU virtual adalah unit kapasitas yang dapat Anda gunakan untuk membandingkan kelas instans. Alih-alih membeli atau menyewa prosesor tertentu untuk digunakan selama beberapa bulan atau tahun, Anda menyewa kapasitas per jam. Tujuan kami adalah menyediakan jumlah kapasitas CPU yang konsisten, apa pun perangkat keras yang mendasarinya.
2. Memori (GiB) — RAM, dalam gigabyte, yang dialokasikan ke instans. Sering kali ada rasio yang konsisten antara memori dan vCPU.
3. Cache berjenjang SSD NVMe — Ruang pada volume SSD, diukur dalam gigabyte, dialokasikan sebagai cache tambahan untuk menyimpan data sementara. Cache ini hanya tersedia dalam NVMe-backed kasus.
4. Maks. temp. penyimpanan (GiB) - Ruang, diukur dalam gigabyte, dialokasikan ke instance untuk penyimpanan file sementara yang tidak persisten. Misalnya NVMe-backed , penyimpanan ini di-host pada volume NVMe-based SSD. Dalam semua kasus lainnya, ini di-host di Amazon Elastic Block Store (EBS).
5. Baseline /burst bandwidth (Gbps) — Burst bandwidth mewakili bandwidth maksimum dalam gigabit per detik. Bagilah dengan 8 untuk mendapatkan throughput yang diharapkan dalam gigabyte per detik. Contoh ukuran 4xlarge dan lebih kecil memiliki bandwidth dasar. Untuk memenuhi permintaan tambahan, mereka dapat menggunakan mekanisme I/O kredit jaringan untuk melampaui bandwidth dasar mereka. Instans dapat menggunakan lonjakan bandwidth untuk waktu yang terbatas, biasanya dari 5 hingga 60 menit, tergantung pada ukuran instans.
6. Mesin Pendukung — Mesin Amazon DocumentDB yang mendukung kelas instans.

NVMe-backed contoh

NVMe-backed instans menawarkan kinerja kueri hingga 7x lebih cepat untuk aplikasi dengan kumpulan data besar yang melebihi memori instans biasa. Instans ini memanfaatkan penyimpanan Solid-State Drive (SSD) berbasis memori non-volatile (NVMe) lokal yang tersedia pada instans r6gd untuk menyimpan data sementara, mengurangi akses penyimpanan berbasis jaringan, dan meningkatkan latensi dan throughput baca.

Ruang SSD lokal dibagi menjadi dua bagian:

- Cache berjenjang — Sekitar 73% dari SSD lokal dialokasikan sebagai cache database, memungkinkan sistem untuk menyimpan hingga lima kali lebih banyak halaman database daripada memori utama saja. SSD lokal bertindak sebagai cache tingkat kedua, sedangkan cache buffer dalam memori yang ada tetap menjadi cache tingkat pertama. Kueri hanya mengakses penyimpanan eksternal jika cache buffer dan cache SSD mengalami kehilangan.
- Penyimpanan sementara — 27% sisanya dicadangkan untuk penyimpanan file sementara yang tidak persisten, digunakan untuk kueri kompleks yang melibatkan jenis atau operasi intensif sumber daya seperti pembuatan indeks. Pada instans reguler, ruang sementara berada di volume Amazon Elastic Block Store (EBS). Penyimpanan sementara yang dihosting secara lokal pada SSD mengurangi latensi kueri yang melibatkan jenis hingga dua kali dan mempercepat operasi intensif sumber daya seperti pembuatan indeks.

Rincian mengenai jenis NVMe-backed instans yang didukung dan spesifikasinya dapat ditemukan diSpesifikasi kelas instans.

Topik

- Kasus penggunaan yang disarankan untuk NVMe-backed instance
- Menggunakan NVMe-backed instance dengan Amazon DocumentDB
- NVMe-backed Contoh pemantauan

Kasus penggunaan yang disarankan untuk NVMe-backed instance

Kami menyarankan Anda menggunakan NVMe-backed instance dalam skenario berikut:

- Read-heavy beban kerja - Jika beban kerja Anda padat baca, dan kumpulan data Anda lebih besar dari cache buffer, yang ditunjukkan oleh `ReadIOPS` metrik rendah `BufferCacheHitRatio` dan tinggi, instance dapat memberikan manfaat kinerja. NVMe-backed
- Update-heavy beban kerja - Jika beban kerja Anda intensif pembaruan, dan pengumpulan sampah tidak dapat mengikuti karena latensi baca pada penyimpanan jaringan, NVMe-backed instance dapat membantu mengurangi masalah.

NVMe-backed contoh dapat menguntungkan berbagai kasus penggunaan, termasuk:

- Internet-scale Aplikasi seperti pemrosesan pembayaran, penagihan, dan e-commerce dengan Perjanjian Tingkat Layanan (SLA) kinerja yang ketat dapat memanfaatkan keunggulan kinerja instans. NVMe-backed
- Real-time dasbor pelaporan — Dasbor yang menjalankan ratusan kueri untuk metrics/data pengumpulan dapat memanfaatkan latensi rendah dan throughput instans yang tinggi. NVMe-backed
- Aplikasi AI generatif — Aplikasi yang menggunakan pencarian vektor untuk menemukan tetangga yang tepat atau terdekat di jutaan penyematan vektor dapat memanfaatkan kinerja instance yang tinggi. NVMe-backed

Menggunakan NVMe-backed instance dengan Amazon DocumentDB

Untuk menggunakan NVMe-backed instance Amazon DocumentDB:

- Buat klaster Amazon DocumentDB dan tambahkan salah satu kelas instance. NVMe-backed Untuk informasi selengkapnya, lihat Membuat cluster Amazon DocumentDB.
- Atau, ubah cluster Amazon DocumentDB yang ada untuk menggunakan salah satu kelas instance. NVMe-backed Untuk informasi selengkapnya, lihat Memodifikasi cluster Amazon DocumentDB.

Untuk memeriksa ketersediaan NVMe-backed instans di berbagai AWS Wilayah, lihatKelas instance yang didukung berdasarkan wilayah.

Jika Anda ingin beralih kembali dari NVMe-backed instance ke instance biasa, ubah kelas instans database instans Amazon DocumentDB Anda ke kelas instance serupa tanpa penyimpanan NVMe. Misalnya, jika kelas instance Anda saat ini adalah 'db.r6gd.4xlarge', pilih 'db.r6g.4xlarge' untuk beralih kembali. Untuk informasi selengkapnya, lihat Memodifikasi cluster Amazon DocumentDB.

NVMe-backed Contoh pemantauan

Selain metrik instans reguler yang tersedia di Amazon CloudWatch, NVMe-backed instans memancarkan metrik tambahan khusus untuk penyimpanan NVMe-based SSD, IOPS, dan throughput.

```
NVMeStorageCacheHitRatio
FreeNVMeStorage
ReadIOPSNVMeStorage
ReadLatencyNVMeStorage
ReadThroughputNVMeStorage
```

```
WriteIOPSNVMeStorage
WriteLatencyNVMeStorage
WriteThroughputNVMeStorage
```

Untuk informasi selengkapnya tentang metrik ini, lihat NVMe-backed metrik contoh

Mengelola grup subnet Amazon DocumentDB

Virtual Private Cloud (VPC) adalah jaringan virtual yang didedikasikan untuk Anda. Akun AWS Ini secara logis terisolasi dari jaringan virtual lain di AWS Cloud. Anda dapat meluncurkan AWS sumber daya Anda, seperti cluster Amazon DocumentDB, ke dalam Amazon VPC Anda. Anda dapat menentukan kisaran alamat IP untuk VPC, menambahkan subnet, grup keamanan yang berkaitan, dan mengonfigurasi tabel rute.

Subnet adalah serangkaian alamat IP di Amazon VPC Anda. Anda dapat meluncurkan AWS sumber daya ke subnet tertentu. Gunakan subnet publik untuk sumber daya yang harus dihubungkan ke internet. Gunakan subnet privat untuk sumber daya yang tidak akan dihubungkan ke internet. Untuk informasi selengkapnya tentang subnet publik dan privat, lihat Basic VPC dan Subnet di Panduan Pengguna Amazon Virtual Private Cloud.

Grup subnet DB adalah kumpulan subnet yang Anda buat di VPC dan kemudian Anda tetapkan untuk klaster Anda. Grup subnet memungkinkan Anda untuk menentukan VPC tertentu saat membuat klaster. Jika Anda menggunakan grup subnet `default`, ini mencakup semua subnet di VPC.

Setiap grup subnet DB harus memiliki subnet di setidaknya dua Availability Zone di Wilayah tertentu. Saat membuat klaster DB dalam VPC, Anda harus memilih grup subnet DB. Amazon DocumentDB menggunakan grup subnet DB dan Availability Zone pilihan Anda untuk memilih subnet dan alamat IP di dalam subnet tersebut untuk dikaitkan dengan klaster Anda. Jika instans primer gagal, Amazon DocumentDB dapat mempromosikan instans replika yang sesuai menjadi primer baru. Hal ini kemudian dapat membuat instans replika baru menggunakan alamat IP subnet di mana primer sebelumnya terletak.

Saat Amazon DocumentDB membuat instans dalam VPC, jaringan ini menetapkan antarmuka jaringan ke klaster Anda menggunakan alamat IP yang dipilih dari grup subnet DB Anda. Kami sangat menyarankan agar Anda menggunakan nama DNS karena alamat IP dapat berubah selama failover. Untuk informasi selengkapnya, lihat Titik akhir Amazon DocumentDB.

Untuk informasi tentang membuat VPC dan subnet Anda sendiri, lihat Bekerja dengan VPC dan Subnet di Panduan Pengguna Amazon Virtual Private Cloud.

Topik

- Membuat grup subnet Amazon DocumentDB
- Menjelaskan grup subnet Amazon DocumentDB
- Memodifikasi grup subnet Amazon DocumentDB
- Menghapus grup subnet Amazon DocumentDB

Membuat grup subnet Amazon DocumentDB

Saat membuat klaster Amazon DocumentDB, Anda harus memilih Amazon VPC dan grup subnet yang sesuai di dalam Amazon VPC tersebut untuk meluncurkan klaster Anda. Subnet menentukan availability zone dan kisaran IP di dalam availability zone yang ingin Anda gunakan untuk meluncurkan instans.

Grup subnet adalah serangkaian subnet (atau AZ) yang diberi nama yang mengizinkan Anda untuk menentukan availability zone yang ingin Anda gunakan untuk meluncurkan instans Amazon DocumentDB. Sebagai contoh, dalam klaster dengan tiga instans, disarankan bahwa setiap instans tersebut disediakan dalam AZ terpisah—sehingga mengoptimalkan ketersediaan tinggi. Dengan demikian, jika AZ tunggal gagal, maka hanya akan memengaruhi instans tunggal.

Saat ini, instans Amazon DocumentDB dapat disediakan dalam hingga tiga AZ. Bahkan jika grup subnet memiliki lebih dari tiga subnet, Anda hanya akan dapat menggunakan tiga subnet tersebut untuk membuat klaster Amazon DocumentDB. Oleh karena itu, kami menyarankan bahwa ketika Anda membuat grup subnet di mana Anda hanya memilih tiga subnet tempat Anda akan menyebarkan instans Anda.

Sebagai contoh: Klaster dibuat dan Amazon DocumentDB memilih AZ {1A, 1B, dan 1C}. Jika Anda mencoba untuk membuat instans di AZ {1D} panggilan API akan gagal. Namun demikian, jika Anda memilih untuk membuat instans, tanpa menentukan AZ tertentu, maka Amazon DocumentDB akan memilih AZ atas nama Anda. Amazon DocumentDB menggunakan algoritme untuk penyeimbangan beban instans di seluruh AZ untuk membantu Anda mencapai ketersediaan tinggi. Jika tiga instans disediakan, secara default, mereka akan disediakan di tiga AZ dan tidak akan disediakan semuanya dalam AZ tunggal.

Praktik Terbaik

- Kecuali Anda memiliki alasan tertentu, selalu buat grup subnet dengan tiga subnet. Ini memastikan bahwa cluster dengan tiga instans atau lebih akan dapat mencapai ketersediaan yang lebih tinggi karena instance akan disediakan di tiga AZ.

- Selalu sebarkan instans di beberapa AZ untuk mencapai ketersediaan tinggi. Jangan pernah menempatkan semua instance untuk cluster dalam satu AZ.
- Karena peristiwa failover dapat terjadi kapan saja, Anda tidak boleh berasumsi bahwa instance utama atau instance replika akan selalu berada di AZ tertentu.

Cara untuk membuat grup subnet

Anda dapat menggunakan Konsol Manajemen AWS atau AWS CLI untuk membuat grup subnet Amazon DocumentDB:

Using the Konsol Manajemen AWS

Gunakan langkah-langkah berikut ini untuk membuat grup subnet Amazon DocumentDB.

Untuk membuat grup subnet Amazon DocumentDB

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Grup subnet, kemudian pilih Buat.

> **Tip**
>
> Jika Anda tidak melihat panel navigasi pada sisi kiri layar Anda, pilih ikon menu (≡) di sudut kiri atas halaman.

3. Pada halaman Buat grup subnet:
   a. Di bagian Detail grup subnet:
      i. Nama—Masukkan nama yang bermakna untuk grup subnet.
      ii. Deskripsi—Masukkan deskripsi untuk grup subnet.
   b. Di bagian Tambahkan subnet:
      i. VPC—Dalam daftar, pilih VPC untuk grup subnet ini.
      ii. Lakukan salah satu tindakan berikut:
         - Untuk memasukkan semua subnet dalam VPC yang dipilih, pilih Tambahkan semua subnet yang berkaitan dengan VPC ini.

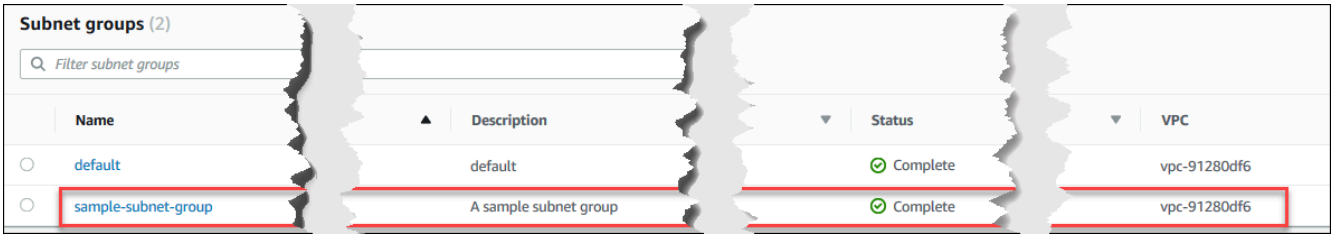
- Untuk menentukan subnet untuk grup subnet ini, lakukan hal berikut ini untuk setiap Availability Zone tempat Anda ingin menyertakan subnet. Anda harus menyertakan setidaknya dua Availability Zone.

  A. Availability Zone—Dalam daftar, pilih Availability Zone.

  B. Subnet—Dalam daftar, pilih subnet dari Availability Zone yang yang dipilih untuk grup subnet ini.

  C. Pilih Tambahkan subnet.

4. Pilih Buat. Ketika grup subnet dibuat, maka akan dicantumkan dengan grup subnet Anda lainnya.



Using the AWS CLI

Sebelum Anda dapat membuat grup subnet menggunakan AWS CLI, Anda harus terlebih dahulu menentukan subnet mana yang tersedia. Jalankan AWS CLI operasi berikut untuk membuat daftar Availability Zones dan subnetnya.

Parameter:

- **`--db-subnet-group`**—Opsional. Menentukan grup subnet tertentu akan mencantumkan Availability Zone dan subnet untuk grup tersebut. Menghilangkan parameter ini akan mencantumkan Availability Zone dan subnet untuk semua grup subnet Anda. Menentukan grup subnet `default` akan mencantumkan semua subnet VPC.

Example

Untuk Linux, macOS, atau Unix:

```
aws docdb describe-db-subnet-groups \
    --db-subnet-group-name default \
    --query 'DBSubnetGroups[*].[DBSubnetGroupName,Subnets[*].
[SubnetAvailabilityZone.Name,SubnetIdentifier]]'
```

Untuk Windows:

```
aws docdb describe-db-subnet-groups ^
    --db-subnet-group-name default ^
    --query 'DBSubnetGroups[*].[DBSubnetGroupName,Subnets[*].
[SubnetAvailabilityZone.Name,SubnetIdentifier]]'
```

Keluaran dari operasi ini terlihat seperti berikut ini (format 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"
            ]
        ]
    ]
]
```

Dengan menggunakan keluaran dari operasi sebelumnya, Anda dapat membuat grup subnet baru. Grup subnet baru harus mencakup subnet dari setidaknya dua Availability Zone.

Parameter:

- **`--db-subnet-group-name`**—Wajib. Nama untuk grup subnet ini.
- **`--db-subnet-group-description`**—Wajib. Deskripsi dari grup subnet ini.
- **`--subnet-ids`**—Wajib. Daftar subnet untuk disertakan dalam grup subnet ini. Contoh:`subnet-53ab3636`.
- --Tanda—Opsional. Daftar tanda (pasangan nilai kunci) untuk dilampirkan ke grup subnet ini.

Kode berikut ini membuat `sample-subnet-group` grup subnet dengan tiga subnet, `subnet-4e26d263`, `subnet-afc329f4`, dan `subnet-b3806e8f`.

Untuk Linux, macOS, atau 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
```

Untuk 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
```

Keluaran dari operasi ini terlihat seperti berikut ini (format 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"
    }
}
```

Menjelaskan grup subnet Amazon DocumentDB

Anda dapat menggunakan Konsol Manajemen AWS atau AWS CLI untuk mendapatkan rincian grup subnet Amazon DocumentDB.

Using the Konsol Manajemen AWS

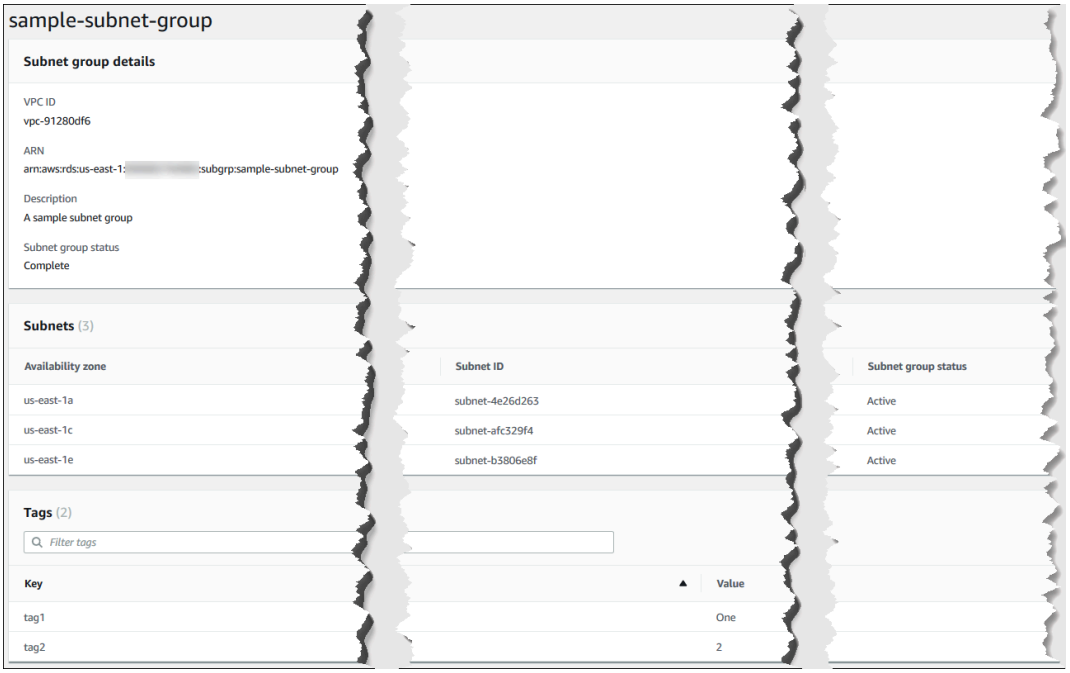
Prosedur berikut ini menunjukkan kepada Anda cara untuk mendapatkan detail grup subnet Amazon DocumentDB.

Untuk menemukan detail grup subnet

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Grup Subnet.

> **Tip**
>
> Jika Anda tidak melihat panel navigasi pada sisi kiri layar Anda, pilih ikon menu (≡) di sudut kiri atas halaman.

3. Untuk melihat detail grup subnet, pilih nama grup subnet tersebut.



Using the AWS CLI

Untuk menemukan detail grup subnet Amazon DocumentDB, gunakan operasi `describe-db-subnet-groups` dengan parameter berikut ini.

Parameter

- `--db-subnet=group-name`—Opsional. Jika disertakan, detail untuk grup subnet yang diberi nama dicantumkan. Jika dihilangkan, detail untuk hingga 100 grup subnet dicantumkan.

Example

Kode berikut ini mencantumkan detail untuk grup subnet `sample-subnet-group` yang kita buat di bagian Membuat grup subnet Amazon DocumentDB.

Untuk Linux, macOS, atau Unix:

```
aws docdb describe-db-subnet-groups \
    --db-subnet-group-name sample-subnet-group
```

Untuk Windows:

```
aws docdb describe-db-subnet-groups ^
    --db-subnet-group-name sample-subnet-group
```

Keluaran dari operasi ini terlihat seperti berikut ini (format 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"
```

```
    }
}
```

Memodifikasi grup subnet Amazon DocumentDB

Anda dapat menggunakan Konsol Manajemen AWS atau AWS CLI untuk memodifikasi deskripsi grup subnet atau untuk menambah atau menghapus subnet dari grup subnet Amazon DocumentDB. Namun demikian, Anda tidak dapat mengubah grup subnet `default`.

Using the Konsol Manajemen AWS

Anda dapat menggunakan Konsol Manajemen AWS untuk mengubah deskripsi grup subnet atau untuk menambah dan menghapus subnet. Ingat bahwa setelah selesai, Anda harus memiliki setidaknya dua Availability Zone yang berkaitan dengan grup subnet Anda.

Untuk memodifikasi grup subnet Anda

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Grup Subnet. Kemudian pilih tombol di sebelah kiri nama grup subnet. Ingat bahwa Anda tidak dapat memodifikasi grup subnet `default`.

   > **Tip**
   >
   > Jika Anda tidak melihat panel navigasi pada sisi kiri layar Anda, pilih ikon menu (≡) di sudut kiri atas halaman.

3. Pilih Tindakan, dan kemudian pilih Modifikasi.
4. Deskripsi—Untuk mengubah deskripsi grup subnet Anda, masukkan deskripsi baru.
5. Untuk mengubah subnet yang berkaitan dengan grup subnet Anda, di bagian Tambahkan subnet, lakukan salah satu atau lebih dari hal berikut ini:
   - Untuk menghapus semua subnet dari grup subnet ini, pilih Hapus semua.
   - Untuk menghapus subnet tertentu dari grup subnet ini, pilih Hapus untuk setiap subnet yang ingin Anda hapus.
   - Untuk menambahkan semua subnet yang berkaitan dengan VPC ini, pilih Tambahkan semua subnet yang berkaitan dengan VPC ini.

- Untuk menambahkan subnet tertentu ke grup subnet ini, lakukan hal berikut ini untuk setiap Availability Zone tempat Anda ingin menambahkan subnet.

  a. Availability zone—Dalam daftar, pilih Availability Zone baru.

  b. Subnet—Dalam daftar, pilih subnet dari Availability Zone yang yang dipilih untuk grup subnet ini.

  c. Pilih Tambahkan subnet.

6. Di kotak dialog konfirmasi:
   - Untuk membuat perubahan tersebut ke grup subnet, pilih Modifikasi.
   - Untuk menyimpan grup subnet agar tidak berubah, pilih Batal.

Using the AWS CLI

Anda dapat menggunakan AWS CLI untuk mengubah deskripsi grup subnet atau untuk menambah dan menghapus subnet. Ingat bahwa setelah selesai, Anda harus memiliki setidaknya dua Availability Zone yang berkaitan dengan grup subnet Anda. Anda tidak dapat memodifikasi grup subnet `default`.

Parameter:

- `--db-subnet-group-name`—Wajib. Nama grup subnet Amazon DocumentDB yang Anda modifikasi.
- `--subnet-ids`—Wajib. Daftar semua subnet yang Anda inginkan dalam grup subnet setelah perubahan ini dilakukan.

  > **Important**
  >
  > Setiap subnet yang saat ini berada dalam grup subnet yang tidak disertakan dalam daftar ini akan dihapus dari grup subnet. Jika Anda ingin menyimpan salah satu subnet yang saat ini dalam grup subnet, Anda harus menyertakannya dalam daftar ini.

- `--db-subnet-group-description`—Opsional. Deskripsi grup subnet.

Example

Kode berikut memodifikasi deskripsi dan menggantikan subnet yang ada dengan subnet `subnet-991cb8d0`, `subnet-53ab3636`, dan `subnet-29ab1025`.

Untuk Linux, macOS, atau 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"
```

Untuk 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"
```

Keluaran dari operasi ini terlihat seperti berikut ini (format JSON). Perhatikan bahwa ini adalah grup subnet yang sama yang dibuat di bagian Membuat grup subnet Amazon DocumentDB. Namun demikian, subnet dalam grup subnet diganti dengan yang tercantum dalam operasi `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"
    }
}
```

Menghapus grup subnet Amazon DocumentDB

Anda dapat menggunakan Konsol Manajemen AWS atau AWS CLI untuk menghapus grup subnet Amazon DocumentDB. Namun demikian, Anda tidak dapat menghapus grup subnet `default`.

Using the Konsol Manajemen AWS

Anda dapat menggunakan Konsol Manajemen AWS untuk menghapus grup subnet. Namun Anda tidak dapat menghapus grup subnet `default`.

Untuk menghapus grup subnet

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Grup Subnet. Kemudian pilih tombol di sebelah kiri nama grup subnet. Ingat bahwa Anda tidak dapat menghapus grup subnet `default`.

   > **Tip**
   >
   > Jika Anda tidak melihat panel navigasi pada sisi kiri layar Anda, pilih ikon menu (≡) di sudut kiri atas halaman.

3. Pilih Tindakan, lalu pilih Hapus.
4. Di kotak dialog konfirmasi:
   - Untuk menghapus grup subnet, pilih Hapus.
   - Untuk menyinmpan grup subnet, pilih Batal.

Using the AWS CLI

Untuk menghapus grup subnet Amazon DocumentDB menggunakan, gunakan operasi AWS CLI dengan parameter `delete-db-subnet-group` berikut.

Parameter

- `--db-subnet-group-name`—Wajib. Nama grup subnet Amazon DocumentDB yang akan dihapus. Ingat bahwa Anda tidak dapat menghapus grup subnet `default`.

Example

Kode berikut ini akan menghapus `sample-subnet-group`.

Untuk Linux, macOS, atau Unix:

```
aws docdb delete-db-subnet-group \
    --db-subnet-group-name sample-subnet-group
```

Untuk Windows:

```
aws docdb delete-db-subnet-group ^
    --db-subnet-group-name sample-subnet-group
```

Operasi ini tidak menghasilkan keluaran.

Amazon DocumentDB Ketersediaan dan replikasi tinggi

Anda dapat mencapai ketersediaan tinggi dan membaca penskalaan di Amazon DocumentDB (dengan kompatibilitas MongoDB) dengan menggunakan instans replika. Klaster Amazon DocumentDB tunggal mendukung instans primer tunggal dan hingga 15 instans replika. Instans tersebut dapat didistribusikan di seluruh Availability Zone di dalam Wilayah klaster. Instans primer menerima lalu lintas baca dan tulis, dan instans replika hanya menerima permintaan baca.

Volume kluster dibuat dari beberapa salinan data untuk kluster. Namun demikian, data dalam volume klaster direpresentasikan sebagai volume tunggal yang logis ke instans primer dan replika Amazon DocumentDB dalam klaster. Instans replika pada akhirnya konsisten. Mereka mengembalikan hasil kueri dengan sedikit penundaan replika—biasanya kurang dari 100 milidetik setelah instans primer menulis pembaruan. Lag replika bervariasi bergantung pada laju perubahan basis data. Artinya,

selama periode di mana sejumlah besar operasi tulis terjadi untuk basis data, Anda mungkin melihat peningkatan lag replika.

Penskalaan baca

Replika Amazon DocumentDB berfungsi dengan baik untuk penskalaan baca karena didedikasikan sepenuhnya untuk membaca operasi pada volume klaster Anda. Operasi tulis dikelola oleh instans primer. Volume klaster dibagikan di semua instans di klaster Anda. Oleh karena itu, Anda tidak perlu mereplikasi dan mempertahankan salinan data untuk setiap replika Amazon DocumentDB.

Ketersediaan tinggi

Saat Anda membuat klaster Amazon DocumentDB, bergantung pada jumlah Availability Zone dalam grup subnet (harus ada setidaknya dua), Amazon DocumentDB menyediakan instans di seluruh Availability Zone. Ketika Anda membuat instans dalam klaster, Amazon DocumentDB secara otomatis mendistribusikan instans di seluruh Availability Zone dalam grup subnet untuk menyeimbangkan klaster. Tindakan ini juga mencegah semua instans diletakkan di Availability Zone yang sama.

Contoh

Untuk menggambarkan poin, pertimbangkan contoh di mana Anda membuat klaster yang memiliki grup subnet dengan tiga Availability Zone: AZ1, AZ2, dan AZ3.

Ketika instans pertama dalam klaster dibuat, ini adalah instans primer dan terletak di salah satu Availability Zone. Dalam contoh ini, instans berada di AZ1. Instans kedua yang dibuat adalah instans replika dan terletak di salah satu dari dua Availability Zone lainnya, yaitu AZ2. Instans ketiga yang dibuat adalah instans replika dan terletak di Availability Zone yang tersisa, AZ3. Jika Anda membuat lebih banyak instans, mereka didistribusikan di seluruh Availability Zone sehingga Anda mencapai keseimbangan dalam klaster.

Jika terjadi kegagalan dalam instans primer (AZ1), failover dipicu, dan salah satu replika yang ada dipromosikan ke primer. Ketika primer yang lama pulih, maka menjadi replika di Availability Zone yang sama di mana replika tersebut disediakan (AZ1). Ketika Anda menyediakan tiga klaster instans, Amazon DocumentDB terus menjaga tiga instans klaster tersebut. Amazon DocumentDB secara otomatis menangani deteksi, failover, dan pemulihan kegagalan instans tanpa intervensi manual apa pun.

Ketika Amazon DocumentDB melakukan failover dan memulihkan instans, instans yang dipulihkan tetap di Availability Zone di mana instans tersebut disediakan pada awalnya. Namun demikian, peran

instans mungkin berubah dari primer ke replika. Melakukan hal ini akan mencegah skenario di mana serangkaian failover dapat mengakibatkan semua instans berada di Availability Zone yang sama.

Anda dapat menentukan replika Amazon DocumentDB sebagai target failover. Artinya, jika instans primer gagal, replika Amazon DocumentDB yang ditentukan atau replika dari tingkat dipromosikan menjadi instans primer. Terdapat gangguan singkat selama permintaan baca dan tulis dibuat ke instans primer gagal dengan pengecualian. Jika klaster Amazon DocumentDB Anda tidak mencakup replika Amazon DocumentDB apa pun, ketika instans primer gagal, replika tersebut kembali dibuat. Mempromosikan replika Amazon DocumentDB jauh lebih cepat daripada membuat ulang instans primer.

Untuk skenario dengan ketersediaan tinggi, kami sarankan Anda membuat satu atau lebih replika Amazon DocumentDB. Replika tersebut harus berasal dari kelas instans yang sama dengan instans primer dan dalam Availability Zone yang berbeda untuk klaster Amazon DocumentDB Anda.

Untuk informasi selengkapnya, lihat berikut ini:

- Memahami toleransi kesalahan klaster Amazon DocumentDB
- Failover Amazon DocumentDB
  - Mengontrol target failover

Di Amazon DocumentDB versi 5.0 dan 8.0, instance replika tidak dimulai ulang saat instans utama dimulai ulang. Mereka terus melayani permintaan baca selama restart instance utama, menjaga ketersediaan yang lebih baik.

Ketersediaan tinggi dengan cluster global

Untuk ketersediaan tinggi di beberapa Wilayah AWS, Anda dapat mengatur klaster global Amazon DocumentDB. Setiap cluster global mencakup beberapa wilayah, memungkinkan pembacaan global latensi rendah dan pemulihan bencana dari pemadaman di seluruh wilayah. Wilayah AWS Amazon DocumentDB secara otomatis menangani replikasi semua data dan pembaruan dari wilayah utama ke masing-masing wilayah sekunder.

Menambahkan replika

Instans pertama yang ditambahkan ke klaster adalah instans primer. Setiap instans yang ditambahkan setelah instans pertama adalah instans replika. Klaster dapat memiliki hingga 15 instans replika di samping instans primer.

Ketika Anda membuat cluster menggunakan Konsol Manajemen AWS, instance utama secara otomatis dibuat pada saat yang sama. Untuk membuat replika pada saat yang sama seperti saat Anda membuat klaster dan instans primer, pilih Membuat replika di zona yang berbeda. Untuk informasi lebih lanjut, lihat langkah 4.d. di Membuat cluster Amazon DocumentDB. Untuk menambahkan lebih banyak replika untuk klaster Amazon DocumentDB, lihat Menambahkan instance Amazon DocumentDB ke klaster.

Saat menggunakan AWS CLI untuk membuat cluster Anda, Anda harus secara eksplisit membuat instance primer dan replika Anda. Untuk informasi lebih lanjut, lihat bagian “Menggunakan AWS CLI" dalam topik berikut:

- Membuat cluster Amazon DocumentDB
- Menambahkan instance Amazon DocumentDB ke klaster

Failover Amazon DocumentDB

Dalam kasus tertentu, seperti jenis tertentu dari pemeliharaan yang direncanakan, atau peristiwa simpul primer yang tidak dimungkinkan node utama atau kegagalan Availability Zone, Amazon DocumentDB (dengan kompatibilitas MongoDB) mendeteksi kegagalan dan menggantikan simpul primer. Selama failover, penulisan waktu diminimalkan. Hal ini karena peran simpul primer tidak berhasil ke salah satu replika baca alih-alih harus membuat dan menyediakan simpul primer baru. Deteksi kegagalan dan promosi replika ini memastikan bahwa Anda dapat melanjutkan penulisan ke primer baru segera setelah promosi selesai.

Agar failover berfungsi, klaster Anda harus memiliki setidaknya dua instans — primer dan setidaknya satu instans replika.

> **Note**
>
> Topik ini hanya berlaku untuk cluster berbasis instans Amazon DocumentDB asli. Ini tidak berlaku untuk cluster elastis atau global.

Mengontrol target failover

Amazon DocumentDB menyediakan Anda dengan tingkatan failover sebagai sarana untuk mengontrol instans replika mana yang dipromosikan ke primer ketika terjadi failover.

Tingkatan Failover

Setiap instans replika berkaitan dengan tingkatan failover (0-15). Ketika failover terjadi akibat pemeliharaan atau kegagalan perangkat keras yang tidak dimungkinkan, instan utama tidak berhasil menjadi replika dengan prioritas tertinggi (tingkatan bernomor terendah). Jika beberapa replika memiliki tingkatan prioritas yang sama, primer tidak berhasil menjadi replika tingkatan tersebut yang paling dekat dalam ukuran primer sebelumnya.

Dengan menetapkan tingkatan failover untuk grup pilih replika menjadi `0` (prioritas tertinggi), Anda dapat memastikan bahwa failover akan mempromosikan salah satu replika dalam grup tersebut. Anda dapat secara efektif mencegah replika spesifik yang dipromosikan ke primer dalam kasus failover dengan menetapkan tingkatan prioritas rendah (nomor tinggi) untuk replika tersebut. Hal ini berguna dalam kasus di mana replika spesifik menerima penggunaan berat oleh aplikasi dan ketidakberhasilan untuk salah satu dari mereka akan berdampak negatif pada aplikasi kritis.

Anda dapat mengatur tingkatan failover instans ketika Anda membuatnya atau kemudian dengan memodifikasinya. Menetapkan tingkatan failover instans dengan memodifikasi instans tidak memicu failover. Untuk informasi selengkapnya lihat topik berikut:

- Menambahkan instance Amazon DocumentDB ke klaster
- Memodifikasi instance Amazon DocumentDB

Ketika secara manual menginisiasi failover, Anda memiliki dua cara untuk mengontrol instans replika yang dipromosikan ke primer: tingkatan failover seperti yang diuraiakn sebelumnya, dan parameter `--target-db-instance-identifier`.

-- **`target-db-instance-identifier`**

Untuk pengujian, Anda dapat memaksa peristiwa failover menggunakan operasi `failover-db-cluster`. Anda dapat menggunakan parameter `--target-db-instance-identifier` untuk menentukan replika mana yang akan dipromosikan ke primer. Menggunakan parameter `--target-db-instance-identifier` akan menggantikan tingkatan prioritas failover. Jika Anda tidak menentukan parameter `--target-db-instance-identifier`, failover primer adalah sesuai dengan tingkatan prioritas failover.

Apa yang terjadi selama failover

Failover secara otomatis ditangani oleh Amazon DocumentDB sehingga aplikasi Anda dapat melanjutkan operasi basis data secepat mungkin tanpa intervensi administratif.

- Jika Anda memiliki instans replika Amazon DocumentDB di Availability Zone yang sama atau berbeda ketika terjadi failover: Amazon DocumentDB membalik catatan nama kanonik (CNAME) untuk instans Anda untuk menunjuk pada replika sehat, yang, pada gilirannya, dipromosikan untuk menjadi primer baru. Failover biasanya selesai dalam waktu 30 detik dari awal sampai akhir.
- Jika Anda tidak memiliki instans replika Amazon DocumentDB (sebagai contoh, klaster instans tunggal): Amazon DocumentDB akan mencoba untuk membuat instans baru di Availability Zone yang sama seperti instans asli. Penggantian instans asli ini dilakukan atas dasar upaya terbaik dan mungkin tidak berhasil jika, sebagai contoh, terdapat masalah yang secara luas memengaruhi Availability Zone.

Aplikasi Anda harus mencoba kembali koneksi basis data dalam peristiwa kehilangan koneksi.

Menguji failover

Failover untuk klaster mempromosikan salah satu replika Amazon DocumentDB (instans baca-saja) di klaster menjadi instans primer (penulis klaster).

Ketika instans primer gagal, Amazon DocumentDB secara otomatis melakukan failover ke replika Amazon DocumentDB, jika ada. Anda dapat memaksa failover saat Anda ingin menyimulasikan kegagalan instans primer untuk pengujian. Setiap instans dalam klaster memiliki alamat titik akhir sendiri. Oleh karena itu, Anda perlu membersihkan dan membentuk kembali koneksi yang sudah ada yang menggunakan titik akhir tersebut yang ditujukan saat failover selesai.

Untuk memaksa failover, gunakan operasi `failover-db-cluster` dengan parameter tersebut.

- `--db-cluster-identifier`—Wajib. Nama kluster yang akan di-failover.
- `--target-db-instance-identifier`—Opsional. Nama instans yang akan dipromosikan ke instans primer.

Example

Operasi berikut ini memaksa failover kluster `sample-cluster`. Operasi tersebut tidak menentukan instans mana yang akan menjadi instans primer baru, sehingga Amazon DocumentDB memilih instans sesuai dengan prioritas tingkatan failover.

Untuk Linux, macOS, atau Unix:

```
aws docdb failover-db-cluster \
```

```
    --db-cluster-identifier sample-cluster
```

Untuk Windows:

```
aws docdb failover-db-cluster ^
    --db-cluster-identifier sample-cluster
```

Operasi berikut ini memaksa failover klaster `sample-cluster`, yang menentukan bahwa `sample-cluster-instance` akan dipromosikan menjadi peran primer. (Perhatikan `"IsClusterWriter": true` dalam keluaran.)

Untuk Linux, macOS, atau Unix:

```
aws docdb failover-db-cluster \
    --db-cluster-identifier sample-cluster \
    --target-db-instance-identifier sample-cluster-instance
```

Untuk Windows:

```
aws docdb failover-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --target-db-instance-identifier sample-cluster-instance
```

Output dari operasi ini terlihat seperti berikut (format 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"
    }
}
```

Replikasi lag

Lag replikasi biasanya 50ms atau kurang. Alasan paling umum untuk peningkatan lag replika adalah:

- Tingkat tulis yang tinggi pada primer yang menyebabkan replika baca jatuh di belakang primer.
- Pertikaian pada replika baca antara kueri yang berjalan lama (misalnya, pindaian sekuensial besar, kueri agregasi) dan replikasi tulis yang masuk.
- Jumlah kueri bersamaan yang sangat besar pada replika baca.

Untuk meminimalkan lag replikasi, cobalah teknik pemecahan masalah ini:

- Jika Anda memiliki tingkat tulis tinggi atau utilisasi CPU yang tinggi, kami sarankan Anda meningkatkan instans di klaster Anda.
- Jika ada kueri yang berjalan lama pada replika baca Anda, dan pembaruan yang sangat sering pada dokumen yang ditanyakan, pertimbangkan untuk mengubah kueri Anda yang sudah berjalan lama, atau menjalankannya terhadap replika untuk menghindari pertengkaran pada primary/write replika baca.
- Jika terdapat jumlah kueri bersamaan yang sangat besar atau utilisasi CPU yang tinggi hanya pada replika baca, pilihan lainnya adalah untuk menskalakan keluar jumlah replika baca untuk menyebarkan beban kerja.
- Karena kelambatan replikasi adalah hasil dari throughput penulisan yang tinggi dan kueri yang berjalan lama, kami merekomendasikan pemecahan masalah lag replikasi dengan memanfaatkan metrik DBClusterReplicaLagMaximum CW dalam kombinasi dengan logger kueri lambat dan metrik/. `WriteThroughput` `WriteIOPS`

Secara umum, kami menyarankan bahwa semua replika Anda adalah dari tipe instans yang sama, sehingga failover klaster tidak akan menyebabkan penurunan performa.

Jika Anda memilih antara menaikkan skala dan menskalakan keluar (misalnya enam instans yang lebih kecil vs tiga instans yang lebih besar), kami umumnya merekomendasikan untuk pertama-tama mencoba menaikkan skala (instans yang lebih besar) sebelum menskalakan keluar, karena Anda akan mendapatkan cache buffer yang lebih besar per instans DB.

Secara proaktif, Anda harus mengatur alarm lag replikasi dan mengatur ambang batasnya untuk nilai yang Anda rasa merupakan batas atas untuk seberapa jauh dapat meninggalkan (atau “menahan”) data Anda pada instans replika sebelum mulai memengaruhi fungsionalitas aplikasi Anda. Secara

umum, kami akan menyarankan bahwa ambang batas lag replikasi terlampaui untuk beberapa titik data sebelum menciptakan alarm, akibat beban kerja sementara.

> **Note**
>
> Selain itu, kami sarankan Anda mengatur alarm lainnya untuk lag replikasi yang melebihi 10 detik. Jika Anda melampaui ambang batas ini untuk beberapa titik data, kami sarankan Anda menaikkan skala instans Anda atau mengurangi throughput tulis Anda pada instans primer.

Mengelola indeks Amazon DocumentDB

Topik

- Pembuatan indeks Amazon DocumentDB
- Mempertahankan indeks Amazon DocumentDB

Pembuatan indeks Amazon DocumentDB

Membangun indeks di Amazon DocumentDB memerlukan sejumlah keputusan yang harus dibuat:

- Seberapa cepat itu harus diselesaikan?
- Bisakah koleksi tidak dapat diakses saat build sedang berlangsung?
- Berapa banyak daya komputasi instans yang dapat dialokasikan ke build?
- Jenis indeks apa yang harus dibuat?

Bagian ini membantu Anda menjawab pertanyaan-pertanyaan ini dan memberikan perintah dan contoh pemantauan untuk membuat indeks Amazon DocumentDB pada koleksi cluster berbasis instans Anda.

Pedoman

Pedoman berikut mencakup batasan dasar dan pengorbanan konfigurasi saat membuat indeks baru:

- Dukungan versi Amazon DocumentDB - Meskipun pengindeksan pekerja tunggal didukung di semua versi Amazon DocumentDB, beberapa pengindeksan pekerja hanya didukung di Amazon DocumentDB versi 4.0, 5.0, dan 8.0.

- Trade-off kinerja - Meningkatkan jumlah pekerja dalam proses pembuatan indeks meningkatkan pemanfaatan CPU dan membaca IO pada instance utama database Amazon DocumentDB Anda. Sumber daya yang dibutuhkan untuk membuat indeks baru tidak akan tersedia untuk beban kerja Anda yang sedang berjalan.
- Cluster elastis - Pengindeksan paralel tidak didukung pada kluster elastis Amazon DocumentDB.
- Pekerja maksimum - Jumlah maksimum pekerja yang dapat Anda konfigurasi tergantung pada ukuran instance utama Anda di cluster database Anda. Ini adalah setengah dari jumlah total vCPU pada instance utama cluster database Anda. Misalnya, Anda dapat menjalankan maksimal 32 pekerja pada instance db.r6g.16xlarge yang memiliki 64 vCPU.

  > **Note**
  >
  > Pekerja paralel tidak didukung pada kelas instance 2xlarge dan lebih rendah.

- Pekerja minimum - Jumlah minimum pekerja yang dapat Anda konfigurasikan adalah satu. Pengaturan default untuk pembuatan indeks pada cluster berbasis instance adalah dua pekerja. Namun, Anda dapat mengurangi jumlah pekerja menjadi satu dengan menggunakan opsi “utas pekerja”. Ini akan menjalankan proses dengan satu pekerja.
- Kompresi indeks - Amazon DocumentDB tidak mendukung kompresi indeks. Ukuran data untuk indeks mungkin lebih besar daripada saat Anda menggunakan opsi lain.
- Mengindeks beberapa koleksi - Setengah dari vCPU pada instance utama cluster database Anda dapat digunakan untuk pekerja yang dikonfigurasi yang melakukan pembuatan indeks pada beberapa koleksi.
- Jenis indeks - Lihat posting blog ini untuk penjelasan lengkap tentang jenis indeks yang didukung di Amazon DocumentDB.

Memulai

Untuk memulai pembuatan indeks pada koleksi, gunakan `createIndexes` perintah. Secara default, perintah akan menjalankan dua pekerja paralel yang meningkatkan kecepatan proses pembuatan indeks sebanyak dua kali.

Misalnya, proses perintah berikut menunjukkan cara membuat indeks untuk bidang “user_name” dalam dokumen dan meningkatkan kecepatan proses pengindeksan menjadi empat pekerja:

1. Buat indeks menggunakan dua pekerja paralel di cluster:

```
db.runCommand({"createIndexes":"test","indexes":[{"key": {"user_name":1},
 "name":"username_idx"}]})
```

2. Untuk mengoptimalkan kecepatan proses pembuatan indeks, Anda dapat menentukan jumlah pekerja dengan menggunakan opsi “thread pekerja” (`"workers":<number>`) dalam `db.runCommand createIndexes` perintah.

   Tingkatkan kecepatan proses menjadi empat pekerja paralel:

   ```
   db.runCommand({"createIndexes":"test","indexes":[{"key": {"user_name":1},
    "name":"username_idx", "workers":4}]})
   ```

   > **Note**
   >
   > Semakin tinggi jumlah pekerja, semakin cepat pembuatan indeks berlangsung. Namun, semakin tinggi jumlah pekerja meningkat, semakin tinggi beban meningkat pada vCPU dan membaca IO dari instance utama Anda. Pastikan klaster Anda cukup disediakan untuk menangani beban yang meningkat tanpa menurunkan beban kerja lainnya.

Status kemajuan pengindeksan

Proses pembuatan indeks bekerja dengan menginisialisasi, memindai koleksi, menyortir kunci, dan, akhirnya, memasukkan kunci melalui pembuat indeks. Prosesnya memiliki hingga enam tahap saat Anda menjalankannya di latar depan, dan hingga sembilan tahap saat Anda menjalankannya di latar belakang. Anda dapat melihat metrik status seperti penyelesaian persentase, jumlah total blok penyimpanan yang dipindai, kunci yang diurutkan, dan kunci yang disisipkan berdasarkan tahap demi tahap.

Pantau kemajuan proses pengindeksan dengan menggunakan `db.currentOp()` perintah di shell mongo. Penyelesaian 100% dari tahap terakhir menunjukkan bahwa semua indeks telah berhasil dibuat:

```
db.currentOp({"command.createIndexes": { $exists : true } })
```

> **Note**
>
> Melihat status kemajuan pengindeksan hanya didukung di Amazon DocumentDB 5.0.

Jenis build indeks

Empat jenis build indeks adalah:

- Foreground — Build indeks latar depan memblokir semua operasi database lainnya hingga indeks dibuat. Build latar depan Amazon DocumentDB terdiri dari lima tahap.
- Foreground (unique) - Build indeks latar depan dokumen tunggal (unik) memblokir operasi database lain seperti build latar depan biasa. Berbeda dengan build latar depan dasar, build unik menggunakan tahap tambahan (kunci pengurutan 2) untuk mencari kunci duplikat. Build latar depan (unik) terdiri dari enam tahap.
- Latar Belakang - Build indeks latar belakang memungkinkan operasi database lainnya berjalan di latar depan saat indeks sedang dibuat. Build latar belakang Amazon DocumentDB terdiri dari delapan tahap.
- Latar belakang (unik) - Build indeks latar belakang dokumen tunggal (unik) memungkinkan operasi database lainnya berjalan di latar depan saat indeks sedang dibuat. Tidak seperti build latar belakang dasar, build unik menggunakan tahap tambahan (kunci pengurutan 2) untuk mencari kunci duplikat. Build latar belakang (unik) terdiri dari sembilan tahap.

Tahapan pembuatan indeks

Stage	Latar depan	Latar depan (unik)	Latar Belakang	Latar belakang (unik)
Inisialisasi	1	1	1	1
indeks bangunan: menginisialisasi	2	2	2	2
indeks bangunan:	3	3	3	3

Stage	Latar depan	Latar depan (unik)	Latar Belakang	Latar belakang (unik)
koleksi pemindaian				
indeks bangunan: kunci penyortiran 1	4	4	4	4
indeks bangunan: kunci penyortiran 2		5		5
indeks bangunan: memasukkan kunci	5	6	5	6
memvalidasi: indeks pemindaian			6	7
memvalidasi: menyortir tupel			7	8
memvalidasi: koleksi pemindaian			8	9

- inisialisasi - createIndex sedang mempersiapkan pembuat indeks. Fase ini harus sangat singkat.
- indeks bangunan: menginisialisasi - Pembangun indeks sedang mempersiapkan untuk membuat indeks. Fase ini harus sangat singkat.
- indeks bangunan: koleksi pemindaian - Pembuat indeks melakukan pemindaian koleksi untuk mengumpulkan kunci indeks. Satuan ukuran adalah “blok”.

> **Note**
>
> Jika lebih dari satu pekerja dikonfigurasi untuk build indeks, itu akan ditampilkan di tahap ini. Tahap “pengumpulan pemindaian” adalah satu-satunya tahap yang menggunakan banyak pekerja selama proses pembuatan indeks. Semua tahapan lainnya akan menampilkan satu pekerja.

- indeks bangunan: kunci penyortiran 1 - Pembuat indeks menyortir kunci indeks yang dikumpulkan. Satuan ukuran adalah “kunci”.
- indeks bangunan: kunci pengurutan 2 - Pembuat indeks menyortir kunci indeks yang dikumpulkan yang sesuai dengan tupel mati. Fase ini hanya ada untuk bangunan indeks yang unik. Satuan ukuran adalah “kunci”.
- indeks bangunan: memasukkan kunci - Pembuat indeks memasukkan kunci indeks ke dalam indeks baru. Satuan ukuran adalah “kunci”.
- memvalidasi: indeks pemindaian - createIndex memindai indeks untuk menemukan kunci yang perlu divalidasi. Satuan ukuran adalah “blok”.
- memvalidasi: menyortir tupel - createIndex menyortir output dari fase pemindaian indeks.
- memvalidasi: koleksi pemindaian - createIndex memindai koleksi untuk memvalidasi kunci indeks yang ditemukan dalam dua fase sebelumnya. Satuan ukuran adalah “blok”.

Contoh keluaran pembuatan indeks

Dalam contoh keluaran berikut (membangun indeks latar depan), status pembuatan indeks ditampilkan. Bidang “msg” merangkum kemajuan build dengan menunjukkan tahapan dan persentase penyelesaian build. Bidang “pekerja” menunjukkan jumlah pekerja yang digunakan selama tahap pembuatan indeks tersebut. Bidang “kemajuan” menunjukkan angka aktual yang digunakan untuk menghitung persentase penyelesaian.

> **Note**
>
> Bidang “saat iniIndexBuildName”, “msg”, dan “kemajuan” tidak didukung di Amazon DocumentDB versi 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
}
```

Mempertahankan indeks Amazon DocumentDB

Topik

- Bloat indeks
- Pemeliharaan indeks menggunakan reIndex

Bloat indeks

Amazon DocumentDB Multi-Version menggunakan Concurrency Control (MVCC) untuk mengelola transaksi bersamaan. Ketika dokumen dihapus atau diperbarui, versi sebelumnya tetap dalam koleksi dan indeks sebagai versi “mati”. Proses pengumpulan sampah secara otomatis merebut kembali ruang dari versi mati ini untuk operasi masa depan.

Indeks kembung terjadi ketika indeks koleksi menjadi lebih besar karena akumulasi entri indeks mati atau usang atau fragmentasi dalam halaman. Persentase yang dilaporkan mewakili jumlah ruang indeks yang dapat digunakan oleh entri indeks future. Bloat ini menghabiskan ruang di cache buffer dan penyimpanan. Jika Anda ingin menghapus kembung, Anda perlu membangun kembali indeks.

Example Contoh

Jalankan perintah berikut untuk menentukan penyimpanan yang tidak digunakan untuk indeks Anda:

```
db.coll.aggregate({$indexStats:{}});
```

Hasilnya terlihat mirip dengan ini:

```
{
    "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
    }
}
```

Anda dapat membangun kembali indeks tanpa downtime menggunakan `reIndex` perintah, yang memerlukan pemindaian seluruh koleksi. Lihat Pemeliharaan indeks menggunakan `reIndex`.

Pemeliharaan indeks menggunakan `reIndex`

`reIndex`adalah perintah yang digunakan untuk membangun kembali indeks. Hal ini biasanya digunakan ketika indeks telah menjadi rusak atau tidak efisien. Seiring waktu, indeks dapat mengakumulasi ruang yang tidak terpakai karena banyak pembaruan, penyisipan, atau penghapusan, yang menyebabkan kinerja menurun. Pengindeksan ulang membantu menghilangkan ruang yang tidak terpakai tersebut dan mengembalikan efisiensi indeks.

`Pedoman ReIndex`

- `reIndex`hanya didukung di Amazon DocumentDB 5.0.
- Amazon DocumentDB `reindex` mendukung indeks tunggal di latar belakang, memungkinkan untuk beberapa pekerja. Indeks lama dapat digunakan oleh kueri saat `reIndex` proses berjalan.
- Amazon DocumentDB mendukung pengindeksan laporan kemajuan melalui. `currentOp` Anda dapat melihat tahapan pembuatan indeks yang mirip dengan yang Tahapan pembuatan indeks dilihat selama pembuatan indeks. Satu-satunya perbedaan adalah bahwa `reIndex` selalu memiliki delapan tahap, terlepas dari apakah itu unik atau tidak. Tidak ada tahap “indeks bangunan: kunci pengurutan 2".
- `reIndex`dapat berjalan bersamaan dengan perintah apa pun kecuali perintah terkait indeks pada koleksi yang sama:`createIndexes`,,`dropIndexes`, `collMod` dan. `renameCollection`
- `reIndex`saat ini tidak didukung untuk teks, geospasial, vektor, dan indeks sebagian.

`Membangun reIndex`

Gunakan perintah berikut untuk membangun kembali indeks Anda:

```
db.runCommand({ reIndex: "collection-name", index: "index-name"})
```

Secara opsional, Anda juga dapat mengontrol jumlah pekerja yang ditugaskan untuk proses pembangunan kembali:

```
db.runCommand({ reIndex: "collection-name", index: "index-name", workers: number })
```

Mengelola kompresi dokumen tingkat koleksi

Kompresi dokumen tingkat koleksi Amazon DocumentDB 5.0 memungkinkan Anda menurunkan biaya penyimpanan dan IO dengan mengompresi dokumen dalam koleksi Anda. Anda dapat

mengaktifkan kompresi dokumen pada tingkat pengumpulan dan melihat metrik kompresi sesuai kebutuhan dengan mengukur keuntungan penyimpanan melalui metrik kompresi seperti ukuran penyimpanan dokumen terkompresi dan status kompresi. Amazon DocumentDB 5.0 menggunakan algoritma kompresi LZ4 untuk mengompres dokumen.

Amazon DocumentDB mendukung kompresi dokumen yang dimulai dengan versi 5.0. Berikut ini adalah fungsi kompresi dokumen tingkat koleksi:

- Perilaku default - Pengaturan kompresi default untuk koleksi baru pada klaster 5.0 ditentukan oleh parameter cluster`default_collection_compression`. Parameter ini diatur ke “dinonaktifkan” secara default.
- Mengompresi koleksi yang ada - Pengaturan kompresi untuk koleksi yang ada dapat diubah menggunakan `collMod` perintah.
- Mengubah ambang kompresi - Ambang kompresi default adalah 2KB. Nilai ini dapat ditentukan untuk koleksi baru menggunakan `createCollection` perintah, dan diubah untuk koleksi yang ada menggunakan `collMod` perintah.

> **Note**
>
> Kompresi dokumen Amazon DocumentDB tidak didukung di Amazon DocumentDB versi 3.6 dan 4.0.

Topik

- Mengelola kompresi dokumen
- Memantau kompresi dokumen

Mengelola kompresi dokumen

Mengaktifkan kompresi dokumen dalam koleksi

Aktifkan kompresi dokumen saat membuat koleksi di Amazon DocumentDB 5.0 dengan menggunakan metode: `db.createCollection()`

```
db.createCollection( sample_collection,{
    storageEngine : {
        documentDB: {
```

```
            compression:{enable: <true | false>}
        }
    }
})
```

Mengaktifkan kompresi dokumen dalam sebuah cluster

Kompresi dokumen dapat diaktifkan secara default untuk semua koleksi baru pada tingkat cluster dengan mengatur parameter cluster `default_collection_compression` ke “enabled”. Ketika parameter ini diatur ke “diaktifkan”, koleksi yang baru dibuat di cluster akan mengaktifkan kompresi secara default dengan ambang kompresi 2 KB.

Mengompresi koleksi yang ada

Anda juga dapat memodifikasi pengaturan kompresi untuk koleksi yang ada menggunakan `collMod` operasi dan menentukan `storageEngine` konfigurasi berikut. Perhatikan bahwa perubahan yang dilakukan menggunakan perintah ini hanya akan berlaku untuk inserted/updated dokumen baru dan kompresi pada dokumen yang dimasukkan sebelumnya tidak akan berubah.

```
db.runCommand({
    collMod: "orders",
    storageEngine: {
        documentDB: {compression: {enable: <true | false>} }
    }
})
```

Mengatur ambang kompresi

Secara default, ambang kompresi untuk koleksi terkompresi adalah 2032 byte. Nilai ambang batas ini dapat diatur dalam `createCollection` perintah saat membuat koleksi baru dengan kompresi diaktifkan:

```
db.createCollection( sample_collection, {
    storageEngine : {
        documentDB: {
            compression: {
                enable: true,
                threshold: <128 - 8000>
            }
        }
```

```
    }
})
```

Anda juga dapat memodifikasi ambang kompresi untuk koleksi terkompresi yang ada menggunakan `collMod` operasi dan menentukan konfigurasi berikut: `storageEngine`

```
db.runCommand({
    collMod: "orders",
    storageEngine: {
        documentDB: {
            compression: {
                enable: true,
                threshold: <128 - 8000>
            }
        }
    }
})
```

Perhatikan bahwa ambang kompresi hanya dapat diatur ke nilai antara 128 hingga 8000 byte. Selain itu, `enable` opsi perlu disetel ke “true” saat menentukan ambang kompresi.

Memantau kompresi dokumen

Anda dapat memeriksa apakah koleksi dikompresi dan menghitung rasio kompresi sebagai berikut.

Lihat statistik kompresi dengan menjalankan `db.collection.stats()` perintah `db.printCollectionStats()` or dari shell mongo. Output menunjukkan ukuran asli dan ukuran terkompresi yang dapat Anda bandingkan untuk menganalisis keuntungan penyimpanan dari kompresi dokumen. Dalam contoh ini, statistik untuk koleksi bernama “sample_collection” ditunjukkan di bawah ini. Faktor penskalaan 1024* 1024 digunakan di bawah ini untuk menampilkan nilai dan dalam `size` MB. `storageSize`

```
db.sample_collection.stats(1024*1024)
```

Berikut ini adalah contoh output untuk perintah di atas:

```
{
    "ns" : "test.sample_collection",
    "count" : 1000000,
    "size" : 3906.3,
```

```
    "avgObjSize" : 4096,
    "storageSize" : 1953.1,
    compression:{"enabled" : true,"threshold" : 2032},
    ...
}
```

- ukuran - Ukuran asli dari koleksi dokumen.
- rata-rata ObjSize - Ukuran dokumen rata-rata sebelum kompresi dibulatkan ke desimal pertama. Satuan ukuran adalah byte.
- StorageSize - Ukuran penyimpanan koleksi setelah kompresi. Satuan ukuran adalah byte.
- diaktifkan - Menunjukkan jika kompresi diaktifkan atau dinonaktifkan.

Untuk menghitung rasio kompresi aktual, bagi ukuran koleksi dengan ukuran penyimpanan (size/storageSize). Untuk contoh di atas, perhitungannya adalah 3906. 3/1953.1 yang diterjemahkan menjadi rasio kompresi 2:1.

Mengelola kompresi berbasis kamus di Amazon DocumentDB 8.0

Amazon DocumentDB 8.0 memperkenalkan algoritma kompresi dokumen baru, zstd, sebagai alternatif yang ditingkatkan untuk LZ4. Saat Anda mengaktifkan kompresi kamus pada koleksi Amazon DocumentDB 8.0 dengan memilih Zstd sebagai algoritma kompresi, dokumen dari koleksi Anda diambil sampelnya untuk membuat kamus kompresi khusus. Dokumen baru dan yang diperbarui dikompresi menggunakan kamus ini dan algoritma zstd. Pendekatan ini biasanya mencapai rasio kompresi yang lebih tinggi daripada metode kompresi standar, terutama untuk koleksi dengan skema dokumen yang konsisten atau nama bidang berulang.

Lz4 adalah algoritma yang dirancang untuk kompresi dan dekompresi cepat. Hal ini dikenal ringan pada CPU sambil mencapai kompresi yang nyata. Zstd adalah algoritma tujuan umum yang dengan pengaturan default menggunakan lebih banyak CPU tetapi mencapai rasio kompresi yang lebih baik daripada lz4. Penggunaan kamus meningkatkan kompresi lebih jauh untuk sebagian besar dokumen JSON. Beberapa manfaat dari algoritma Zstd adalah:

- Mengurangi Biaya Penyimpanan: Rasio kompresi yang lebih tinggi berarti penggunaan penyimpanan yang lebih sedikit dan biaya yang lebih rendah.
- Lebih rendah I/O: Dokumen terkompresi membutuhkan lebih sedikit I/O, berpotensi meningkatkan kinerja.
- Dioptimalkan untuk koleksi Anda: Kamus dilatih khusus untuk pola data koleksi Anda.

> **Note**
>
> Dictionary-based kompresi tidak didukung di Amazon DocumentDB versi 3.6, 4.0, dan 5.0.

Pertimbangan Kinerja

Kompresi Zstd melibatkan trade-off berikut:

- Penyimpanan vs CPU: Kompresi Zstd mencapai rasio kompresi yang lebih baik tetapi mungkin menggunakan sumber daya CPU yang sedikit lebih banyak dibandingkan dengan kompresi LZ4.
- Kompresi Awal: Koleksi baru mungkin tidak mencapai kompresi optimal sampai cukup dokumen dimasukkan untuk melatih kamus yang efektif. Saat ini, kamus dilatih jika koleksi setidaknya 100 dokumen.
- Jenis Beban Read-intensive Kerja: beban kerja di mana seluruh data masuk ke dalam cache buffer mungkin mengalami peningkatan latensi dan penggunaan CPU karena overhead dekompresi.

Kompresi Zstd sangat efektif untuk koleksi dengan dokumen kecil, array dokumen dan nama bidang berulang.

Mengaktifkan kompresi berbasis kamus

Untuk koleksi baru Anda dapat menggunakan perintah di bawah ini untuk mengaktifkan kompresi Zstd:

```
db.createCollection("myCollection",
    {
        storageEngine: {
            documentDB: {
                compression: {
                    enable: true,
                    algorithm: "zstd"
                }
            }
        }
    }
 )
```

Anda juga dapat mengaktifkan atau memodifikasi kompresi pada koleksi yang ada:

```
db.runCommand({
    collMod: "myCollection",
    storageEngine: {
        documentDB: {
            compression: {
                enable: true,
                algorithm: "zstd"
            }
        }
    }
 })
```

Untuk mengaktifkan algoritma Zstd di semua koleksi di cluster Anda, Anda dapat memodifikasi grup parameter cluster untuk memilih “zstd” sebagai nilai untuk parameter “default_collection_compression”.

Memulai

Amazon DocumentDB 8.0 hadir dengan kompresi Zstd diaktifkan secara default. Anda selalu dapat mematikannya dengan menyetel nilai 'default_compression' menjadi dinonaktifkan di grup parameter cluster Anda. Perlu dicatat bahwa dimulai dengan Amazon DocumentDB 8.0, 'enabled' bukan lagi pilihan yang valid untuk default_compression, dan Anda harus memilih dari Zstd dan LZ4.

Memantau

Anda dapat melihat informasi kompresi untuk koleksi menggunakan salah satu perintah berikut:

- db.runCommand ({collStats: “myCollection”}) ATAU
- db.collection.stats ()

Perintah ini mengembalikan statistik kunci yang dapat Anda gunakan untuk menghitung rasio kompresi:

- compression.algorithm: Algoritma yang digunakan (“lz4" atau “zstd”)
- StorageSize: Penyimpanan aktual yang digunakan oleh koleksi, setelah kompresi. Perhatikan bahwa nomor ini mencakup fragmentasi (yaitu, ruang yang tidak digunakan di halaman database)

- rata-rataObjSize: Ukuran logis rata-rata dokumen koleksi, didekompresi. Perhatikan bahwa jika koleksi Anda memiliki lebih dari 20k dokumen, nomor ini akan menjadi perkiraan berdasarkan sampel 20k dokumen.
- ukuran: Ukuran logis dari koleksi tanpa kompresi. Angka ini diperoleh dengan mengalikan rata-rata ObjSize dengan jumlah total dokumen dalam koleksi, jadi jika rata-rata ObjSize adalah perkiraan, angka ini juga akan menjadi perkiraan.
- hitung: Jumlah dokumen dalam koleksi

CloudWatch Metrik berikut dapat membantu saat mengevaluasi kompresi berbasis kamus:

- CPUUtilization
- FreeableMemory
- VolumeBytesUsed
- VolumeReadIOPs
- VolumeWriteIOPs

Metrik CollStats:

- storageSize
- ukuran

Selain itu, dapat berguna untuk melacak metrik khusus untuk aplikasi Anda, seperti latensi dan throughput per jenis kueri atau API.

Mengelola acara Amazon DocumentDB

Amazon DocumentDB (dengan kompabilitas MongoDB) menyimpan catatan peristiwa yang terkait dengan klaster, instans, snapshot, grup keamanan, dan grup parameter klaster Anda. Informasi ini mencakup tanggal dan waktu acara, nama sumber dan jenis sumber acara, dan pesan yang terkait dengan acara tersebut.

> **Important**
>
> Untuk fitur pengelolaan tertentu, Amazon DocumentDB menggunakan teknologi operasional yang dibagi dengan Amazon RDS dan Amazon Neptune. Batas wilayah, batas yang diatur di

tingkat Wilayah, dibagi antara Amazon DocumentDB, Amazon RDS, dan Amazon Neptune. Untuk informasi selengkapnya, lihat Kuota regional.

Topik

- Melihat kategori acara Amazon DocumentDB
- Melihat acara Amazon DocumentDB

Melihat kategori acara Amazon DocumentDB

Setiap jenis sumber daya Amazon DocumentDB memiliki jenis kejadian tertentu yang dapat dikaitkan dengannya. Anda dapat menggunakan AWS CLI `describe-event-categories` operasi untuk melihat pemetaan antara jenis peristiwa dan jenis sumber daya Amazon DocumentDB.

Parameter

- **`--source-type`**—Opsional. Gunakan parameter `--source-type` untuk melihat kategori kejadian pada jenis sumber tertentu. Nilai berikut diizinkan:
  - `db-cluster`
  - `db-instance`
  - `db-parameter-group`
  - `db-security-group`
  - `db-cluster-snapshot`
- **`--filters`**—Opsional. Untuk menampilkan kategori peristiwa hanya untuk Amazon DocumentDB, gunakan filter `--filter Name=engine,Values=docdb`.

Example

Kode berikut mencantumkan kategori peristiwa yang terkait dengan klaster.

Untuk Linux, macOS, atau Unix:

```
aws docdb describe-event-categories \
    --filter Name=engine,Values=docdb \
    --source-type db-cluster
```

Untuk Windows:

```
aws docdb describe-event-categories ^
    --filter Name=engine,Values=docdb ^
    --source-type db-cluster
```

Output dari operasi ini terlihat seperti berikut (format JSON).

```
{
    "EventCategoriesMapList": [
        {
            "EventCategories": [
                "notification",
                "failure",
                "maintenance",
                "failover"
            ],
            "SourceType": "db-cluster"
        }
    ]
}
```

Kode berikut mencantumkan kategori peristiwa yang terkait dengan setiap jenis sumber Amazon DocumentDB.

```
aws docdb describe-event-categories
```

Output dari operasi ini terlihat seperti berikut (format 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"
            ]
        }
    ]
}
```

Melihat acara Amazon DocumentDB

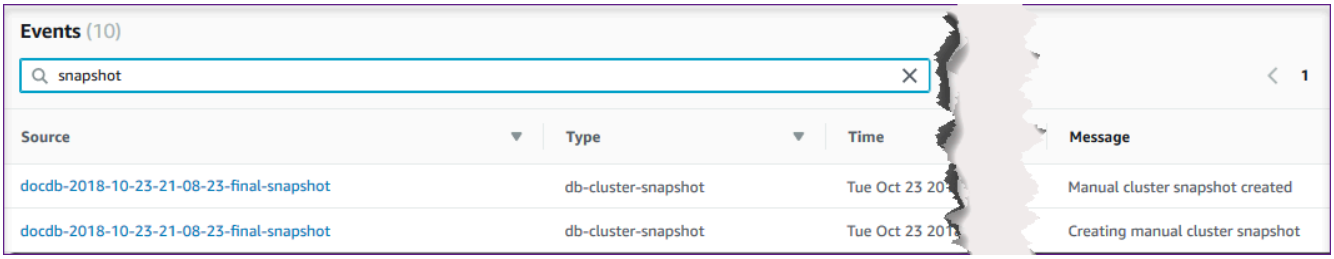
Anda dapat mengambil kejadian untuk sumber daya Amazon DocumentDB Anda melalui konsol Amazon DocumentDB, yang menampilkan kejadian selama 24 jam terakhir. Anda juga dapat

mengambil peristiwa untuk resource Amazon DocumentDB Anda dengan menggunakan AWS CLI perintah describe-events, atau DescribeEventsoperasi Amazon DocumentDB API. Jika Anda menggunakan AWS CLI atau Amazon DocumentDB API untuk melihat peristiwa, Anda dapat mengambil peristiwa hingga 14 hari terakhir.

Using the Konsol Manajemen AWS

Untuk menampilkan semua acara instans Amazon DocumentDB selama 24 jam terakhir

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Peristiwa. Peristiwa yang tersedia akan muncul dalam daftar.
3. Gunakan daftar Filter untuk menyelekasi peristiwa menurut jenisnya. Masukkan suatu istilah dalam kotak teks untuk menyeleksi hasil Anda lebih lanjut. Misalnya, cuplikan layar berikut menunjukkan pemfilteran semua peristiwa Amazon DocumentDB untuk peristiwa snapshot.



Using the AWS CLI

Untuk menampilkan semua peristiwa instans Amazon DocumentDB selama 7 hari terakhir

Anda dapat melihat semua kejadian instans Amazon DocumentDB selama 7 hari terakhir dengan menjalankan operasi AWS CLI deskripsi-peristiwa dengan parameter disetel ke (10.080 `--duration` menit). `10080`

```
aws docdb describe-events --duration 10080
```

Pemfilteran untuk Acara Amazon DocumentDB

Untuk menampilkan peristiwa Amazon DocumentDB tertentu, gunakan operasi `describe-events` dengan parameter berikut.

Parameter

- **`--filter`**—Diperlukan untuk membatasi nilai yang ditampilkan ke peristiwa Amazon DocumentDB. Gunakan **`Name=engine,Values=docdb`** untuk memfilter semua peristiwa untuk Amazon DocumentDB saja.
- **`--source-identifier`**—Opsional. Pengidentifikasi sumber peristiwa yang akan menerima peristiwa tersebut. Jika dihilangkan, peristiwa dari semua sumber disertakan ke dalam hasilnya.
- **`--source-type`**—Opsional, jika `--source-identifier` tidak disediakan, maka ini diperlukan. Jika `--source-identifier` disediakan, `--source-type` harus setuju dengan jenis `--source-identifier`. Nilai berikut diizinkan:
  - `db-cluster`
  - `db-instance`
  - `db-parameter-group`
  - `db-security-group`
  - `db-cluster-snapshot`

Contoh berikut mencantumkan semua persitiwa Amazon DocumentDB Anda.

```
aws docdb describe-events --filters Name=engine,Values=docdb
```

Output dari operasi ini terlihat seperti berikut (format 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",
```

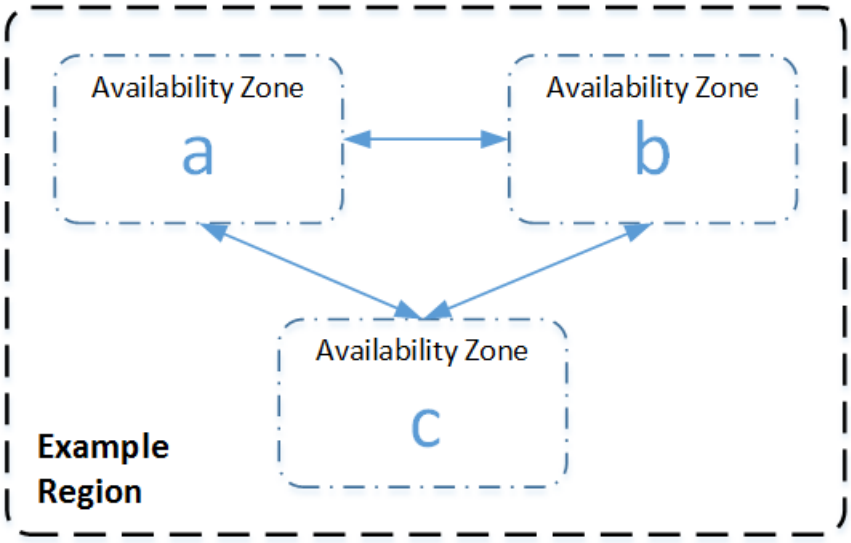
```
            "Message": "instance shutdown",
            "SourceType": "db-instance",
            "Date": "2018-12-11T21:25:01.245Z",
            "SourceIdentifier": "docdb-2018-12-11-21-08-23",
            "EventCategories": [
                "availability"
            ]
        },
        {
            "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"
            ]
        }
    ]
}
```

Lihat informasi yang lebih lengkap di Mengaudit acara Amazon DocumentDB.

Memilih wilayah dan zona ketersediaan

Sumber daya komputasi cloud Amazon di-host di beberapa lokasi di seluruh dunia. Lokasi ini terdiri dari Wilayah AWS dan Availability Zones. Setiap Wilayah AWS adalah wilayah geografis yang terpisah. Setiap Wilayah memiliki beberapa lokasi terisolasi yang dikenal sebagai Availability Zone. Amazon DocumentDB memberi Anda kemampuan untuk menempatkan sumber daya, seperti instans, dan data di beberapa lokasi. Sumber daya tidak direplikasi Wilayah AWS kecuali Anda melakukannya secara khusus.

Amazon mengoperasikan pusat data yang canggih dan sangat tersedia. Meskipun jarang terjadi, kegagalan dapat terjadi yang memengaruhi ketersediaan instans yang berada di lokasi yang sama. Jika Anda menghosting semua instans Anda di satu lokasi yang terpengaruh oleh kegagalan tersebut, tidak ada instans Anda yang akan tersedia. Diagram berikut menunjukkan Wilayah AWS dengan tiga Availability Zones.



Penting untuk diingat bahwa setiap Wilayah adalah independen. Aktivitas Amazon DocumentDB apa pun yang Anda mulai (misalnya, membuat instans atau membuat daftar instans yang tersedia) hanya berjalan di Wilayah AWS default Anda saat ini. Anda dapat mengubah Wilayah default di konsol tersebut dengan mengatur variabel lingkungan `EC2_REGION`. Atau Anda dapat menimpanya dengan menggunakan parameter `--region` di AWS CLI. Untuk informasi selengkapnya, lihat Mengkonfigurasi AWS Command Line Interface, secara khusus, bagian tentang variabel lingkungan dan opsi baris perintah.

Saat Anda membuat klaster menggunakan konsol Amazon DocumentDB, dan Anda memilih untuk membuat replika di Availability Zone yang berbeda, Amazon DocumentDB membuat dua instans. Ini membuat instans primer di satu Availability Zone dan instans replika di Availability Zone yang berbeda. Volume klaster selalu direplikasi di tiga Availability Zone.

Untuk membuat atau bekerja dengan instans Amazon DocumentDB secara Wilayah AWS spesifik, gunakan titik akhir layanan regional yang sesuai.

Ketersediaan wilayah

Amazon DocumentDB tersedia di Wilayah berikut. AWS

Wilayah yang didukung oleh Amazon DocumentDB

Nama Wilayah	Region	Availability Zone (penghitungan)
AS Timur (Ohio)	`us-east-2`	3

Nama Wilayah	Region	Availability Zone (penghitungan)
AS Timur (Virginia Utara)	us-east-1	6
AS Barat (Oregon)	us-west-2	4
Africa (Cape Town)	af-south-1	3
Amerika Selatan (Sao Paulo)	sa-east-1	3
Asia Pasifik (Hong Kong)	ap-east-1	3
Asia Pasifik (Hyderabad)	ap-south-2	3
Asia Pasifik (Malaysia)	ap-southeast-5	3
Asia Pasifik (Mumbai)	ap-south-1	3
Asia Pasifik (Osaka)	ap-northeast-3	3
Asia Pasifik (Seoul)	ap-northeast-2	4
Asia Pasifik (Singapura)	ap-southeast-1	3
Asia Pasifik (Sydney)	ap-southeast-2	3
Asia Pasifik (Jakarta)	ap-southeast-3	3
Asia Pacific (Melbourne)	ap-southeast-4	3
Asia Pasifik (Thailand)	ap-southeast-7	3
Asia Pasifik (Tokyo)	ap-northeast-1	3

Nama Wilayah	Region	Availability Zone (penghitungan)
Kanada (Pusat)	`ca-central-1`	3
Kanada Barat (Calgary)	`ca-west-1`	3
Wilayah Tiongkok (Beijing)	`cn-north-1`	3
Tiongkok (Ningxia)	`cn-northwest-1`	3
Eropa (Frankfurt)	`eu-central-1`	3
Europe (Zurich)	`eu-central-2`	3
Eropa (Irlandia)	`eu-west-1`	3
Eropa (London)	`eu-west-2`	3
Europe (Milan)	`eu-south-1`	3
Eropa (Paris)	`eu-west-3`	3
Eropa (Spanyol)	`eu-south-2`	3
Eropa (Stockholm)	`eu-north-1`	3
Meksiko (Tengah)	`mx-central-1`	3
Timur Tengah (UAE)	`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

Secara default, zona waktu untuk klaster Amazon DocumentDB adalah Universal Time Coordinated (UTC).

Untuk informasi tentang menemukan titik akhir koneksi untuk klaster dan instans di wilayah tertentu, lihat Memahami titik akhir Amazon DocumentDB.

Mengelola grup parameter cluster Amazon DocumentDB

Anda dapat mengelola konfigurasi mesin Amazon DocumentDB dengan menggunakan parameter dalam grup parameter klaster. Grup parameter klaster adalah kumpulan nilai konfigurasi Amazon DocumentDB yang mempermudah pengelolaan parameter klaster Amazon DocumentDB Anda. Grup parameter klaster bertindak sebagai kontainer untuk nilai konfigurasi mesin yang diterapkan ke semua instans dalam klaster.

Bagian ini menjelaskan cara membuat, melihat, dan mengubah grup parameter klaster. Ini juga menunjukkan bagaimana Anda dapat menentukan grup parameter klaster mana yang dikaitkan dengan klaster tertentu.

Topik

- Menjelaskan grup parameter cluster Amazon DocumentDB
- Membuat grup parameter cluster Amazon DocumentDB
- Memodifikasi grup parameter cluster Amazon DocumentDB
- Memodifikasi cluster Amazon DocumentDB untuk menggunakan grup parameter cluster kustom
- Menyalin grup parameter cluster Amazon DocumentDB
- Menyetel ulang grup parameter cluster Amazon DocumentDB
- Menghapus grup parameter cluster Amazon DocumentDB
- Referensi parameter cluster Amazon DocumentDB

Menjelaskan grup parameter cluster Amazon DocumentDB

Grup parameter `default` cluster dibuat secara otomatis saat Anda membuat cluster Amazon DocumentDB pertama di yang Wilayah AWS baru atau menggunakan mesin baru. Cluster berikutnya, yang dibuat di Wilayah yang sama dan memiliki versi mesin yang sama, dibuat dengan grup parameter `default` cluster.

Topik

- Menjelaskan detail grup parameter cluster Amazon DocumentDB
- Menentukan grup parameter cluster Amazon DocumentDB
- Menentukan cluster dan instance yang terkait dengan grup parameter cluster Amazon DocumentDB

Menjelaskan detail grup parameter cluster Amazon DocumentDB

Untuk menjelaskan rincian kelompok parameter cluster yang diberikan, selesaikan langkah-langkah berikut menggunakan Konsol Manajemen AWS atau AWS Command Line Interface (AWS CLI).

Using the Konsol Manajemen AWS

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Grup parameter.

   > **Tip**
   >
   > Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu (≡) di pojok kiri atas halaman.

3. Di panel Grup parameter klaster , pilih nama grup parameter yang ingin Anda lihat detailnya.
4. Halaman yang dihasilkan menunjukkan parameter grup parameter, aktivitas terbaru, dan tag.

   - Di bawah Parameter klaster, Anda dapat melihat nama parameter, nilai saat ini, nilai yang diizinkan, apakah parameter dapat dimodifikasi, jenis penerapannya, tipe data, dan deskripsinya. Anda dapat mengubah parameter individual dengan memilih parameter lalu memilih Edit di bagian Parameter klaster. Untuk informasi selengkapnya, lihat Memodifikasi parameter cluster Amazon DocumentDB.
   - Di bawah Peristiwa terbaru, Anda dapat melihat peristiwa terbaru untuk grup parameter ini. Anda dapat memfilter acara ini menggunakan bar pencarian di bagian ini. Untuk informasi selengkapnya, lihat Mengelola acara Amazon DocumentDB.
   - Di bawah Tag, Anda dapat melihat tag yang ada di grup parameter klaster ini. Anda dapat menambah atau menghapus tag dengan memilih Edit di bagian Tag. Untuk informasi selengkapnya, lihat Menandai sumber daya Amazon DocumentDB.

Using the AWS CLI

Anda dapat menggunakan `describe-db-cluster-parameter-groups` AWS CLI perintah untuk melihat Amazon Resource Name (ARN), keluarga, deskripsi, dan nama grup parameter cluster tunggal atau semua grup parameter cluster yang Anda miliki untuk Amazon DocumentDB. Anda juga dapat menggunakan `describe-db-cluster-parameters` AWS CLI perintah untuk melihat parameter dan detailnya di dalam grup parameter cluster tunggal.

- **`--describe-db-cluster-parameter-groups`** — Untuk melihat daftar semua grup parameter klaster dan detailnya.
  - **`--db-cluster-parameter-group-name`** – Opsional. Nama grup parameter klaster yang ingin Anda jelaskan. Jika parameter ini dihilangkan, semua grup parameter klaster akan dijelaskan.
- **`--describe-db-cluster-parameters`** — Untuk membuat daftar semua parameter di dalam grup parameter dan nilainya.
  - **`--db-cluster-parameter-group name`** — Diperlukan. Nama grup parameter klaster yang ingin Anda jelaskan.

Example

Kode berikut mencantumkan hingga 100 grup parameter klaster dan ARN, keluarga, deskripsi, dan namanya.

```
aws docdb describe-db-cluster-parameter-groups
```

Output dari operasi ini terlihat seperti berikut (format 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

Kode berikut mencantumkan ARN, keluarga, deskripsi, dan nama untuk `sample-parameter-group`.

Untuk Linux, macOS, atau Unix:

```
aws docdb describe-db-cluster-parameter-groups \
        --db-cluster-parameter-group-name sample-parameter-group
```

Untuk Windows:

```
aws docdb describe-db-cluster-parameter-groups ^
        --db-cluster-parameter-group-name sample-parameter-group
```

Output dari operasi ini terlihat seperti berikut (format 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

Kode berikut mencantumkan nilai parameter dalam `sample-parameter-group`.

Untuk Linux, macOS, atau Unix:

```
aws docdb describe-db-cluster-parameters \
    --db-cluster-parameter-group-name sample-parameter-group
```

Untuk Windows:

```
aws docdb describe-db-cluster-parameters ^
    --db-cluster-parameter-group-name sample-parameter-group
```

Output dari operasi ini terlihat seperti berikut (format JSON).

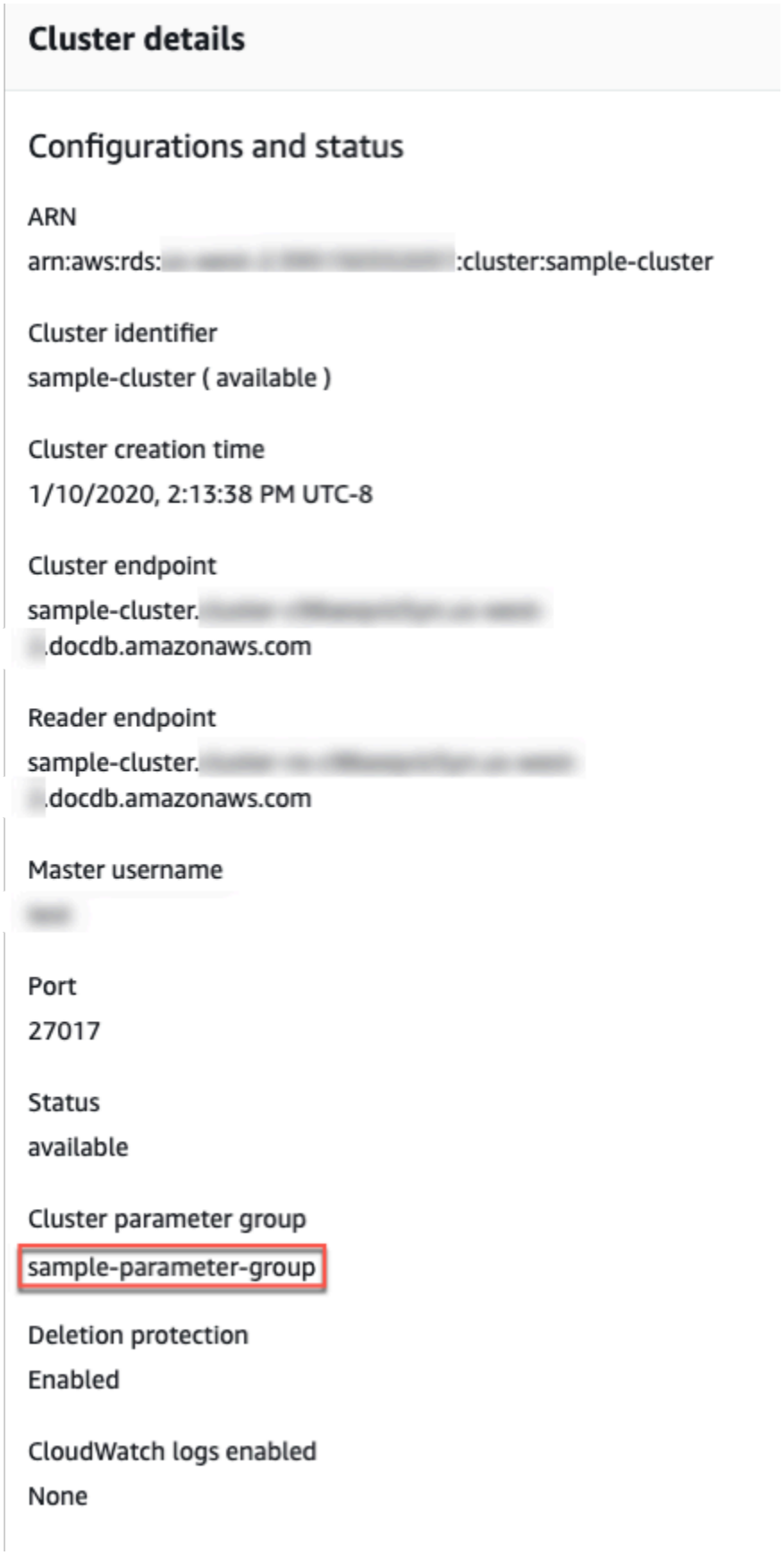
```
{
   "Parameters": [
         {
            "ParameterName": "audit_logs",
            "ParameterValue": "disabled",
            "Description": "Enables auditing on cluster.",
            "Source": "system",
            "ApplyType": "dynamic",
            "DataType": "string",
            "AllowedValues": "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"
         }
   ]
}
```

Menentukan grup parameter cluster Amazon DocumentDB

Untuk menentukan kelompok parameter mana yang terkait dengan cluster tertentu, selesaikan langkah-langkah berikut menggunakan Konsol Manajemen AWS atau AWS CLI.

Using the Konsol Manajemen AWS

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi kiri, pilih Klaster.
3. Dalam daftar klaster, pilih nama klaster yang Anda minati.
4. Halaman yang dihasilkan menunjukkan detail klaster yang Anda pilih. Gulir ke bawah ke Detail klaster. Di bagian bawah bagian tersebut, temukan nama grup parameter di bawah Grup parameter klaster.


Cluster details
Configurations and status
ARN
arn:aws:rds: :cluster:sample-cluster
Cluster identifier
sample-cluster (available)
Cluster creation time
1/10/2020, 2:13:38 PM UTC-8
Cluster endpoint
sample-cluster.
.docdb.amazonaws.com
Reader endpoint
sample-cluster.
.docdb.amazonaws.com
Master username
Port
27017
Status
available
Cluster parameter group
sample-parameter-group
Deletion protection
Enabled
CloudWatch logs enabled
None

Using the AWS CLI

AWS CLI Kode berikut menentukan kelompok parameter mana yang mengatur cluster`sample-cluster`.

```
aws docdb describe-db-clusters \
      --db-cluster-identifier sample-cluster \
      --query 'DBClusters[*].[DBClusterIdentifier,DBClusterParameterGroup]'
```

Keluaran dari operasi ini terlihat seperti berikut ini (format JSON).

```
[
       [
           "sample-cluster",
           "sample-parameter-group"
       ]
]
```

Menentukan cluster dan instance yang terkait dengan grup parameter cluster Amazon DocumentDB

Untuk menentukan berapa banyak cluster dan instance yang terkait dengan setiap grup parameter, selesaikan langkah-langkah berikut menggunakan. AWS CLI

Using the AWS CLI

AWS CLI Kode berikut menentukan berapa banyak cluster dan instance yang terkait dengannya. `sample-parameter-group`

Untuk Linux, macOS, atau 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
      }) | .[]'
```

Untuk 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
      }) | .[]'
```

Keluaran dari operasi ini terlihat seperti berikut ini (format JSON).

```
{
   "parameter_group_name": "sample-parameter-group",
   "total_clusters": 50,
   "total_instances": 150
}
```

Membuat grup parameter cluster Amazon DocumentDB

Grup parameter cluster default sepertidefault.docdb5.0,default.docdb4.0, ataudefault.docdb3.6, dibuat saat Anda membuat klaster dengan versi mesin baru dan yang baru Wilayah AWS. Cluster berikutnya dibuat di Wilayah yang sama dan dengan versi mesin yang sama mewarisi grup parameter default cluster. Setelah dibuat, grup default parameter tidak dapat dihapus atau diganti namanya. Anda dapat memodifikasi perilaku engine instance cluster dengan membuat grup parameter kustom dengan nilai parameter pilihan dan melampirkannya ke cluster Amazon DocumentDB Anda.

Prosedur berikut memandu Anda melalui pembuatan grup parameter cluster kustom. Anda kemudian dapat mengubah parameter dalam grup parameter tersebut.

> **Note**
>
> Setelah Anda membuat grup parameter klaster, Anda harus menunggu setidaknya 5 menit sebelum menggunakan grup parameter tersebut. Ini memungkinkan Amazon DocumentDB untuk menyelesaikan tindakan create sepenuhnya sebelum grup parameter klaster

digunakan untuk klaster baru. Anda dapat menggunakan Konsol Manajemen AWS atau `describe-db-cluster-parameter-groups` AWS CLI operasi untuk memverifikasi bahwa grup parameter cluster Anda telah dibuat. Untuk informasi selengkapnya, lihat Menjelaskan grup parameter cluster Amazon DocumentDB.

Using the Konsol Manajemen AWS

Untuk membuat grup parameter klaster

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Grup parameter.

   > **Tip**
   >
   > Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu (≡) di pojok kiri atas halaman.

3. Di panel Grup parameter klaster, pilih Buat.
4. Di panel Buat grup parameter klaster, masukkan berikut ini:

   a. Nama grup parameter cluster baru - Masukkan nama untuk grup parameter cluster. Misalnya, `sample-parameter-group`. Grup parameter klaster memiliki batasan penamaan berikut:

      - Panjangnya [1–255] karakter alfanumerik.
      - Karakter pertama harus berupa huruf.
      - Tidak dapat diakhiri dengan tanda hubung atau berisi dua tanda hubung berurutan.

   b. Keluarga - Pilih versi DocumentDB yang ingin Anda gunakan untuk cluster Anda.

   c. Deskripsi — Memberikan deskripsi untuk grup parameter klaster ini.

5. Untuk membuat grup parameter klaster, pilih Buat. Untuk membatalkan operasi, pilih Batal.
6. Setelah Anda memilih Buat, teks berikut akan muncul di bagian atas halaman untuk memverifikasi bahwa grup parameter klaster Anda telah berhasil dibuat:

```
Successfully created cluster parameter group 'sample-parameter-group'.
```

Using the AWS CLI

Untuk membuat grup parameter cluster baru untuk cluster Amazon DocumentDB 4.0, gunakan operasi AWS CLI `create-db-cluster-parameter-group` dengan parameter berikut:

- **`--db-cluster-parameter-group-name`** — Nama grup parameter klaster kustom. Misalnya, `sample-parameter-group`.
- **`--db-cluster-parameter-group-family`** — Keluarga grup parameter klaster yang digunakan sebagai templat untuk grup parameter klaster kustom.
- **`--description`** — Deskripsi yang diberikan pengguna untuk grup parameter klaster ini. Contoh berikut menggunakan "`Custom docdb4.0 parameter group`".

Untuk Linux, macOS, atau 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"
```

Untuk 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"
```

Output dari operasi ini terlihat seperti berikut (format JSON).

```
{
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupName": "sample-parameter-group",
        "DBParameterGroupFamily": "docdb5.0",
        "Description": "Custom docdb4.0 parameter group",
```

```
        "DBClusterParameterGroupArn": "sample-parameter-group-arn"
    }
}
```

Memodifikasi grup parameter cluster Amazon DocumentDB

Bagian ini menjelaskan cara memodifikasi grup parameter Amazon DocumentDB kustom. Di Amazon DocumentDB, Anda tidak dapat memodifikasi `default` grup parameter cluster yang dibuat saat pertama kali membuat klaster dengan versi engine baru di Wilayah baru. Jika klaster Amazon DocumentDB Anda menggunakan grup parameter klaster default dan Anda ingin mengubah nilai di dalamnya, Anda harus terlebih dahulu membuat grup parameter baru atau menyalin grup parameter yang ada, memodifikasinya, lalu menerapkan grup parameter yang dimodifikasi ke klaster Anda.

Selesaikan langkah-langkah berikut untuk memodifikasi grup parameter cluster kustom. Memodifikasi tindakan bisa memakan waktu beberapa saat untuk disebarkan. Tunggu grup parameter cluster yang dimodifikasi tersedia sebelum melampirkannya ke cluster Anda. Anda dapat menggunakan Konsol Manajemen AWS atau AWS CLI `describe-db-cluster-parameters` operasi untuk memverifikasi bahwa grup parameter cluster Anda telah dimodifikasi. Untuk informasi selengkapnya, lihat Menjelaskan kelompok parameter cluster.

Using the Konsol Manajemen AWS

Ikuti langkah-langkah ini untuk mengubah grup parameter Amazon DocumentDB kustom. Anda tidak dapat mengubah grup parameter `default`. Jika Anda ingin mengubah nilai dalam grup parameter `default`, Anda dapat menyalin grup parameter klaster default, memodifikasinya, lalu menerapkan grup parameter yang dimodifikasi ke klaster Anda. Untuk informasi selengkapnya tentang menerapkan grup parameter ke klaster Anda, lihat Memodifikasi cluster Amazon DocumentDB.

Untuk mengubah grup parameter klaster kustom

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi di sisi kiri konsol tersebut, pilih Grup parameter. Dalam daftar grup parameter, pilih nama grup parameter yang ingin Anda ubah.

> **Tip**
>
> Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu (≡) di pojok kiri atas halaman.

3. Untuk setiap parameter dalam grup parameter yang ingin Anda ubah, lakukan hal berikut:

   a. Temukan parameter yang ingin Anda ubah, dan verifikasi bahwa parameter tersebut dapat dimodifikasi dengan memeriksa apakah parameter tersebut terdaftar sebagai `true` di bawah kolom yang Dapat dimodifikasi.

   b. Jika dapat dimodifikasi, pilih parameter dan pilih Edit dari kanan atas halaman konsol tersebut.

   c. Dalam kotak dialog Modifikasi **<parameter-name>**, buat perubahan yang Anda inginkan. Kemudian pilih Ubah parameter klaster, atau pilih Batal untuk membuang perubahan.

Using the AWS CLI

Anda dapat memodifikasi `ParameterValue`, `Description`, atau `ApplyMethod` dari parameter apa pun yang dapat dimodifikasi di dalam grup parameter klaster Amazon DocumentDB kustom menggunakan AWS CLI. Anda tidak dapat membuat modifikasi langsung ke grup parameter klaster default.

Untuk mengubah parameter grup parameter klaster kustom, gunakan operasi `modify-db-cluster-parameter-group` dengan parameter berikut.

- **`--db-cluster-parameter-group-name`** — Diperlukan. Nama grup parameter klaster yang Anda modifikasi.
- **`--parameters`** — Diperlukan. Parameter yang Anda modifikasi. Untuk daftar parameter yang berlaku untuk semua instans di klaster Amazon DocumentDB, lihat Referensi parameter cluster Amazon DocumentDB. Setiap entri parameter harus menyertakan hal berikut:
  - **`ParameterName`** — Nama parameter yang Anda ubah.
  - **`ParameterValue`** — Nilai baru untuk parameter ini.
  - **`ApplyMethod`** — Bagaimana Anda ingin perubahan pada parameter ini diterapkan. Nilai yang diizinkan adalah `immediate` dan `pending-reboot`.

> **Note**
>
> Parameter dengan `ApplyType` dari `static` harus memiliki `ApplyMethod` dari `pending-reboot`.

Example- Memodifikasi nilai parameter

Dalam contoh ini, Anda mencantumkan nilai parameter `sample-parameter-group` dan memodifikasi parameter `tls`. Kemudian, setelah menunggu 5 menit, Anda kembali mencantumkan nilai parameter `sample-parameter-group` untuk melihat nilai parameter yang diubah.

1. Buat daftar parameter dan nilai `sample-parameter-group` mereka.

   Untuk Linux, macOS, atau Unix:

   ```
   aws docdb describe-db-cluster-parameters \
          --db-cluster-parameter-group-name sample-parameter-group
   ```

   Untuk Windows:

   ```
   aws docdb describe-db-cluster-parameters ^
          --db-cluster-parameter-group-name sample-parameter-group
   ```

   Output dari operasi ini terlihat seperti berikut (format 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. Ubah parameter `tls` sehingga nilainya menjadi `disabled` .

   Anda tidak dapat mengubah `ApplyMethod` karena `ApplyType` adalah `static`.

   Untuk Linux, macOS, atau Unix:

```
aws docdb modify-db-cluster-parameter-group \
       --db-cluster-parameter-group-name sample-parameter-group \
       --parameters
 "ParameterName"=tls,"ParameterValue"=disabled,"ApplyMethod"=pending-reboot
```

   Untuk Windows:

```
aws docdb modify-db-cluster-parameter-group ^
       --db-cluster-parameter-group-name sample-parameter-group ^
       --parameters
 "ParameterName"=tls,"ParameterValue"=disabled,"ApplyMethod"=pending-reboot
```

   Output dari operasi ini terlihat seperti berikut (format JSON).

```
{
       "DBClusterParameterGroupName": "sample-parameter-group"
   }
```

3. Tunggu paling tidak 5 menit.

4. Buat daftar nilai parameter `sample-parameter-group` untuk memverifikasi bahwa parameter `tls` telah dimodifikasi.

   Untuk Linux, macOS, atau Unix:

```
aws docdb describe-db-cluster-parameters \
       --db-cluster-parameter-group-name sample-parameter-group
```

   Untuk Windows:

```
aws docdb describe-db-cluster-parameters ^
       --db-cluster-parameter-group-name sample-parameter-group
```

   Keluaran dari operasi ini terlihat seperti berikut ini (format 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"
           }
       ]
}
```

Memodifikasi cluster Amazon DocumentDB untuk menggunakan grup parameter cluster kustom

Saat Anda membuat klaster Amazon DocumentDB, grup parameter `default.docdb4.0` secara otomatis dibuat untuk klaster tersebut. Anda tidak dapat mengubah grup parameter klaster `default`. Sebagai gantinya, Anda dapat memodifikasi klaster Amazon DocumentDB untuk mengaitkan grup parameter baru yang disesuaikan dengan itu.

Bagian ini menjelaskan cara memodifikasi klaster Amazon DocumentDB yang ada untuk menggunakan grup parameter cluster kustom menggunakan Konsol Manajemen AWS dan AWS Command Line Interface ().AWS CLI

Using the Konsol Manajemen AWS

Untuk memodifikasi klaster Amazon DocumentDB untuk menggunakan grup parameter klaster bukan-default yang baru

1. Sebelum Anda mulai, pastikan Anda telah membuat klaster Amazon DocumentDB dan grup parameter klaster. Lihat Membuat cluster Amazon DocumentDB dan Membuat grup parameter cluster Amazon DocumentDB untuk instruksi lebih lanjut.
2. Setelah membuat grup parameter klaster Anda, buka konsol Amazon DocumentDB di https://console.aws.amazon.com/docdb. Di panel navigasi, pilih Klaster untuk menambahkan grup parameter baru Anda ke sebuah klaster.
3. Pilih klaster yang ingin Anda kaitkan dengan grup parameter Anda. Pilih Tindakan, lalu pilih Ubah untuk mengubah klaster Anda.
4. Di bawah Opsi klaster, pilih grup parameter baru yang ingin Anda kaitkan dengan klaster Anda.
5. Pilih Lanjutkan untuk melihat ringkasan dari modifikasi Anda.
6. Setelah memverifikasi perubahan Anda, Anda dapat menerapkannya segera atau selama jendela pemeliharaan berikutnya di bawah Penjadwalan modifikasi.
7. Pilih Ubah klaster untuk memperbarui klaster Anda dengan grup parameter baru Anda.

Using the AWS CLI

Sebelum Anda memulai, pastikan Anda telah membuat klaster Amazon DocumentDB dan grup parameter klaster. Anda dapat membuat cluster Amazon DocumentDB menggunakan operasi.

AWS CLI `create-db-cluster` Anda dapat membuat grup parameter cluster menggunakan AWS CLI `create-db-cluster-parameter-group` operasi.

Untuk menambahkan grup parameter cluster baru Anda ke cluster Anda, gunakan AWS CLI `modify-db-cluster` operasi dengan parameter berikut.

- --db-cluster-identifier — Nama klaster Anda (misalnya, `sample-cluster`).
- --db-cluster-parameter-group-name — Nama grup parameter yang ingin Anda kaitkan dengan klaster Anda (misalnya, `sample-parameter-group`).

Example

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --db-cluster-parameter-group-name sample-parameter-group
```

Output dari operasi ini terlihat seperti berikut (format JSON).

```
"DBCluster": {
    "AvailabilityZones": [
            "us-west-2c",
            "us-west-2b",
            "us-west-2a"
    ],
    "BackupRetentionPeriod": 1,
    "DBClusterIdentifier": "sample-cluster",
    "DBClusterParameterGroup": "sample-parameter-group",
    "DBSubnetGroup": "default",
   ...
}
```

Menyalin grup parameter cluster Amazon DocumentDB

Anda dapat membuat salinan grup parameter cluster di Amazon DocumentDB menggunakan Konsol Manajemen AWS atau AWS Command Line Interface ().AWS CLI

Using the Konsol Manajemen AWS

Prosedur berikut memandu Anda membuat grup parameter klaster baru dengan membuat salinan grup parameter klaster yang ada.

Untuk menyalin grup parameter klaster

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Grup parameter.
3. Di panel Grup parameter klaster, pilih nama grup parameter klaster yang ingin Anda salin.
4. Pilih Tindakan, lalu pilih Salin untuk menyalin grup parameter tersebut.
5. Di bawah Opsi salin, masukkan nama dan deskripsi untuk grup parameter klaster baru. Kemudian pilih Salin untuk menyimpan perubahan Anda.

Using the AWS CLI

Untuk membuat salinan grup parameter klaster, gunakan operasi `copy-db-cluster-parameter-group` dengan parameter berikut.

- **`--source-db-cluster-parameter-group-identifier`** — Diperlukan. Nama atau Amazon Resource Name (ARN) dari grup parameter klaster yang ingin Anda buat salinannya.

  Jika kelompok parameter cluster sumber dan target sama Wilayah AWS, pengenal dapat berupa nama atau ARN.

  Jika kelompok parameter cluster sumber dan target berbeda Wilayah AWS, pengenal harus ARN.
- **`--target-db-cluster-parameter-group-identifier`** — Diperlukan. Nama atau ARN salinan grup parameter klaster.

  Batas:

  - Tidak dapat berupa null, kosong, atau blank.
  - Harus berisi 1-255 huruf, angka, atau tanda hubung.
  - Karakter pertama harus berupa huruf.
  - Tidak dapat diakhiri dengan tanda hubung atau berisi dua tanda hubung berurutan.
- **`--target-db-cluster-parameter-group-description`** — Diperlukan. Deskripsi yang diberikan pengguna untuk salinan grup parameter klaster.

Example

Kode berikut membuat salinan `sample-parameter-group`, menamai salinan `sample-parameter-group-copy`.

Untuk Linux, macOS, atau 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"
```

Untuk 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"
```

Output dari operasi ini terlihat seperti berikut (format 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"
    }
}
```

Menyetel ulang grup parameter cluster Amazon DocumentDB

Anda dapat mengatur ulang beberapa atau semua nilai parameter grup parameter klaster Amazon DocumentDB ke nilai defaultnya dengan menggunakan Konsol Manajemen AWS atau AWS Command Line Interface AWS CLI() untuk mengatur ulang grup parameter cluster.

Using the Konsol Manajemen AWS

Ikuti langkah-langkah ini untuk mengatur ulang beberapa atau semua nilai parameter grup parameter klaster ke nilai defaultnya.

Untuk mengatur ulang nilai parameter grup parameter klaster

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi di sisi kiri konsol tersebut, pilih Grup parameter.
3. Di panel Grup parameter klaster, pilih nama grup parameter klaster yang ingin Anda atur ulang.
4. Pilih Tindakan, lalu pilih Atur Ulang untuk mengatur ulang grup parameter tersebut.
5. Pada halaman Konfirmasi reset grup parameter klaster yang dihasilkan, konfirmasi bahwa Anda ingin mengatur ulang semua parameter klaster untuk grup parameter tersebut ke defaultnya. Kemudian pilih Atur ulang untuk mengatur ulang grup parameter Anda. Anda juga dapat memilih Batal untuk membuang perubahan Anda.

Using the AWS CLI

Untuk mereset beberapa atau semua nilai parameter grup parameter klaster ke nilai defaultnya, gunakan operasi `reset-db-cluster-parameter-group` dengan parameter berikut.

- **`--db-cluster-parameter-group-name`** — Diperlukan. Nama grup parameter klaster yang akan diatur ulang.
- **`--parameters`** – Opsional. Daftar `ParameterName` dan `ApplyMethod` dalam grup parameter klaster untuk diatur ulang ke nilai defaultnya. Parameter statis harus disetel ke `pending-reboot` agar berlaku pada mulai ulang instans atau permintaan `reboot-db-instance` berikutnya. Anda harus memanggil `reboot-db-instance` untuk setiap instans di klaster yang Anda inginkan untuk menerapkan parameter statis yang diperbarui.

  Parameter ini dan `--reset-all-parameters` saling eksklusif: Anda dapat menggunakan salah satunya tetapi tidak keduanya.
- **`--reset-all-parameters`** atau **`--no-reset-all-parameters`** — Opsional. Menentukan apakah akan mengatur ulang semua parameter (`--reset-all-parameters` atau hanya beberapa parameter (`--no-reset-all-parameters`) ke nilai defaultnya. Parameter `--reset-all-parameters` dan `--parameters` saling eksklusif: Anda dapat menggunakan salah satunya tetapi tidak keduanya.

  Saat Anda mengatur ulang seluruh grup, parameter dinamis segera diperbarui. Parameter statis diatur ke `pending-reboot` untuk diterapkan pada mulai ulang instans atau permintaan `reboot-db-instance` berikutnya. Anda harus memanggil `reboot-db-instance` untuk

setiap instans di klaster yang Anda inginkan untuk menerapkan parameter statis yang diperbarui.

Example

Contoh 1: Mengatur ulang semua parameter ke nilai defaultnya

Kode berikut mengatur ulang semua parameter di grup parameter klaster `sample-parameter-group` nilai defaultnya.

Untuk Linux, macOS, atau Unix:

```
aws docdb reset-db-cluster-parameter-group \
       --db-cluster-parameter-group-name sample-parameter-group \
       --reset-all-parameters
```

Untuk Windows:

```
aws docdb reset-db-cluster-parameter-group ^
       --db-cluster-parameter-group-name sample-parameter-group ^
       --reset-all-parameters
```

Contoh 2: Mengatur ulang parameter yang ditentukan ke nilai defaultnya

Kode berikut mengatur ulang parameter `tls` di grup parameter klaster `sample-parameter-group` ke nilai defaultnya.

Untuk Linux, macOS, atau 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
```

Untuk 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
```

Output dari operasi ini terlihat seperti berikut (format JSON).

```
{
        "DBClusterParameterGroupName": "sample-parameter-group"
   }
```

Melakukan reboot instans klaster

Sebelum nilai parameter statis diubah, instans klaster harus di-boot ulang. Reboot setiap instans di klaster Anda yang Anda inginkan untuk menerapkan parameter statis yang diperbarui.

Untuk Linux, macOS, atau Unix:

```
aws docdb reboot-db-instance \
        --db-instance-identifier sample-cluster-instance
```

Untuk Windows:

```
aws docdb reboot-db-instance ^
        --db-instance-identifier sample-cluster-instance
```

Menghapus grup parameter cluster Amazon DocumentDB

Anda dapat menghapus grup parameter cluster Amazon DocumentDB kustom menggunakan Konsol Manajemen AWS atau AWS Command Line Interface ().AWS CLI Anda tidak dapat menghapus grup parameter klaster `default.docdb4.0`.

Using the Konsol Manajemen AWS

Untuk menghapus grup parameter klaster

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Grup parameter.

> **Tip**
>
> Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu (≡) di pojok kiri atas halaman.

3. Di panel Grup parameter, pilih tombol radio di sebelah kiri grup parameter klaster yang ingin Anda hapus.
4. Pilih Tindakan, lalu pilih Hapus.
5. Di panel konfirmasi Hapus, pilih Hapus untuk menghapus grup parameter klaster. Untuk mempertahankan grup parameter klaster, pilih Batal.

Using the AWS CLI

Untuk menghapus grup parameter klaster, gunakan operasi `delete-db-cluster-parameter-group` dengan parameter berikut.

- **`--db-cluster-parameter-group-name`** — Diperlukan. Nama grup parameter klaster yang akan dihapus. Ini harus menjadi grup parameter klaster yang sudah ada. Anda tidak dapat menghapus grup parameter *`default.docdb4.0`* cluster.

Example- Menghapus grup parameter klaster

Contoh berikut memandu Anda menjalani tiga langkah untuk menghapus grup parameter klaster:

1. Menemukan nama grup parameter klaster yang ingin Anda hapus.
2. Menghapus grup parameter klaster yang ditentukan.
3. Memverifikasi bahwa grup parameter klaster telah dihapus.

1. Temukan nama grup parameter klaster yang ingin Anda hapus.

Kode berikut mencantumkan nama semua grup parameter klaster.

Untuk Linux, macOS, atau Unix:

```
aws docdb describe-db-cluster-parameter-groups \
```

```
        --query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

Untuk Windows:

```
aws docdb describe-db-cluster-parameter-groups ^
        --query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

Output dari operasi sebelumnya adalah daftar nama grup parameter klaster yang mirip dengan berikut ini (format JSON).

```
[
        [
            "default.docdb4.0"
        ],
        [
            "sample-parameter-group"
        ],
        [
            "sample-parameter-group-copy"
        ]
   ]
```

2. Menghapus grup parameter klaster tertentu.

Kode berikut menghapus grup parameter klaster `sample-parameter-group-copy`.

Untuk Linux, macOS, atau Unix:

```
aws docdb delete-db-cluster-parameter-group \
        --db-cluster-parameter-group-name sample-parameter-group-copy
```

Untuk Windows:

```
aws docdb delete-db-cluster-parameter-group ^
        --db-cluster-parameter-group-name sample-parameter-group-copy
```

Tidak ada output dari operasi ini.

3. Verifikasi bahwa grup parameter klaster yang ditentukan telah dihapus.

Kode berikut mencantumkan nama semua grup parameter klaster yang tersisa.

Untuk Linux, macOS, atau Unix:

```
aws docdb describe-db-cluster-parameter-groups \
       --query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

Untuk Windows:

```
aws docdb describe-db-cluster-parameter-groups ^
       --query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

Output dari operasi sebelumnya adalah daftar grup parameter klaster yang mirip dengan berikut ini (format JSON). Grup parameter klaster yang baru saja Anda hapus tidak boleh ada dalam daftar.

Output dari operasi ini terlihat seperti berikut (format JSON).

```
[
       [
           "default.docdb4.0"
       ],
       [
           "sample-parameter-group"
       ]
   ]
```

Referensi parameter cluster Amazon DocumentDB

Saat Anda mengubah parameter dinamis dan menyimpan grup parameter klaster, perubahan akan segera diterapkan terlepas dari pengaturan Terapkan segera. Saat Anda mengubah parameter statis dan menyimpan grup parameter klaster, perubahan parameter akan berlaku setelah Anda melakukan reboot instans secara manual. Anda dapat melakukan reboot instans menggunakan konsol Amazon DocumentDB atau dengan memanggil `reboot-db-instance` secara eksplisit.

Tabel berikut menunjukkan parameter yang berlaku untuk semua instans di klaster Amazon DocumentDB.

Parameter tingkat klaster Amazon DocumentDB

Parameter	Nilai default	Nilai valid	Dapat diubah	Tipe Penerapan	Tipe data	Deskripsi
`audit_logs`	dinonaktifkan	diaktifkan, dinonaktifkan, ddl, dml_read, dml_write, semua, tidak ada	Ya	Dinamis	String	Menentukan apakah log CloudWatch audit Amazon diaktifkan. • **`enabled`**—log CloudWatch audit diaktifkan. • **`disabled`**—log CloudWatch audit dinonaktifkan. • **`ddl`**—audit untuk acara DDL diaktifkan. • **`dml_read`**—audit untuk acara baca

Parameter	Nilai default	Nilai valid	Dapat diubah	Tipe Penerapan	Tipe data	Deskripsi
						DMLdiaktifkan. • **dml_write** — audit untuk acara penulisan DMLdiaktifkan. • **all**— audit untuk semua peristiwa database diaktifkan. • **none**— audit dinonaktifkan.

Parameter	Nilai default	Nilai valid	Dapat diubah	Tipe Penerapan	Tipe data	Deskripsi
change_stream_log_retention_duration	10800	3600-604800	Ya	Dinamis	Bilangan Bulat	Menentukan durasi waktu (dalam detik) bahwa log aliran perubahan dipertahankan dan dapat digunakan.

Parameter	Nilai default	Nilai valid	Dapat diubah	Tipe Penerapan	Tipe data	Deskripsi
`default_collection_compression`	dinonaktifkan	diaktifkan, dinonaktifkan (Amazon DocumentDB 5.0) / zstd, lz4, tidak ada (Amazon) DocumentDB8.0	Ya	Dinamis	String	Mendefinisikan pengaturan kompresi default untuk koleksi baru dalam sebuah cluster • **`enabled`**- Kompresi Lz4 diaktifkan secara default di Amazon DocumentDB 5.0. kompresi zstd diaktifkan secara default di Amazon DocumentDB 8.0. • **`disabled, none`**— kompresi dinonaktifkan secara

Parameter	Nilai default	Nilai valid	Dapat diubah	Tipe Penerapan	Tipe data	Deskripsi
						default (dinonaktifkan di Amazon DocumentDB 5.0, tidak ada di Amazon DocumentDB 8.0).

Parameter	Nilai default	Nilai valid	Dapat diubah	Tipe Penerapan	Tipe data	Deskripsi
`profiler`	dinonaktifkan	diaktifkan, dinonaktifkan	Ya	Dinamis	String	Mengaktifkan pembuatan profil untuk operasi yang lambat. • **`enabled`**— operasi yang memakan waktu lebih lama dari nilai ambang batas yang ditentukan pelanggan (misalnya, 100 ms) dicatat ke Amazon Logs. CloudWatch • **`disabled`**— operasi lambat

Parameter	Nilai default	Nilai valid	Dapat diubah	Tipe Penerapan	Tipe data	Deskripsi
						tidak dicatat ke CloudWatch Log.
`profiler_sampling_rate`	1.0	0,0-1,0	Ya	Dinamis	Desimal	Menentukan laju pengambilan sampel untuk operasi yang dicatat.

Parameter	Nilai default	Nilai valid	Dapat diubah	Tipe Penerapan	Tipe data	Deskripsi
`profiler_threshold_ms`	100	50-2147483646	Ya	Dinamis	Bilangan Bulat	Mendefinisikan ambang batas untuk `profiler`. • Semua operasi yang lebih besar dari `profiler_threshold_ms` yang dicatat ke CloudWatch Log.

Parameter	Nilai default	Nilai valid	Dapat diubah	Tipe Penerapan	Tipe data	Deskripsi
`planner_version`	3.0	1.0, 2.0, 3.0	Ya	Dinamis	Desimal	Mendefinisikan versi perencana kueri yang akan digunakan untuk kueri. • **`1.0`**- Didukung di Amazon DocumentDB 5.0. • **`2.0`**— Perbaikan untuk menemukan dan memperbarui operator. Didukung di Amazon DocumentDB 5.0. • **`3.0`**— Perbaikan untuk menemukan,

Parameter	Nilai default	Nilai valid	Dapat diubah	Tipe Penerapan	Tipe data	Deskripsi
						memperbarui, dan agregasi operator. Perencana kueri default di Amazon DocumentDB 8.0.

Parameter	Nilai default	Nilai valid	Dapat diubah	Tipe Penerapan	Tipe data	Deskripsi
`tls`	diaktifkan	diaktifkan, dinonaktifkan, fips-140-3, tls1.2 +, tls1.3 +	Ya	Statis	String	Menentukan apakah koneksi Transport Layer Security (TLS) diperlukan. • **`enabled`**—Koneksi TLS diperlukan untuk terhubung. Untuk Amazon DocumentDB 5.0 dan yang lebih lama, ini memungkinkan TLS 1.0 hingga 1.3, tetapi kami merekomendasikan TLS 1.2 atau lebih

Parameter	Nilai default	Nilai valid	Dapat diubah	Tipe Penerapan	Tipe data	Deskripsi
						tinggi. Hanya TLS 1.2 dan yang lebih tinggi yang didukung di Amazon DocumentDB 5.0 dimulai dengan versi minor 5.0.1, Amazon DocumentDB 8.0 dan yang lebih baru. • **`disabled`**— Koneksi TLS tidak dapat digunakan untuk terhubung.

Parameter	Nilai default	Nilai valid	Dapat diubah	Tipe Penerapan	Tipe data	Deskripsi
						• **`fips-140-3`** Koneksi TLS dengan atribut Federal Information Processing Standards (FIPS) diperlukan untuk terhubung. Cluster hanya menerima koneksi aman per Publikasi FIPS 140-3. Ini hanya didukung mulai dengan cluster Amazon DocumentDB 5.0 (versi engine

Parameter	Nilai default	Nilai valid	Dapat diubah	Tipe Penerapan	Tipe data	Deskripsi
						3.0.3727) di Wilayah ini: ca-central-1, us-west-2, us-east-1, us-east-1, us-east-2, us-east-2, us-gov-east-1, us-gov-east-1. • **`tls1.2+`**— Koneksi TLS menggunakan TLS versi 1.2 ke atas diperlukan untuk terhubung. Ini hanya didukung mulai dengan

Parameter	Nilai default	Nilai valid	Dapat diubah	Tipe Penerapan	Tipe data	Deskripsi
						Amazon DocumentDB 4.0 (versi mesin 2.0.10980) dan Amazon DocumentDB 5.0 (versi mesin 3.0.11051). • **`tls1.3+`**—Koneksi TLS menggunakan TLS versi 1.3 ke atas diperlukan untuk terhubung. Ini hanya didukung mulai dengan Amazon DocumentDB 4.0 (versi mesin

Parameter	Nilai default	Nilai valid	Dapat diubah	Tipe Penerapan	Tipe data	Deskripsi
						2.0.10980) dan Amazon DocumentDB 5.0 (versi mesin 3.0.11051).
`ttl_monitor`	diaktifkan	diaktifkan, dinonaktifkan	Ya	Dinamis	String	Menentukan apakah pemantauan Time to Live (TTL) diaktifkan untuk klaster. • **`enabled`**- Pemantauan TTL diaktifkan. • **`disabled`**- Pemantauan TTL dinonaktifkan.

Memodifikasi parameter cluster Amazon DocumentDB

Di Amazon DocumentDB, grup parameter klaster terdiri dari parameter yang berlaku untuk semua instans yang Anda buat di klaster. Untuk grup parameter klaster kustom, Anda dapat mengubah nilai

parameter kapan saja atau mengatur ulang semua nilai parameter ke default untuk grup parameter yang Anda buat. Bagian ini menjelaskan cara melihat parameter yang membentuk grup parameter klaster Amazon DocumentDB dan nilainya, dan bagaimana Anda dapat mengubah atau memperbarui nilai ini.

Parameter bisa dinamis atau statis. Saat Anda mengubah parameter dinamis dan menyimpan grup parameter klaster, perubahan akan segera diterapkan terlepas dari pengaturan `Apply Immediately`. Saat Anda mengubah parameter statis dan menyimpan grup parameter klaster, perubahan parameter hanya berlaku setelah Anda melakukan reboot instans secara manual.

Melihat parameter grup parameter cluster Amazon DocumentDB

Anda dapat melihat parameter cluster Amazon DocumentDB dan nilainya menggunakan atau. Konsol Manajemen AWS AWS CLI

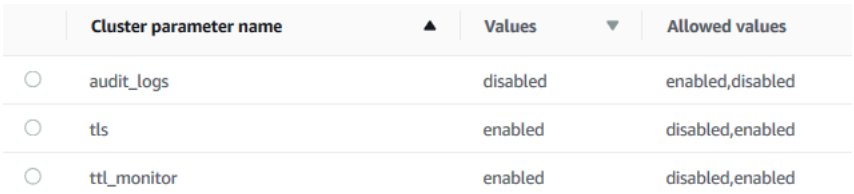
Using the Konsol Manajemen AWS

Untuk melihat detail grup parameter klaster

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Grup parameter.

> **Tip**
>
> Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu (≡) di pojok kiri atas halaman.

3. Di panel Grup parameter, pilih nama grup parameter klaster yang ingin Anda lihat detailnya.
4. Halaman yang dihasilkan menunjukkan nilai berikut untuk setiap parameter: nama parameter, nilai saat ini, nilai yang diizinkan, apakah parameter dapat dimodifikasi, jenis penerapan, tipe data, dan deskripsi.



	Cluster parameter name	Values	Allowed values
○	audit_logs	disabled	enabled,disabled
○	tls	enabled	disabled,enabled
○	ttl_monitor	enabled	disabled,enabled

Using the AWS CLI

Untuk melihat parameter grup parameter klaster dan nilainya, gunakan operasi `describe-db-cluster-parameters` dengan parameter berikut.

- **`--db-cluster-parameter-group-name`** — Diperlukan. Nama grup parameter klaster yang Anda inginkan daftar parameternya secara mendetail.
- **`--source`** – Opsional. Jika disediakan, mengembalikan hanya parameter untuk sumber tertentu. Sumber parameter dapat berupa `engine-default`, `system`, atau `user`.

Example

Kode berikut mencantumkan parameter dan nilainya untuk grup parameter `custom3-6-param-grp`. Untuk informasi lebih lanjut tentang grup parameter, hilangkan garis `--query`. Untuk informasi tentang semua grup parameter, hilangkan garis `--db-cluster-parameter-group-name`.

Untuk Linux, macOS, atau Unix:

```
aws docdb describe-db-cluster-parameters \
   --db-cluster-parameter-group-name custom3-6-param-grp \
   --query 'Parameters[*].[ParameterName,ParameterValue]'
```

Untuk Windows:

```
aws docdb describe-db-cluster-parameters ^
   --db-cluster-parameter-group-name custom3-6-param-grp ^
   --query 'Parameters[*].[ParameterName,ParameterValue]'
```

Output dari operasi ini terlihat seperti berikut (format JSON).

```
[
    [
        "audit_logs",
        "disabled"
    ],
    [
        "tls",
        "enabled"
    ],
    [
```

```
        "ttl_monitor",
        "enabled"
    ]
]
```

Memodifikasi parameter grup parameter cluster Amazon DocumentDB

Anda dapat memodifikasi parameter grup parameter menggunakan Konsol Manajemen AWS atau AWS CLI.

Using the Konsol Manajemen AWS

Untuk memperbarui parameter grup parameter klaster

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Grup parameter.

   > **Tip**
   >
   > Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu (≡) di pojok kiri atas halaman.

3. Di panel Grup parameter, pilih grup parameter klaster yang ingin Anda perbarui parameternya.
4. Halaman yang dihasilkan menunjukkan parameter dan detail yang sesuai untuk grup parameter klaster ini. Pilih parameter untuk diperbarui.
5. Di kanan atas halaman, pilih Edit untuk mengubah nilai parameter. Untuk informasi selengkapnya tentang tipe parameter klaster, lihat Referensi parameter cluster Amazon DocumentDB.
6. Buat perubahan Anda, lalu pilih Ubah parameter klaster untuk menyimpan perubahan. Untuk membuang perubahan Anda, pilih Batal.

Using the AWS CLI

Untuk mengubah parameter grup parameter klaster, gunakan operasi `modify-db-cluster-parameter-group` dengan parameter berikut:

- **`--db-cluster-parameter-group-name`** — Diperlukan. Nama grup parameter klaster yang Anda modifikasi.
- **`--parameters`** — Diperlukan. Parameter atau parameter yang Anda modifikasi. Setiap entri parameter harus menyertakan hal berikut:
  - **`ParameterName`** — Nama parameter yang Anda ubah.
  - **`ParameterValue`** — Nilai baru untuk parameter ini.
  - **`ApplyMethod`** — Bagaimana Anda ingin perubahan pada parameter ini diterapkan. Nilai yang diizinkan adalah `immediate` dan `pending-reboot`.

    > **Note**
    >
    > Parameter dengan `ApplyType` dari `static` harus memiliki `ApplyMethod` dari `pending-reboot`.

Untuk mengubah nilai parameter kelompok parameter cluster (AWS CLI)

Contoh berikut mengubah parameter `tls`.

1. Buat daftar parameter dan nilainya **`sample-parameter-group`**

   Untuk Linux, macOS, atau Unix:

   ```
   aws docdb describe-db-cluster-parameters \
       --db-cluster-parameter-group-name sample-parameter-group
   ```

   Untuk Windows:

   ```
   aws docdb describe-db-cluster-parameters ^
       --db-cluster-parameter-group-name sample-parameter-group
   ```

   Output dari operasi ini terlihat seperti berikut (format 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. Ubah parameter **`tls`** sehingga nilainya menjadi **`disabled`**. Anda tidak dapat mengubah `ApplyMethod` karena `ApplyType` adalah `static`.

   Untuk Linux, macOS, atau Unix:

```
aws docdb modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name sample-parameter-group \
    --parameters
 "ParameterName"=tls,ParameterValue=disabled,ApplyMethod=pending-reboot"
```

   Untuk Windows:

```
aws docdb modify-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name sample-parameter-group ^
    --parameters "ParameterName=tls,ParameterValue=disabled,ApplyMethod=pending-
reboot"
```

   Output dari operasi ini terlihat seperti berikut (format JSON).

```
{
```

```
    "DBClusterParameterGroupName": "sample-parameter-group"
}
```

3. Tunggu setidaknya 5 menit.
4. Buat daftar nilai parameter dari**`sample-parameter-group`**.

   Untuk Linux, macOS, atau Unix:

```
aws docdb describe-db-cluster-parameters \
    --db-cluster-parameter-group-name sample-parameter-group
```

   Untuk Windows:

```
aws docdb describe-db-cluster-parameters ^
    --db-cluster-parameter-group-name sample-parameter-group
```

   Output dari operasi ini terlihat seperti berikut (format 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"
        }
```

```
    ]
}
```

Memahami titik akhir Amazon DocumentDB

Anda dapat menggunakan titik akhir Amazon DocumentDB (dengan kompatibilitas MongoDB) untuk terhubung ke klaster atau instans. Amazon DocumentDB memiliki tiga jenis titik akhir yang berbeda, masing-masing dengan tujuannya sendiri.

Topik

- Menemukan titik akhir klaster
- Menemukan titik akhir instance
- Menghubungkan ke titik akhir

Titik akhir klaster

Titik akhir klaster adalah titik akhir untuk klaster Amazon DocumentDB yang tersambung ke instans utama saat ini untuk klaster tersebut. Setiap klaster Amazon DocumentDB memiliki titik akhir klaster tunggal dan satu instans utama. Jika terjadi failover, titik akhir klaster dipetakan ulang ke instans utama baru.

Reader endpoint

Reader endpoint adalah titik akhir untuk klaster Amazon DocumentDB yang terhubung ke salah satu replika yang tersedia untuk klaster tersebut. Setiap klaster Amazon DocumentDB memiliki reader endpoint. Jika ada lebih dari satu replika, reader endpoint mengarahkan setiap permintaan koneksi ke salah satu replika Amazon DocumentDB.

Titik akhir instans

Titik akhir instans adalah titik akhir yang terhubung ke instans tertentu. Setiap instans dalam sebuah klaster, terlepas dari apakah itu instans primer atau replika, memiliki titik akhir instans uniknya sendiri. Yang terbaik adalah tidak menggunakan titik akhir instans di aplikasi Anda. Ini karena mereka dapat mengubah peran jika terjadi failover, sehingga memerlukan perubahan kode dalam aplikasi Anda.

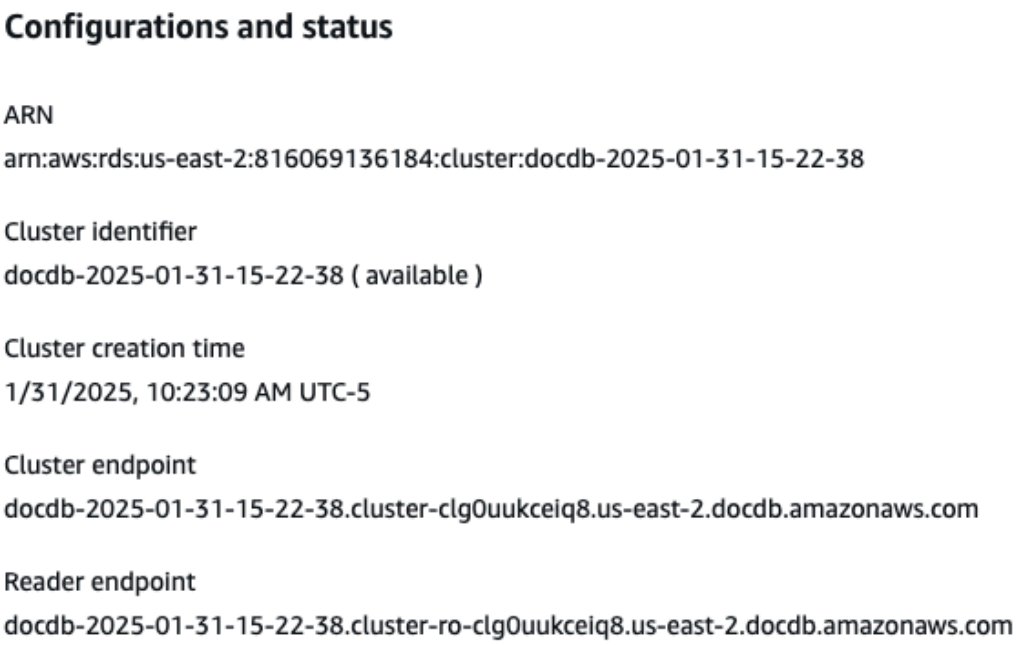
Menemukan titik akhir klaster

Anda dapat menemukan titik akhir klaster dan reader endpoint pada suatu klaster dengan menggunakan konsol Amazon DocumentDB atau AWS CLI.

Using the Konsol Manajemen AWS

Untuk menemukan titik akhir cluster menggunakan konsol:

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Pada panel navigasi, silakan pilih Klaster.
3. Dari daftar klaster, pilih nama klaster yang Anda minati.
4. Pada halaman detail cluster, pilih tab Konfigurasi. Di bagian Konfigurasi dan status, Anda akan menemukan titik akhir Cluster dan titik akhir Pembaca.



5. Untuk terhubung ke cluster ini, pilih tab Konektivitas & keamanan. Temukan string koneksi untuk `mongo` shell dan string koneksi yang dapat digunakan dalam kode aplikasi untuk terhubung ke cluster Anda.



Using the AWS CLI

Untuk menemukan titik akhir cluster dan pembaca untuk cluster Anda menggunakan AWS CLI, jalankan `describe-db-clusters` perintah dengan parameter ini.

Parameter

- **`--db-cluster-identifier`**—Opsional. Menentukan klaster untuk mengembalikan titik akhir. Jika dihilangkan, menampilkan titik akhir hingga 100 klaster Anda.
- **`--query`**—Opsional. Menentukan bidang yang akan menampilkan. Bermanfaat dengan mengurangi jumlah data yang Anda butuhkan untuk melihat serta menemukan titik akhir. Jika dihilangkan, semua informasi tentang klaster dikembalikan.
- **`--region`**—Opsional. Gunakan parameter `--region` untuk menentukan Wilayah tempat Anda ingin menerapkan perintah. Jika dihilangkan, Wilayah default Anda digunakan.

Example

Contoh berikut mengembalikan `DBClusterIdentifier`, titik akhir (titik akhir klaster), dan `ReaderEndpoint` untuk `sample-cluster`.

Untuk Linux, macOS, atau Unix:

```
aws docdb describe-db-clusters \
   --region us-east-1 \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].[DBClusterIdentifier,Port,Endpoint,ReaderEndpoint]'
```

Untuk Windows:

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

```
    --region us-east-1 ^
    --db-cluster-identifier sample-cluster ^
    --query 'DBClusters[*].[DBClusterIdentifier,Port,Endpoint,ReaderEndpoint]'
```

Output dari operasi ini terlihat seperti berikut (format 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"
  ]
]
```

Sekarang setelah Anda memiliki titik akhir klaster, Anda dapat terhubung ke klaster menggunakan salah satu `mongo` atau `mongodb`. Untuk informasi selengkapnya, lihat Menghubungkan ke titik akhir.

Menemukan titik akhir instance

Anda dapat menemukan titik akhir untuk instans dengan menggunakan konsol Amazon DocumentDB atau AWS CLI.

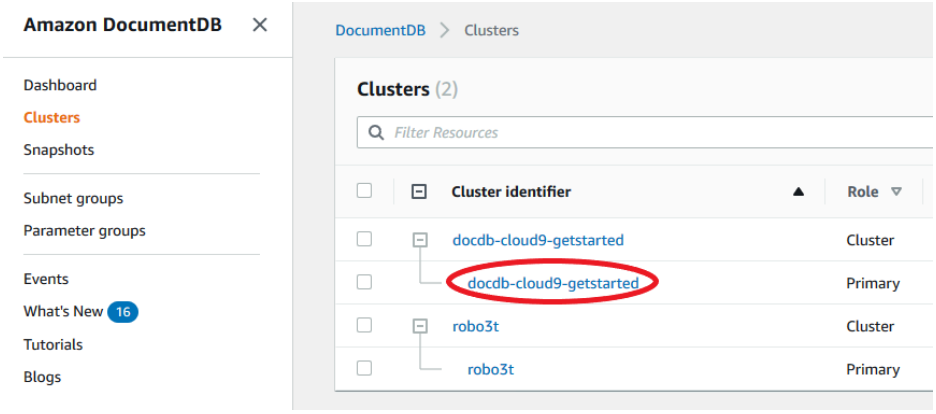
Using the Konsol Manajemen AWS

Untuk menemukan titik akhir instans dengan menggunakan konsol

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Klaster.

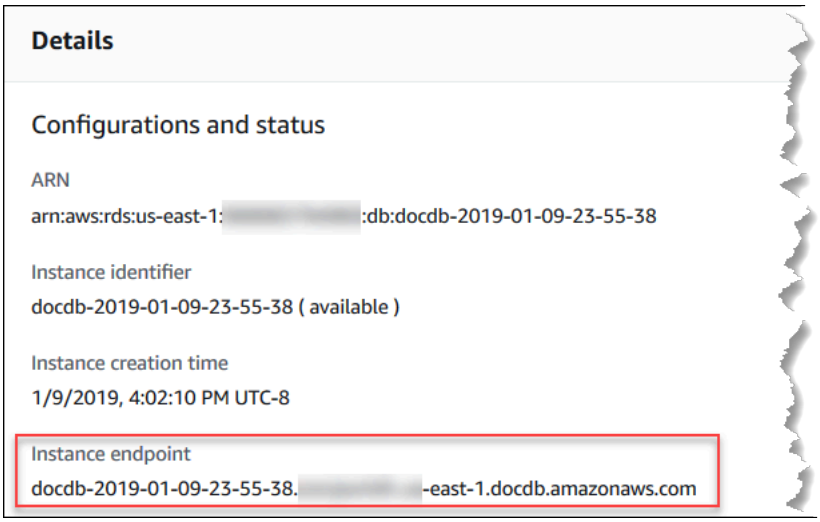
> **Tip**
>
> Jika Anda tidak melihat panel navigasi pada sisi kiri layar Anda, pilih ikon menu (≡) di sudut kiri atas halaman.

3. Di kotak navigasi Klaster, Anda akan melihat kolom Pengidentifikasi klaster. Instance Anda tercantum di bawah cluster, mirip dengan tangkapan layar berikut.



4. Centang kotak di sebelah kiri instans yang Anda minati.

5. Gulir ke bawah ke bagian Detail lalu temukan titik akhir instans.



6. Untuk terhubung ke instans ini, gulir ke atas ke bagian Koneksi. Temukan string koneksi untuk shell `mongo` dan string koneksi yang dapat digunakan dalam kode aplikasi Anda untuk terhubung ke instans Anda.



Using the AWS CLI

Untuk menemukan titik akhir instance menggunakan AWS CLI, jalankan perintah berikut dengan argumen ini.

Argumen

- **`--db-instance-identifier`**—Opsional. Menentukan instans untuk menampilkan titik akhir. Jika dihilangkan, mengembalikan titik akhir hingga 100 instans Anda.
- **`--query`**—Opsional. Menentukan bidang yang akan menampilkan. Bermanfaat dengan mengurangi jumlah data yang Anda butuhkan untuk melihat serta menemukan titik akhir. Jika dihilangkan, semua informasi pada sebuah instans dikembalikan. Bidang `Endpoint` memiliki tiga anggota, jadi mencantumkannya dalam kueri seperti pada contoh berikut mengembalikan ketiga anggota. Jika Anda hanya tertarik pada beberapa anggota `Endpoint`, ganti `Endpoint` pada kueri dengan anggota yang Anda minati, seperti pada contoh kedua.
- **`--region`**—Opsional. Gunakan parameter `--region` untuk menentukan Wilayah tempat Anda ingin menerapkan perintah. Jika dihilangkan, Wilayah default Anda digunakan.

Example

Untuk Linux, macOS, atau Unix:

```
aws docdb describe-db-instances \
    --region us-east-1 \
    --db-instance-identifier sample-cluster-instance \
    --query 'DBInstances[*].[DBInstanceIdentifier,Endpoint]'
```

Untuk Windows:

```
aws docdb describe-db-instances ^
    --region us-east-1 ^
    --db-instance-identifier sample-cluster-instance ^
    --query 'DBInstances[*].[DBInstanceIdentifier,Endpoint]'
```

Output dari operasi ini terlihat seperti berikut (format JSON).

```
[
    [
```

```
        "sample-cluster-instance",
        {
            "Port": 27017,
            "Address": "sample-cluster-instance.corcjozrlsfc.us-
east-1.docdb.amazonaws.com",
            "HostedZoneId": "Z2R2ITUGPM61AM"
        }
    ]
]
```

Mengurangi output untuk menghilangkan `HostedZoneId` pada titik akhir, Anda dapat memodifikasi kueri Anda dengan menentukan `Endpoint.Port` dan `Endpoint.Address`.

Untuk Linux, macOS, atau Unix:

```
aws docdb describe-db-instances \
    --region us-east-1 \
    --db-instance-identifier sample-cluster-instance \
    --query 'DBInstances[*].[DBInstanceIdentifier,Endpoint.Port,Endpoint.Address]'
```

Untuk Windows:

```
aws docdb describe-db-instances ^
    --region us-east-1 ^
    --db-instance-identifier sample-cluster-instance ^
    --query 'DBInstances[*].[DBInstanceIdentifier,Endpoint.Port,Endpoint.Address]'
```

Output dari operasi ini terlihat seperti berikut (format JSON).

```
[
    [
        "sample-cluster-instance",
        27017,
        "sample-cluster-instance.corcjozrlsfc.us-east-1.docdb.amazonaws.com"
    ]
]
```

Sekarang setelah Anda memiliki titik akhir instans, Anda dapat terhubung ke instans dengan menggunakan `mongo` atau `mongodb`. Untuk informasi selengkapnya, lihat Menghubungkan ke titik akhir.

Menghubungkan ke titik akhir

Setelah Anda memiliki titik akhir, baik klaster atau instans, Anda dapat menghubungkannya dengan menggunakan shell mongo atau string koneksi.

Menghubungkan menggunakan cangkang mongo

Gunakan struktur berikut untuk membuat string yang perlu Anda sambungkan ke klaster atau instans Anda dengan menggunakan shell mongo:

```
mongo \
    --ssl \
    --host Endpoint:Port \
    --sslCAFile global-bundle.pem \
    --username UserName \
    --password Password
```

contoh cangkang **mongo**

Sambungkan ke klaster:

```
mongo \
    --ssl \
    --host sample-cluster.corcjozrlsfc.us-east-1.docdb.amazonaws.com:27017 \
    --sslCAFile global-bundle.pem \
    --username UserName \
    --password Password
```

Sambungkan ke instans:

```
mongo \
    --ssl \
    --host sample-cluster-instance.corcjozrlsfc.us-east-1.docdb.amazonaws.com:27017 \
    --sslCAFile global-bundle.pem \
    --username UserName \
    --password Password
```

Menghubungkan menggunakan string koneksi

Gunakan struktur berikut untuk membuat string koneksi yang Anda perlukan untuk terhubung ke klater atau instans Anda.

```
mongodb://UserName:Password@endpoint:port?replicaSet=rs0&ssl_ca_certs=global-
bundle.pem
```

Contoh string koneksi

Sambungkan ke klaster:

```
mongodb://UserName:Password@sample-cluster.cluster-corlsfccjozr.us-
east-1.docdb.amazonaws.com:27017?replicaSet=rs0&ssl_ca_certs=global-bundle.pem
```

Sambungkan ke instans:

```
mongodb://UserName:Password@sample-cluster-instance.cluster-corlsfccjozr.us-
east-1.docdb.amazonaws.com:27017?replicaSet=rs0&ssl_ca_certs=global-bundle.pem
```

Memahami Amazon DocumentDB Amazon Resource Name (ARN)

Sumber daya yang Anda buat masing-masing AWS diidentifikasi secara unik dengan Nama Sumber Daya Amazon (ARN). Untuk operasi Amazon DocumentDB (dengan kompatibilitas MongoDB) tertentu, Anda harus secara unik mengidentifikasi sumber daya Amazon DocumentDB dengan menentukan ARN-nya. Misalnya, saat Anda menambahkan tag ke sumber daya, Anda harus menyediakan ARN sumber daya.

Topik

- Membangun ARN untuk sumber daya Amazon DocumentDB
- Menemukan ARN sumber daya Amazon DocumentDB

Membangun ARN untuk sumber daya Amazon DocumentDB

Anda dapat membangun ARN untuk sumber daya Amazon DocumentDB menggunakan sintaks berikut. Amazon DocumentDB membagikan format ARN Amazon Relational Database Service (Amazon RDS). ARN Amazon DocumentDB terdiri atas `rds` dan bukan `docdb`.

`arn:aws:rds:region:account_number:resource_type:resource_id`

Nama Wilayah	Region	Availability Zone (penghitungan)
AS Timur (Ohio)	us-east-2	3
AS Timur (Virginia Utara)	us-east-1	6
AS Barat (Oregon)	us-west-2	4
Africa (Cape Town)	af-south-1	3
Amerika Selatan (Sao Paulo)	sa-east-1	3
Asia Pasifik (Hong Kong)	ap-east-1	3
Asia Pasifik (Hyderabad)	ap-south-2	3
Asia Pasifik (Malaysia)	ap-southeast-5	3
Asia Pasifik (Mumbai)	ap-south-1	3
Asia Pasifik (Osaka)	ap-northeast-3	3
Asia Pasifik (Seoul)	ap-northeast-2	4
Asia Pasifik (Singapura)	ap-southeast-1	3
Asia Pasifik (Sydney)	ap-southeast-2	3
Asia Pasifik (Jakarta)	ap-southeast-3	3
Asia Pacific (Melbourne)	ap-southeast-4	3
Asia Pasifik (Thailand)	ap-southeast-7	3

Nama Wilayah	Region	Availability Zone (penghitungan)
Asia Pasifik (Tokyo)	`ap-northeast-1`	3
Kanada (Pusat)	`ca-central-1`	3
Kanada Barat (Calgary)	`ca-west-1`	3
Wilayah Tiongkok (Beijing)	`cn-north-1`	3
Tiongkok (Ningxia)	`cn-northwest-1`	3
Eropa (Frankfurt)	`eu-central-1`	3
Europe (Zurich)	`eu-central-2`	3
Eropa (Irlandia)	`eu-west-1`	3
Eropa (London)	`eu-west-2`	3
Europe (Milan)	`eu-south-1`	3
Eropa (Paris)	`eu-west-3`	3
Eropa (Spanyol)	`eu-south-2`	3
Eropa (Stockholm)	`eu-north-1`	3
Meksiko (Tengah)	`mx-central-1`	3
Timur Tengah (UAE)	`me-central-1`	3
Israel (Tel Aviv)	`il-central-1`	3
AWS GovCloud (US-West)	`us-gov-west-1`	3
AWS GovCloud (US-East)	`us-gov-east-1`	3

> **Note**
>
> Arsitektur Amazon DocumentDB memisahkan penyimpanan dan komputasi. Untuk lapisan penyimpanan, Amazon DocumentDB mereplikasi enam salinan data Anda di AWS tiga Availability Zones (AZ). AZ yang tercantum dalam tabel di atas adalah jumlah AZ yang dapat Anda gunakan di region tertentu untuk menyediakan instans komputasi. Sebagai contoh, jika Anda meluncurkan klaster Amazon DocumentDB di ap-northeast-1, penyimpanan Anda akan direplikasi enam cara di tiga AZ tetapi instans komputasi Anda hanya akan tersedia di dua AZ.

Tabel berikut menunjukkan format yang harus Anda gunakan saat membangun ARN untuk jenis sumber daya Amazon DocumentDB tertentu. Amazon DocumentDB membagikan format Amazon RDS ARN. ARN Amazon DocumentDB terdiri atas `rds` dan bukan `docdb`.

Jenis Sumber Daya	Format/Contoh ARN
Contoh (db)	`arn:aws:rds:` *`region`*`:`*`account_number`* `:db:`*`resource_id`* `arn:aws:rds:us-east-1:` *`1234567890`* `:db:`*`sample-db-instance`*
Klaster (`cluster`)	`arn:aws:rds:` *`region`*`:`*`account_number`* `:cluster:` *`resource_id`* `arn:aws:rds:us-east-1:` *`1234567890`* `:cluster:` *`sample-db-cluster`*
Grup parameter klaster (`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`*
Grup keamanan (`secgrp`)	`arn:aws:rds:` *`region`*`:`*`account_number`* `:secgrp:`*`resource_id`*

Jenis Sumber Daya	Format/Contoh ARN
	`arn:aws:rds:us-east-1:` *1234567890* `:secgrp:`*sample-public-secgrp*
Snapshot klaster (`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*
Grup subnet (`subgrp`)	`arn:aws:rds:` *region*`:`*account_number* `:subgrp:`*resource_id* `arn:aws:rds:us-east-1:` *1234567890* `:subgrp:`*sample-subnet-10*

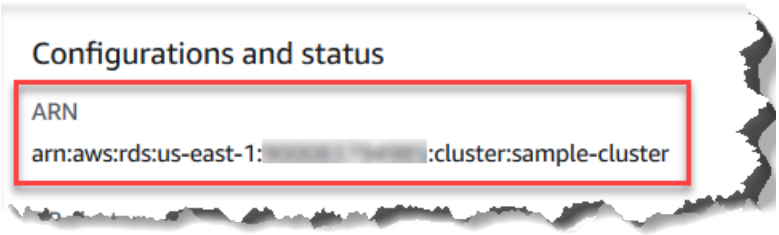
Menemukan ARN sumber daya Amazon DocumentDB

Anda dapat menemukan ARN dari sumber daya Amazon DocumentDB menggunakan atau file. Konsol Manajemen AWS AWS CLI

Using the Konsol Manajemen AWS

Untuk menemukan ARN dengan menggunakan konsol, arahkan navigasi ke sumber daya yang Anda inginkan untuk ARN, dan lihat detail untuk sumber daya tersebut.

Misalnya, Anda bisa mendapatkan ARN untuk klaster dengan memilih tab Konfigurasi pada halaman detail cluster. ARN dapat ditemukan di bagian Konfigurasi dan status, seperti yang ditunjukkan pada gambar berikut.



Using the AWS CLI

Untuk mendapatkan ARN menggunakan sumber daya Amazon DocumentDB tertentu, gunakan `describe` operasi AWS CLI untuk sumber daya tersebut. Tabel berikut menunjukkan setiap AWS CLI operasi dan properti ARN yang digunakan dengan operasi untuk mendapatkan ARN.

AWS CLI Perintah	Properti 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- Menemukan ARN untuk klaster

AWS CLI Operasi berikut menemukan ARN untuk cluster. `sample-cluster`

Untuk Linux, macOS, atau Unix:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].DBClusterArn'
```

Untuk Windows:

```
aws docdb describe-db-clusters ^
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].DBClusterArn'
```

Output dari operasi ini terlihat seperti berikut (format JSON).

```
[
    "arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster"
]
```

Example- Menemukan ARN untuk beberapa grup parameter

Untuk Linux, macOS, atau Unix:

```
aws docdb describe-db-cluster-parameter-groups \
   --query 'DBClusterParameterGroups[*].DBClusterParameterGroupArn'
```

Untuk Windows:

```
aws docdb describe-db-cluster-parameter-groups ^
   --query 'DBClusterParameterGroups[*].DBClusterParameterGroupArn'
```

Output dari operasi ini terlihat seperti berikut (format 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"
]
```

Menandai sumber daya Amazon DocumentDB

Anda dapat menggunakan tanda Amazon DocumentDB (dengan kompatibilitas MongoDB) untuk menambahkan metadata ke sumber daya Amazon DocumentDB Anda. Tag ini dapat digunakan dengan kebijakan AWS Identity and Access Management (IAM) untuk mengelola akses ke sumber daya Amazon DocumentDB dan untuk mengontrol tindakan apa yang dapat diterapkan ke sumber daya. Anda juga dapat menggunakan tanda untuk melacak biaya dengan mengelompokkan pengeluaran untuk sumber daya serupa yang ditandai.

Anda dapat menandai pada sumber Amazon DocumentDB berikut ini:

- klaster

- Contoh
- Snapshot
- Snapshot klaster
- Grup parameter
- Grup parameter klaster
- Grup keamanan
- Grup subnet

Ikhtisar tag sumber daya Amazon DocumentDB

Tanda Amazon DocumentDB adalah pasangan nama dan nilai yang Anda tentukan dan berkaitan dengan sumber daya Amazon DocumentDB. Nama ini disebut sebagai kunci. Memberikan nilai untuk kunci bersifat opsional. Anda dapat menggunakan tanda untuk menetapkan informasi tambahan ke sumber daya Amazon DocumentDB. Anda dapat menggunakan kunci tag, misalnya, untuk menentukan kategori, dan nilai tag mungkin merupakan item dalam kategori tersebut. Sebagai contoh, Anda dapat menentukan kunci tanda `project` dan nilai tanda `Salix`, yang menunjukkan bahwa sumber daya Amazon DocumentDB ditugaskan ke proyek Salix. Anda juga bisa menggunakan tanda untuk menunjuk domain Amazon DocumentDB agar digunakan untuk uji atau produksi dengan menggunakan kunci seperti `environment=test` atau `environment=production`. Kami menyarankan Anda untuk menggunakan kumpulan kunci tanda yang konsisten guna mempermudah untuk melacak metadata yang berkaitan dengan sumber daya Amazon DocumentDB.

Anda dapat menggunakan tag untuk mengatur AWS tagihan Anda untuk mencerminkan struktur biaya Anda sendiri. Untuk melakukan ini, daftar untuk mendapatkan Akun AWS tagihan Anda dengan nilai kunci tag disertakan. Kemudian, untuk melihat biaya sumber daya gabungan, atur informasi penagihan Anda sesuai dengan sumber daya Anda dengan nilai kunci tag yang sama. Misalnya, Anda dapat memberikan tag pada beberapa sumber daya dengan nama aplikasi tertentu, kemudian mengatur informasi penagihan untuk melihat biaya total aplikasi tersebut pada beberapa layanan. Untuk informasi selengkapnya, lihat Menggunakan Tanda Alokasi Biaya dalam Panduan Pengguna Manajemen Penagihan dan Biaya AWS .

Setiap sumber daya Amazon DocumentDB memiliki satu rangkaian tanda, yang berisi semua tanda yang ditetapkan ke sumber daya tersebut. Rangkaian tanda dapat berisi 10 tanda, atau dapat kosong. Jika Anda menambahkan tanda ke sumber daya Amazon DocumentDB yang memiliki kunci yang sama dengan tanda yang ada pada sumber daya, nilai lama akan ditimpa dengan nilai baru.

AWS tidak menerapkan makna semantik apa pun pada tag Anda; tag ditafsirkan secara ketat sebagai string karakter. Amazon DocumentDB dapat menetapkan tanda pada instans atau sumber daya Amazon DocumentDB lainnya, bergantung pada pengaturan yang Anda gunakan saat Anda membuat sumber daya. Sebagai contoh, Amazon DocumentDB dapat menambahkan tanda yang menunjukkan bahwa instans digunakan untuk produksi atau untuk pengujian.

Anda dapat menambahkan tanda ke snapshot, namun tagihan Anda tidak akan mencerminkan pengelompokan ini.

Anda dapat menggunakan Konsol Manajemen AWS atau AWS CLI untuk menambah, mencantumkan, dan menghapus tag di sumber daya Amazon DocumentDB. Saat menggunakan AWS CLI, Anda harus memberikan Nama Sumber Daya Amazon (ARN) untuk sumber daya yang ingin Anda gunakan. Untuk informasi selengkapnya tentang Amazon DocumentDB ARN, lihat Memahami Amazon DocumentDB Amazon Resource Name (ARN).

Kendala tag

Kendala berikut ini berlaku untuk tanda Amazon DocumentDB:

- Jumlah maksimum tanda per sumber daya - 10
- Panjang Kunci maksimum – 128 karakter Unicode
- Panjang Nilai maksimum – 256 karakter Unicode
- Karakter yang valid untuk Kunci dan Nilai - huruf besar dan kecil dalam set UTF-8 karakter, digit, spasi, dan karakter berikut: dan (Java regex:`_ . : / = + -`)`@ "^([\\p{L}\\p{Z}\ \p{N}_.:/=+\\-]*)$"`
- Kunci dan nilai tag peka huruf besar/kecil.
- Awalan `aws:` tidak dapat digunakan untuk kunci atau nilai tanda; awalan tersebut disediakan untuk AWS.

Menambahkan dan memperbarui tag pada sumber daya Amazon DocumentDB

Anda dapat menambahkan hingga 10 tag ke sumber daya menggunakan Konsol Manajemen AWS atau AWS CLI.

Using the Konsol Manajemen AWS

Proses untuk menambahkan tanda ke sumber daya serupa terlepas dari sumber daya apa yang Anda tambahkan tanda. Dalam contoh ini, Anda menambahkan tanda ke klaster.

Untuk menambah atau memperbarui tanda ke klaster menggunakan konsol

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Dari panel navigasi, pilih klaster.
3. Pilih nama klaster yang ingin Anda tambahkan tanda.
4. Gulir ke bawah ke bagian Tanda, dan kemudian pilih Edit.
5. Untuk setiap tanda yang ingin Anda tambahkan ke sumber daya ini, lakukan hal berikut ini:
   a. Untuk menambahkan tanda baru, masukkan nama tanda di kotak Kunci. Untuk mengubah nilai tanda, temukan nama tanda di kolom Kunci.
   b. Untuk memberikan nilai baru atau nilai yang diperbarui pada tanda, masukkan nilai untuk tanda di kotak Nilai.
   c. Jika Anda memiliki lebih banyak tanda untuk ditambahkan, pilih Tambahkan. Jika tidak, saat selesai, pilih Simpan.

Using the AWS CLI

Proses untuk menambahkan tanda ke sumber daya serupa terlepas dari apa sumber daya yang Anda tambahkan tanda. Dalam contoh ini, Anda menambahkan tiga tanda ke klaster. Tanda kedua, `key2`, tidak memiliki nilai.

Gunakan AWS CLI operasi `add-tags-to-resource` dengan parameter ini.

Parameter

- **`--resource-name`**—ARN dari sumber daya Amazon DocumentDB yang ingin Anda tambahkan tanda.
- **`--tags`**—Daftar tanda (pasangan kunci-nilai) yang ingin Anda tambahkan ke sumber daya ini dalam format `Key=`*`key-name`*`,Value=`*`tag-value`*.

Example

Untuk Linux, macOS, atau 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
```

Untuk 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
```

Operasi `add-tags-to-resource` tidak menghasilkan keluaran. Untuk melihat hasil operasi, gunakan operasi `list-tags-for-resource`.

Listing tag pada sumber daya Amazon DocumentDB

Anda dapat menggunakan Konsol Manajemen AWS atau AWS CLI untuk mendapatkan daftar tag untuk sumber daya Amazon DocumentDB.

Using the Konsol Manajemen AWS

Proses untuk mencantumkan tanda pada sumber daya serupa terlepas dari sumber daya yang Anda tambahkan tanda. Dalam contoh ini, Anda mencantumkan tanda untuk klaster.

Untuk mencantumkan tanda pada klaster menggunakan konsol

1. Buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Dari panel navigasi, pilih klaster.
3. Pilih nama klaster yang ingin Anda cantumkan tanda.
4. Untuk melihat daftar tanda pada sumber daya ini, gulir ke bawah ke bagian Tanda.

Using the AWS CLI

Proses untuk mencantumkan tanda pada sumber daya serupa terlepas dari sumber daya yang Anda cantumkan tanda. Dalam contoh ini, Anda mencantumkan tanda pada klaster.

Gunakan AWS CLI operasi `list-tags-for-resource` dengan parameter ini.

Parameter

- **`--resource-name`**—Wajib. ARN dari sumber daya Amazon DocumentDB yang ingin Anda cantumkan tanda.

Example

Untuk Linux, macOS, atau Unix:

```
aws docdb list-tags-for-resource \
    --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster
```

Untuk Windows:

```
aws docdb list-tags-for-resource ^
    --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster
```

Keluaran dari operasi ini terlihat seperti berikut ini (format JSON).

```
{
    "TagList": [
        {
            "Key": "key1",
            "Value": "value1"
        },
        {
            "Key": "key2",
            "Value": ""
        },
        {
            "Key": "key3",
            "Value": "value3"
        }
    ]
}
```

Menghapus tag dari sumber daya Amazon DocumentDB

Anda dapat menggunakan Konsol Manajemen AWS atau AWS CLI untuk menghapus tag dari sumber daya Amazon DocumentDB.

Using the Konsol Manajemen AWS

Proses untuk menghapus tanda dari sumber daya serupa terlepas dari sumber daya apa yang Anda tambahkan tanda. Dalam contoh ini, Anda menghapus tanda dari klaster.

Untuk menghapus tanda dari klaster menggunakan konsol

1. Buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Dari panel navigasi, pilih klaster.
3. Pilih nama klaster yang ingin Anda hapus tandanya.
4. Gulir ke bawah ke bagian Tanda, dan kemudian pilih Edit.
5. Jika Anda ingin menghapus semua tanda dari sumber daya ini, pilih Hapus semua. Jika tidak, untuk setiap tanda yang ingin Anda hapus dari sumber daya ini, lakukan hal berikut ini:
   a. Cari nama tanda di kolom Kunci.
   b. Pilih Hapus pada baris yang sama dengan kunci tanda.
   c. Setelah selesai, pilih Simpan.

Using the AWS CLI

Proses untuk menghapus tanda dari sumber daya serupa terlepas dari sumber daya apa yang Anda hapus tandanya. Dalam contoh ini, Anda menghapus tanda dari klaster.

Gunakan AWS CLI operasi `remove-tags-from-resource` dengan parameter ini.

- **`--resource-name`**—Wajib. ARN dari sumber daya Amazon DocumentDB yang ingin Anda hapus tandanya.
- **`--tag-keys`**—Wajib. Daftar kunci tanda yang ingin Anda hapus dari sumber daya ini.

Example

Untuk Linux, macOS, atau Unix:

```
aws docdb remove-tags-from-resource \
    --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster \
    --tag-keys key1 key3
```

Untuk Windows:

```
aws docdb remove-tags-from-resource ^
    --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster \
    --tag-keys key1 key3
```

Operasi `removed-tags-from-resource` tidak menghasilkan keluaran. Untuk melihat hasil operasi, gunakan operasi `list-tags-for-resource`.

Memelihara Amazon DocumentDB

Amazon DocumentDB secara berkala melakukan dua jenis pemeliharaan:

- Pemeliharaan cluster memperbarui mesin database. Pembaruan mesin membawa perbaikan keamanan, perbaikan bug, fitur baru, dan peningkatan mesin lainnya.
- Pemeliharaan instans memperbarui sistem operasi (OS) pada instance.

Tambalan mesin dan pembaruan OS menggunakan tiga kategori siklus hidup yang sama - opsional, wajib, dan paksa - dengan pemberitahuan yang sama dan perilaku penerapan untuk setiap kategori. Rilis mesin juga memiliki kategori keempat: versi minor, yang Anda tingkatkan secara manual. Kategori-kategorinya adalah:

- Opsional —berisi perbaikan non-kritis. Tidak ada tanggal penerapan otomatis dan tidak ada pemberitahuan AHD; berlaku jika itu cocok untuk Anda. (Untuk pembaruan OS, Anda dapat berlangganan `RDS-EVENT-0230` untuk diberi tahu saat tersedia.)
- Diperlukan —berisi keamanan dan perbaikan penting lainnya. Anda menerima pemberitahuan melalui Dasbor Health (AHD) dan email. Tindakan yang diperlukan berlaku otomatis selama jendela pemeliharaan cluster atau instans Anda setelahnya`AutoAppliedAfterDate`. Anda dapat menunda dengan mengubah jendela pemeliharaan sebelum tanggal tersebut.
- Dipaksakan —perbaikan yang jarang dan sangat kritis. Auto-applies di luar jendela perawatan Anda setelah itu`ForcedApplyDate`. Amazon DocumentDB hanya menunjuk tindakan yang dipaksakan ketika tidak ada opsi lain yang tersedia.

- Versi minor (hanya rilis mesin) —rilis mesin bernomor di atas versi utama (misalnya,`5.0.1`). User-driven: Anda meningkatkan dengan memodifikasi versi mesin cluster. Jangan pernah berlaku secara otomatis; tidak ada pemberitahuan AHD. Versi minor tidak dipublikasikan untuk versi utama lebih awal dari 5.0.

Tambalan mesin dilepaskan ke dalam satu kategori (opsional, wajib, atau dipaksa) dan tetap di sana. Kemajuan pembaruan OS: sebagian besar dimulai sebagai opsional dan, jika tidak diterapkan, transisi ke yang diperlukan dan akhirnya dipaksakan. Waktu yang tepat tergantung pada tambalan dan diterbitkan dalam notifikasi AHD dan bidang tanggal yang dikembalikan oleh `describe-pending-maintenance-actions` (lihatTerapkan tanggal). Catatan Rilis Amazon DocumentDB menggunakan nama kategori ini saat mengumumkan perubahan mesin.

Menerapkan patch mesin apa pun membuat cluster offline sebentar. Sisa topik ini membahas cara kerja jendela pemeliharaan, cara menemukan pekerjaan yang tertunda, cara menerapkan tambalan mesin dan versi minor, cara kerja pembaruan OS, dan penanganan khusus untuk cluster global.

Topik

- Penomoran versi mesin
- Mengelola jendela pemeliharaan Amazon DocumentDB Anda
- Pemberitahuan untuk patch mesin Amazon DocumentDB
- Melihat tindakan pemeliharaan Amazon DocumentDB yang tertunda
- Pembaruan mesin Amazon DocumentDB
- Peningkatan versi minor
- Pembaruan sistem operasi Amazon DocumentDB
- User-initiated pembaruan
- Penambalan cluster global

Penomoran versi mesin

Amazon DocumentDB menggunakan dua pengidentifikasi versi terpisah:

- Versi mesin —nomor tiga bagian dalam bentuk *`major`*`.`*`major`*`.`*`minor`* (misalnya, `5.0.0` atau`5.0.1`). Dua bagian pertama (`5.0`) adalah versi kompatibilitas MongoDB; bagian ketiga adalah versi minor, bertambah ketika Amazon DocumentDB menerbitkan rilis minor yang berisi

perbaikan bug dan perbaikan yang tidak melanggar. Ini adalah versi yang Anda tentukan saat membuat atau memutakhirkan cluster.

- Versi patch engine —nomor tiga bagian terpisah dalam formulir *major*`.0.`*patch* (misalnya,`3.0.17983`) yang mengidentifikasi tingkat patch yang diterapkan ke cluster Anda. Digit tengah selalu`0`. Versi patch berisi perbaikan keamanan dan stabilitas penting.

Anda dapat menentukan versi mesin dari awalan versi patch mesin, seperti yang ditunjukkan pada tabel berikut.

Awalan versi patch mesin	Versi mesin Amazon DocumentDB
`1.0.`*x*	3.6
`2.0.`*x*	4.0
`3.0.`*x*	5.0
`4.0.`*x*	8.0

Untuk memeriksa versi patch cluster Anda berjalan, sambungkan dan jalankan`db.runCommand({getEngineVersion: 1})`.

Untuk daftar versi patch mesin yang dirilis dan isi masing-masing, lihatCatatan rilis.

Mengelola jendela pemeliharaan Amazon DocumentDB Anda

Setiap cluster dan setiap instans memiliki jendela pemeliharaan mingguan 30 menit sendiri—periode ketika modifikasi terjadwal dan patching perangkat lunak dijalankan. Sebagian besar acara selesai dalam 30 menit; yang lebih besar dapat berjalan lebih lama.

Jika Anda tidak memilih jendela saat membuat sumber daya, Amazon DocumentDB menetapkan satu secara acak dalam blok harian 8 jam yang ditentukan untuk Wilayah, pada hari yang dipilih secara acak. Pilih jendela yang meminimalkan dampak pada aplikasi Anda—malam hari atau akhir pekan, misalnya.

Untuk upgrade mesin database, Amazon DocumentDB menggunakan jendela cluster, bukan jendela instance individual.

Tabel berikut menunjukkan blok waktu default per Wilayah.

Nama Wilayah	Region	Blok Waktu UTC
AS Timur (Ohio)	us-east-2	03:00-11:00
US East (Northern Virginia)	us-east-1	03:00-11:00
AS Barat (Oregon)	us-west-2	06:00-14:00
Africa (Cape Town)	af-south-1	03:00 — 11:00
Asia Pasifik (Hong Kong)	ap-east-1	06:00-14:00
Asia Pasifik (Hyderabad)	ap-south-2	06:30 — 14:30
Asia Pasifik (Malaysia)	ap-southeast-5	13:00-21:00
Asia Pasifik (Mumbai)	ap-south-1	06:00-14:00
Asia Pacific (Osaka)	ap-northeast-3	12:00-20:00
Asia Pasifik (Seoul)	ap-northeast-2	13:00-21:00
Asia Pasifik (Singapura)	ap-southeast-1	14:00-22:00
Asia Pasifik (Sydney)	ap-southeast-2	12:00-20:00
Asia Pasifik (Jakarta)	ap-southeast-3	08:00-16:00
Asia Pacific (Melbourne)	ap-southeast-4	11:00-19:00
Asia Pasifik (Thailand)	ap-tenggara 7	15:00-23:00
Asia Pasifik (Tokyo)	ap-northeast-1	13:00-21:00
Kanada (Pusat)	ca-central-1	03:00-11:00
Kanada Barat (Calgary)	ca-west-1	18:00-02:00
Tiongkok (Beijing)	cn-north-1	06:00-14:00
Tiongkok (Ningxia)	cn-northwest-1	06:00-14:00

Nama Wilayah	Region	Blok Waktu UTC
Eropa (Frankfurt)	eu-central-1	21:00-05:00
Europe (Zurich)	eu-central-2	02:00-10:00
Eropa (Irlandia)	eu-west-1	22:00-06:00
Eropa (London)	eu-west-2	22:00-06:00
Europe (Milan)	eu-south-1	02:00-10:00
Eropa (Paris)	eu-west-3	23:59-07:29
Eropa (Spanyol)	eu-south-2	02:00 — 10:00
Eropa (Stockholm)	eu-north-1	04:00 — 12:00
Meksiko (Tengah)	mx-pusat-1	03:00-11:00
Timur Tengah (UAE)	me-central-1	05:00 — 13:00
Amerika Selatan (Sao 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

Mengubah jendela pemeliharaan Amazon DocumentDB Anda

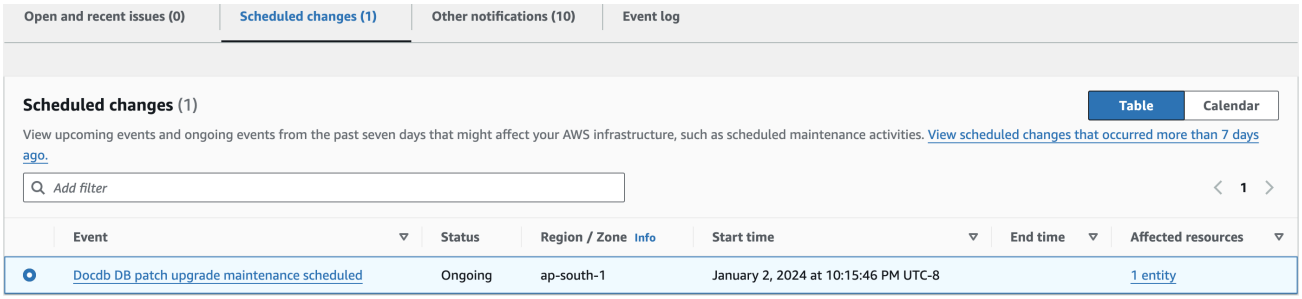
Pilih jendela lalu lintas terendah yang Anda bisa, dan sesuaikan seiring waktu saat pola lalu lintas Anda bergeser. Cluster atau instance tidak tersedia selama jendela hanya jika perubahan sistem—operasi penyimpanan skala atau perubahan kelas instance, misalnya—memerlukan pemadaman, dan hanya selama perubahan itu benar-benar dibutuhkan.

Mengubah waktu pemeliharaan

- Untuk sebuah klaster, lihat Memodifikasi cluster Amazon DocumentDB.
- Untuk sebuah instans, lihat Memodifikasi instance Amazon DocumentDB.

Pemberitahuan untuk patch mesin Amazon DocumentDB

Ketika patch mesin yang diperlukan tersedia di AWS Wilayah, setiap AWS akun dengan klaster Amazon DocumentDB yang terpengaruh di Wilayah tersebut menerima pemberitahuan melalui Dasbor Health (AHD) dan melalui email (dikirim ke AWS alamat pengguna root akun). Satu pemberitahuan dikirimkan per versi mesin Amazon DocumentDB yang terpengaruh. Anda dapat menemukannya di bawah Perubahan terjadwal di AHD. Setiap notifikasi mencantumkan waktu ketersediaan tambalan, jadwal penerapan otomatis, kluster yang terpengaruh, dan catatan rilis.



Pemberitahuan keluar sekitar dua hari sebelum jendela aplikasi otomatis terbuka. Misalnya, patch wajib yang dirilis Senin pukul 00:00 UTC memenuhi syarat untuk mendaftar otomatis pada hari Rabu pukul 00:00 UTC. Jika jendela pemeliharaan klaster Anda adalah Rabu pukul 12:00 UTC, tambalan akan berlaku secara otomatis pada hari Rabu itu—sekitar 12 jam setelah jendela penerapan otomatis terbuka. Jika jendela pemeliharaan Anda adalah Selasa pukul 12:00 UTC, tambalan menunggu seminggu penuh sebelum diterapkan secara otomatis.

Anda memiliki dua opsi setelah menerima pemberitahuan: menerapkan sendiri tambalan sebelum tanggal penerapan otomatis, atau tunggu hingga otomatis diterapkan selama jendela pemeliharaan yang akan datang (default). Untuk mendaftar sendiri, buka tab Maintenance & backup cluster dan cari entri jenisnya. `system-update`

> **Note**
>
> Status notifikasi di AHD tetap berlangsung hingga Amazon DocumentDB merilis patch mesin lain dengan versi patch baru.
> Setelah tambalan diterapkan, versi patch mesin cluster diperbarui
> agar sesuai dengan versi dalam notifikasi. Verifikasi versi baru dengan
> menjalankan`db.runCommand({getEngineVersion: 1})`.

Tambalan opsional dan versi minor baru tidak menghasilkan pemberitahuan AHD atau email. Untuk melacaknya, tonton catatan Rilis Amazon DocumentDB.

Patch paksa (kategori paling langka, dicadangkan untuk perbaikan keamanan paling kritis) juga diumumkan melalui AHD dan email. Tidak seperti tambalan yang diperlukan, tambalan tersebut berlaku di luar jendela pemeliharaan Anda, sehingga contoh pengaturan waktu penerapan otomatis di atas tidak berlaku.

Bereaksi terhadap notifikasi tambalan secara terprogram

AWS Health terintegrasi dengan Amazon EventBridge, yang memungkinkan Anda membangun aplikasi berbasis peristiwa di lebih dari 20 target, termasuk Amazon Simple Queue Service (SQS) AWS Lambda dan Amazon Simple Queue Service (SQS). Untuk bereaksi terhadap ketersediaan tambalan mesin secara terprogram, konfigurasikan terhadap acara tersebut. EventBridge `AWS_DOCDB_DB_PATCH_UPGRADE_MAINTENANCE_SCHEDULED` Dari sana Anda dapat menangkap data acara, meningkatkan acara tambahan, mengirim pemberitahuan push melalui AWS Console Mobile Application, atau mengambil tindakan lain yang Anda butuhkan.

Jika Amazon DocumentDB membatalkan tambalan (jarang), Anda menerima pemberitahuan AHD dan email tentang pembatalan tersebut. Gunakan kode `AWS_DOCDB_DB_PATCH_UPGRADE_MAINTENANCE_CANCELLED` acara dengan Amazon EventBridge untuk menangani kasus ini. Untuk selengkapnya tentang menulis aturan, lihat Panduan EventBridge Pengguna Amazon.

Melihat tindakan pemeliharaan Amazon DocumentDB yang tertunda

Gunakan Konsol Manajemen AWS or AWS CLI untuk memeriksa pemeliharaan apa yang tertunda untuk cluster atau instance.

Pembaruan yang tertunda muncul dengan tipe tindakan`system-update`, yang mencakup tambalan mesin dan pembaruan OS.

Saat pembaruan tertunda, Anda dapat:

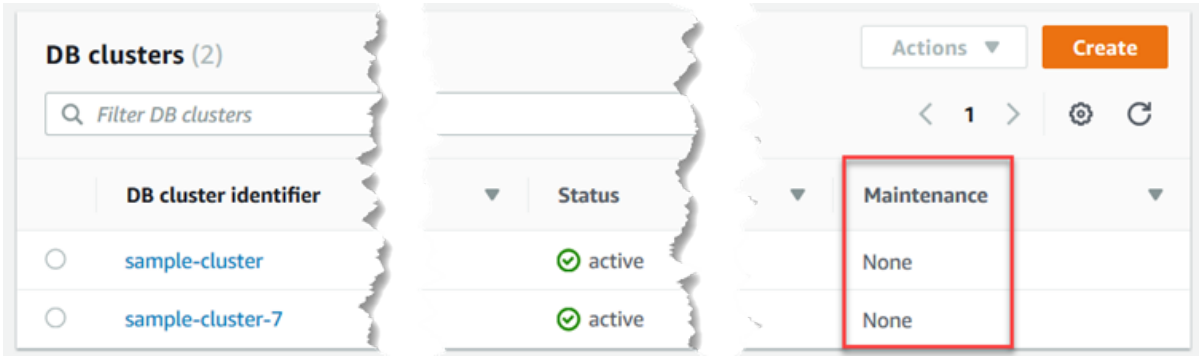
- Terapkan segera.
- Jadwalkan untuk jendela pemeliharaan berikutnya.
- Tunda (patch mesin dan pembaruan OS saja) dengan mengubah jendela perawatan Anda sebelumnya. `AutoAppliedAfterDate` Setelah tanggal tersebut berlalu, tindakan akan diterapkan secara otomatis selama jendela pemeliharaan berikutnya. Setelah `ForcedApplyDate` berlalu, tidak ada penangguhan lebih lanjut yang mungkin dilakukan.

> **Note**
>
> Jika Anda tidak mengambil tindakan, tindakan pemeliharaan yang diperlukan seperti patch engine yang diperlukan berlaku otomatis selama jendela pemeliharaan yang akan datang. Tambalan opsional dan versi minor tidak pernah diterapkan secara otomatis.

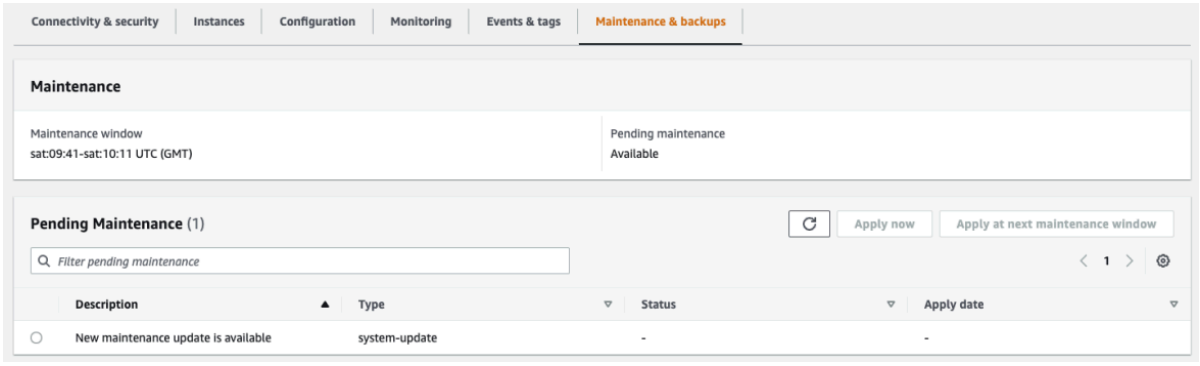
Jendela pemeliharaan mengontrol saat operasi yang tertunda dimulai, bukan berapa lama waktu yang dibutuhkan untuk menyelesaikannya.

Using the Konsol Manajemen AWS

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Pada panel navigasi, silakan pilih Klaster.
3. Kolom Pemeliharaan klaster menunjukkan Jendela Tersedia, Diperlukan, atau Berikutnya saat pembaruan tertunda.



4. Buka cluster, lalu pilih Maintenance & backup untuk melihat item Pending Maintenance dan menindaklanjutinya.



Using the AWS CLI

Jalankan `describe-pending-maintenance-actions` untuk melihat apa yang tertunda. Contoh berikut menunjukkan akun tanpa tindakan yang tertunda.

```
aws docdb describe-pending-maintenance-actions
```

Keluaran dari operasi ini terlihat seperti berikut ini (format JSON).

```
{
    "PendingMaintenanceActions": []
}
```

Akun dengan tindakan tertunda mengembalikan output yang terlihat seperti ini:

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

Anda dapat membuat cakupan daftar ke cluster tertentu dengan`--filters`, dalam formulirName=*`filter-name`*,Values=*`resource-id`*,.... Filter yang diterima Name adalah`db-cluster-id`, yang mengambil daftar pengidentifikasi cluster atau ARN.

Example

Untuk Linux, macOS, atau Unix:

```
aws docdb describe-pending-maintenance-actions \
```

```
    --filters Name=db-cluster-id,Values=sample-cluster1,sample-cluster2
```

Untuk Windows:

```
aws docdb describe-pending-maintenance-actions ^
    --filters Name=db-cluster-id,Values=sample-cluster1,sample-cluster2
```

Terapkan tanggal

Setiap tindakan pemeliharaan yang tertunda membawa hingga tiga tanggal penerapan. Mereka muncul di AWS CLI output untuk `describe-pending-maintenance-actions` dan menunjukkan kapan tindakan akan berjalan. Bidang adalah `null` untuk pemeliharaan opsional.

- **`CurrentApplyDate`**—ketika tindakan dijadwalkan untuk dijalankan, baik sekarang atau di jendela pemeliharaan berikutnya. Dihuni untuk tindakan yang diperlukan dan dipaksakan.
- **`AutoAppliedAfterDate`**—tanggal setelah penerapan otomatis dimulai selama jendela pemeliharaan cluster atau instance. Diisi untuk tindakan yang diperlukan.
- **`ForcedApplyDate`**—tenggat waktu yang sulit. Setelah tanggal ini tindakan berjalan secara otomatis, terlepas dari jendela pemeliharaan Anda. Dihuni karena tindakan paksa.

Untuk menunda tindakan yang tertunda, pindahkan jendela pemeliharaan Anda ke hari berikutnya sebelumnya`AutoAppliedAfterDate`. Setelah `AutoAppliedAfterDate` berlalu, tindakan akan diterapkan secara otomatis selama jendela pemeliharaan berikutnya. Setelah `ForcedApplyDate` berlalu, tidak ada penangguhan lebih lanjut yang mungkin dilakukan. Jendela penangguhan yang tepat bervariasi per tambalan; tanggal diterbitkan dalam pemberitahuan AHD dan output. AWS CLI

Pembaruan mesin Amazon DocumentDB

Ketika Anda telah mengidentifikasi patch mesin yang tertunda, gunakan salah satu prosedur berikut untuk menerapkan atau menjadwalkannya. Anda dapat menjalankan prosedur ini baik dari Konsol Manajemen AWS atau AWS CLI.

Using the Konsol Manajemen AWS

Untuk mengelola pembaruan untuk klaster

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb

2. Pada panel navigasi, silakan pilih Klaster.
3. Pilih cluster yang ingin Anda perbarui.
4. Dari menu Tindakan, pilih salah satu:

   - Tingkatkan sekarang —jalankan pemeliharaan yang tertunda segera.
   - Tingkatkan di jendela berikutnya —jalankan selama jendela pemeliharaan cluster berikutnya.

   Anda juga dapat menggunakan Apply now atau Apply di jendela pemeliharaan berikutnya dari bagian Pending Maintenance pada tab Maintenance & backup cluster (lihatMelihat tindakan pemeliharaan Amazon DocumentDB yang tertunda).

   > **Note**
   >
   > Jika tidak ada yang tertunda, semua opsi ini tidak aktif.

Using the AWS CLI

Terapkan pembaruan yang tertunda denganapply-pending-maintenance-action.

Parameter

- **`--resource-identifier`**—Amazon DocumentDB Amazon Resource Name (ARN) dari sumber daya target tindakan yang tertunda.
- **`--apply-action`**—tindakan pemeliharaan yang tertunda untuk diterapkan. Nilai-nilai yang valid: `system-update`, `db-upgrade`.
- **`--opt-in-type`**—jenis permintaan keikutsertaan, atau apakah akan membatalkannya. Nilai valid:
  - `immediate`-Terapkan sekarang. Tidak dapat dibatalkan setelah dikirimkan.
  - `next-maintenance`—terapkan selama jendela pemeliharaan sumber daya berikutnya.
  - `undo-opt-in`—membatalkan keikutsertaan yang ada`next-maintenance`.

Example

Untuk Linux, macOS, atau 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
```

Untuk 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
```

Baca ketersediaan selama penambalan

Amazon DocumentDB engine 5.0 dan 8.0 mempertahankan ketersediaan baca selama penambalan saat cluster memiliki beberapa instance. Amazon DocumentDB menambal instance pembaca secara bergulir, dalam tiga kelompok, sehingga pembaca yang tersisa terus melayani lalu lintas. Penulis sebentar tidak tersedia saat menambal. Untuk mencapai nol downtime baca, atur preferensi baca Anda sehingga bacaan dapat kembali ke penulis: `secondaryPreferred` atau `primaryPreferred` bekerja; `primary` atau `secondary` sendirian dapat menimbulkan downtime baca.

Baca mode preferensi	Selama upgrade penulis	Selama pemutakhiran pembaca	Jumlah minimum pembaca yang dibutuhkan untuk nol waktu henti baca
`primary`	Read/write downtime	Tidak ada dampak	N/A
`primaryPreferred`	Tulis downtime	Tidak ada dampak	1
`secondary`	Tulis downtime	Baca downtime (jika hanya satu pembaca)	2
`secondaryPreferred`	Tulis downtime	Tidak ada dampak	1

Baca mode preferensi	Selama upgrade penulis	Selama pemutakhiran pembaca	Jumlah minimum pembaca yang dibutuhkan untuk nol waktu henti baca
`nearest`	Tulis downtime	Tidak ada dampak	1

Saat pembaca menambal, keseluruhan throughput baca cluster turun sementara. Untuk menjaga throughput tetap stabil, berikan pembaca tambahan sebelum peningkatan dan hapus setelah selesai.

Pada engine 3.6 dan 4.0, fitur ketersediaan baca ini tidak berlaku: patch mesin menyebabkan waktu henti yang lebih lama yang memengaruhi pembacaan dan penulisan. Untuk meningkatkan ke versi utama yang melakukannya, lihatPeningkatan versi utama Amazon DocumentDB di tempat.

Panjang waktu henti tambalan

Engine-patch downtime bervariasi. Faktor terbesar adalah penggunaan CPU dan tekanan memori pada instance pada saat patch, jadi ukuran yang tepat instance Anda penting. Untuk meminimalkan waktu henti, jalankan versi engine utama Amazon DocumentDB terbaru dan sebarkan instans di beberapa Availability Zone.

Pembaruan dan penggantian tambalan

Amazon DocumentDB memonitor tambalan setelah rilis. Dalam kasus yang jarang terjadi ketika masalah diidentifikasi, Amazon DocumentDB menghentikan sementara peluncuran saat menyiapkan versi yang diperbarui. Ketika ini terjadi, cluster yang belum menerima tambalan tidak lagi melihatnya sebagai tindakan pemeliharaan yang tersedia, dan pemberitahuan perubahan terjadwal yang sesuai di ditarik. Dasbor Health Cluster yang sudah menjalankan versi yang terpengaruh terus beroperasi secara normal dan tidak memerlukan tindakan dari Anda.

Patch yang diperbarui segera menyusul. Ketika tersedia di Wilayah Anda, Anda menerima pemberitahuan baru melalui Dasbor Health dan email, seperti yang dijelaskan dalamPemberitahuan untuk patch mesin Amazon DocumentDB.

Peningkatan versi minor

Amazon DocumentDB menerbitkan versi minor di atas versi utama 5.0 dan yang lebih baru (misalnya,). `5.0.1` Versi minor tidak dipublikasikan untuk versi utama lebih awal dari 5.0. Versi minor berperilaku berbeda dari patch engine yang diperlukan dan opsional:

- Mereka tidak muncul sebagai tindakan pemeliharaan yang tertunda dan tidak pernah diterapkan secara otomatis.
- Mereka tidak menghasilkan pemberitahuan AHD atau email. Versi minor baru diumumkan dalam catatan Rilis Amazon DocumentDB.
- Untuk meningkatkan, Anda memodifikasi versi mesin cluster (segera atau selama jendela pemeliharaan berikutnya). Peningkatan versi minor memerlukan waktu henti singkat dan hanya satu arah — Anda tidak dapat menurunkan versi ke versi minor sebelumnya. Untuk cluster global, tingkatkan cluster sekunder sebelum primer.

Baca lebih lanjut:Peningkatan versi minor Amazon DocumentDB.

Pembaruan sistem operasi Amazon DocumentDB

Instans terkadang membutuhkan pembaruan OS. Amazon DocumentDB memperbarui OS untuk meningkatkan kinerja dan memperketat keamanan. Pembaruan OS membuat versi mesin cluster dan kelas instance tidak berubah. Seperti patch mesin, pembaruan OS menggunakan siklus hidup opsional/wajib/paksa yang dijelaskan di bagian atas topik ini; tidak seperti tambalan mesin, pembaruan OS dapat bertransisi melalui kategori ini dari waktu ke waktu jika Anda menundanya. Terapkan pembaruan OS segera setelah tersedia, dan atur jendela pemeliharaan instans Anda ke waktu yang sesuai dengan bisnis Anda.

Untuk mendapatkan acara saat pembaruan opsional baru tiba, berlangganan `RDS-EVENT-0230` dalam kategori acara penambalan keamanan. Untuk detailnya, lihat Berlangganan Langganan Acara Amazon DocumentDB. Setelah menerima pemberitahuan, Anda dapat menerapkan sendiri patch OS ke setiap instance.

Saat menambal cluster, perbarui instance pembaca terlebih dahulu dan penulis terakhir. Hindari menambal pembaca dan penulis secara bersamaan — failover selama patch dapat memperpanjang waktu henti. Pemeliharaan pada instance utama memicu failover, jadi jalankan lebih dari satu instance per cluster agar tetap tersedia. Lihat perinciannya di Failover Amazon DocumentDB.

> **Important**
>
> Instans Amazon DocumentDB Anda offline untuk upgrade OS. Multi-instance cluster meminimalkan dampak. Jika Anda menjalankan cluster instance tunggal, Anda dapat menambahkan sementara sekunder untuk pemutakhiran dan menghapusnya setelahnya. Sekunder menimbulkan biaya biasa saat ada.

> **Note**
>
> Tetap mengikuti pembaruan opsional dan yang diperlukan mungkin diperlukan untuk kepatuhan. Terapkan pembaruan yang tersedia secara rutin selama jendela pemeliharaan Anda.

Pembaruan OS terkait dengan versi mesin dan kelas instans tertentu, sehingga instance yang berbeda menjadi memenuhi syarat pada waktu yang berbeda. Ketika sebuah instance memenuhi syarat, pembaruan muncul di konsol; Anda juga dapat melihatnya melalui AWS CLI `describe-pending-maintenance-actions` perintah atau `DescribePendingMaintenanceActions` API.

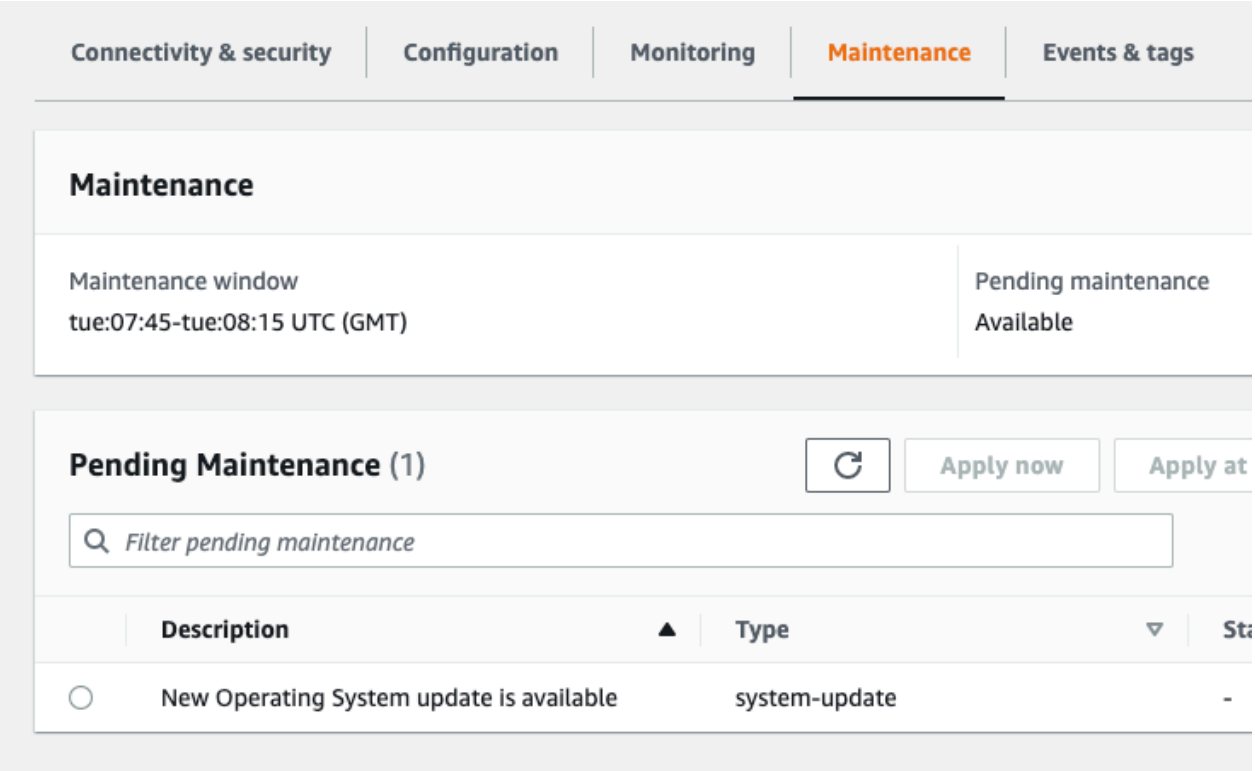
> **Note**
>
> Jika klaster Anda tidak pada rilis patch terbaru dari mesin Amazon DocumentDB-nya, pembaruan OS mungkin tidak muncul sebagai tersedia. Terapkan patch mesin terbaru terlebih dahulu, lalu periksa lagi.

Gunakan Konsol Manajemen AWS atau AWS CLI untuk memeriksa apakah pembaruan tersedia.

Using the Konsol Manajemen AWS

Untuk memeriksa pembaruan OS dari konsol:

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Pada panel navigasi, silakan pilih Klaster. Daftar menunjukkan kedua cluster dan instance di dalamnya, dibedakan oleh kolom Peran.
3. Pilih baris yang Perannya adalah Instance (bukan baris cluster). Pembaruan OS berlaku untuk instance, bukan cluster.
4. Pilih Pemeliharaan.
5. Lihat di bawah Pending Maintenance untuk pembaruan OS.



Dari bagian Pending Maintenance, pilih pembaruan OS dan pilih Terapkan sekarang atau Terapkan di jendela pemeliharaan berikutnya. Jika nilai pemeliharaan adalah jendela berikutnya, Anda dapat menunda pembaruan dengan peningkatan Defer selama belum dimulai.

Anda juga dapat melakukan ini dari daftar klaster: di panel navigasi, pilih Cluster, pilih baris yang Perannya adalah Instance, dan pilih Terapkan sekarang atau Terapkan di jendela pemeliharaan berikutnya dari menu Tindakan.

Using the AWS CLI

Dari AWS CLI, jalankan`describe-pending-maintenance-actions`:

```
aws docdb describe-pending-maintenance-actions
```

```
{
  "PendingMaintenanceActions": [
    {
      "ResourceIdentifier": "arn:aws:rds:us-east-1:123456789012:db:sample-cluster-
instance-1",
      "PendingMaintenanceActionDetails": [
        {
          "Action": "system-update",
```

```
            "Description": "New Operating System update is available"
          }
        ]
      }
    ]
}
```

User-initiated pembaruan

Beberapa perubahan yang Anda mulai sendiri—misalnya, menukar kelas instance dengan kelas dengan memori lebih atau kurang, atau mengubah grup parameter cluster. Amazon DocumentDB memperlakukan ini secara berbeda dari pembaruan yang dimulainya. Untuk detailnya, lihat:

- Memodifikasi cluster Amazon DocumentDB
- Memodifikasi instance Amazon DocumentDB

Untuk membuat daftar perubahan yang dimulai pengguna yang masih tertunda:

Example

Untuk mencantumkan perubahan yang diprakarsai pengguna yang tertunda untuk instans Anda

Untuk Linux, macOS, atau Unix:

```
aws docdb describe-db-instances \
    --query 'DBInstances[*].
[DBClusterIdentifier,DBInstanceIdentifier,PendingModifiedValues]'
```

Untuk Windows:

```
aws docdb describe-db-instances ^
    --query 'DBInstances[*].
[DBClusterIdentifier,DBInstanceIdentifier,PendingModifiedValues]'
```

Keluaran dari operasi ini terlihat seperti berikut ini (format JSON).

Dalam contoh ini, `sample-cluster-instance` memiliki perubahan yang tertunda ke`db.r5.xlarge`; tidak `sample-cluster-instance-2` memiliki.

```
[
```

```
    [
        "sample-cluster",
        "sample-cluster-instance",
        {
            "DBInstanceClass": "db.r5.xlarge"
        }
    ],
    [
        "sample-cluster",
        "sample-cluster-instance-2",
        {}
    ]
]
```

Penambalan cluster global

Dalam klaster global, setiap kelompok anggota — primer dan sekunder — melakukan upgrade selama jendela pemeliharaannya sendiri. Ketika patch mesin yang diperlukan tersedia di setiap Wilayah, Anda menerima pemberitahuan AHD dan email. Tambalan opsional dan versi minor baru tidak menghasilkan notifikasi; periksa catatan Rilis Amazon DocumentDB untuk itu.

Jika Anda melamar sendiri, selalu tambal sekunder terlebih dahulu dan yang utama terakhir. Urutan ini membuat failover dan switchover tetap tersedia selama peluncuran.

> **Important**
>
> Jika Anda menambal primer terlebih dahulu secara tidak sengaja, bawa semua sekunder ke versi yang sama sesegera mungkin. Failover dan switchover tetap dinonaktifkan hingga setiap cluster menggunakan versi yang sama.

Jika Anda tidak mengambil tindakan, tambalan otomatis berlaku selama jendela pemeliharaan berikutnya setiap cluster: sekunder terlebih dahulu, lalu primer di jendelanya setelah sekunder selesai.

Simpan cluster DB primer dan sekunder pada versi yang sama. Failover lintas wilayah yang dikelola hanya berfungsi pada database global ketika setiap cluster berbagi versi mesin dan level patch yang sama. Hal yang sama berlaku jika Anda menambahkan sekunder baru yang menggunakan versi mesin yang lebih baru daripada yang primer—buat sekunder baru pada versi primer sebelum bergabung dengan mereka ke database global.

Setelah pemberitahuan tambalan, tingkatkan versi primer dan sekunder ke versi terbaru secepatnya agar failover dan switchover tetap berfungsi. Jika permintaan failover atau switchover ditolak, bandingkan versi patch engine di seluruh cluster; jika tidak cocok, terapkan patch yang tersedia pada cluster yang tertinggal.

Memahami peran terkait layanan

Amazon DocumentDB (dengan kompatibilitas MongoDB) menggunakan peran terkait layanan (IAM). AWS Identity and Access Management Peran terkait layanan adalah jenis peran IAM unik yang ditautkan langsung ke Amazon DocumentDB. Service-linked peran telah ditentukan sebelumnya oleh Amazon DocumentDB dan mencakup semua izin yang diperlukan layanan untuk AWS memanggil layanan lain atas nama Anda.

Peran tertaut layanan membuat penggunaan Amazon DocumentDB lebih mudah karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. Amazon DocumentDB menentukan izin peran tertaut layanannya, dan kecuali ditentukan lain, hanya Amazon DocumentDB yang dapat menjalankan perannya. Izin yang ditentukan mencakup kebijakan kepercayaan dan kebijakan izin, serta bahwa kebijakan izin tidak dapat dilampirkan ke entitas IAM lainnya.

Anda dapat menghapus peran hanya setelah terlebih dahulu menghapus sumber daya terkaitnya. Ini melindungi sumber daya Amazon DocumentDB karena Anda tidak dapat secara tidak sengaja menghapus izin untuk mengakses sumber daya.

Untuk informasi tentang layanan lain yang mendukung peran terkait layanan, lihat AWS Layanan yang Bekerja dengan IAM dan cari layanan yang memiliki Ya di kolom Service-Linked Peran. Pilih Ya dengan sebuah tautan untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Izin peran terkait layanan Amazon DocumentDB

Amazon DocumentDB (dengan kompatibilitas MongoDB) menggunakan peran terkait layanan bernama AWSServiceRoleForRDSuntuk memungkinkan Amazon DocumentDB memanggil layanan atas nama kluster Anda. AWS

Peran AWSServiceRoleForRDS terkait layanan mempercayai layanan berikut untuk mengambil peran:

- `docdb.amazonaws.com`

Kebijakan izin peran mengizinkan Amazon DocumentDB untuk menyelesaikan tindakan berikut ini pada sumber daya yang ditentukan:

- Tindakan pada `ec2`:
  - `AssignPrivateIpAddresses`
  - `AuthorizeSecurityGroupIngress`
  - `CreateNetworkInterface`
  - `CreateSecurityGroup`
  - `DeleteNetworkInterface`
  - `DeleteSecurityGroup`
  - `DescribeAvailabilityZones`
  - `DescribeInternetGateways`
  - `DescribeSecurityGroups`
  - `DescribeSubnets`
  - `DescribeVpcAttribute`
  - `DescribeVpcs`
  - `ModifyNetworkInterfaceAttribute`
  - `RevokeSecurityGroupIngress`
  - `UnassignPrivateIpAddresses`
- Tindakan pada `sns`:
  - `ListTopic`
  - `Publish`
- Tindakan pada `cloudwatch`:
  - `PutMetricData`
  - `GetMetricData`
  - `CreateLogStream`
  - `PullLogEvents`
  - `DescribeLogStreams`
  - `CreateLogGroup`

> **Note**
>
> Anda harus mengonfigurasi izin agar entitas IAM (seperti pengguna, grup, atau peran) dapat membuat, mengedit, atau menghapus peran terkait layanan. Anda mungkin menemukan pesan kesalahan berikut ini:
> Tidak dapat membuat sumber daya. Verifikasi bahwa Anda memiliki izin untuk membuat peran terkait layanan. Jika tidak, tunggu dan coba lagi nanti.
> Jika Anda melihat kesalahan ini, pastikan Anda mengaktifkan izin berikut ini:
>
> ```
> {
>     "Action": "iam:CreateServiceLinkedRole",
>     "Effect": "Allow",
>     "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
> AWSServiceRoleForRDS",
>     "Condition": {
>         "StringLike": {
>             "iam:AWSServiceName":"rds.amazonaws.com"
>         }
>     }
> }
> ```

Untuk informasi selengkapnya, lihat Izin Service-Linked Peran di Panduan Pengguna IAM.

Membuat peran terkait layanan Amazon DocumentDB

Anda tidak perlu membuat peran terkait layanan secara manual. Saat Anda membuat klaster, Amazon DocumentDB membuat peran tertaut layanan untuk Anda.

Jika Anda menghapus peran tertaut layanan ini dan ingin membuatnya lagi, Anda dapat mengulangi proses yang sama untuk membuat kembali peran tersebut di akun Anda. Saat Anda membuat klaster, Amazon DocumentDB membuat peran tertaut layanan untuk Anda.

Memodifikasi peran terkait layanan Amazon DocumentDB

Amazon DocumentDB tidak mengizinkan Anda memodifikasi AWSServiceRoleForRDS peran terkait layanan. Setelah membuat peran terkait layanan, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin merujuk peran tersebut. Namun demikian, Anda dapat memodifikasi penjelasan peran menggunakan IAM. Untuk informasi selengkapnya, lihat Mengedit Service-Linked Peran di Panduan Pengguna IAM.

Menghapus peran terkait layanan Amazon DocumentDB

Jika Anda tidak lagi perlu menggunakan fitur atau layanan yang memerlukan peran tertaut layanan, sebaiknya hapus peran tersebut. Dengan begitu, Anda tidak perlu lagi memantau atau memelihara entitas yang tidak digunakan. Namun demikian, Anda harus menghapus semua klaster Anda sebelum Anda dapat menghapus peran tertaut layanan.

Membersihkan peran terkait layanan Amazon DocumentDB

Sebelum dapat menggunakan IAM untuk menghapus peran tertaut layanan, Anda harus mengonfirmasi terlebih dahulu bahwa peran tersebut tidak memiliki sesi aktif dan menghapus sumber daya yang digunakan oleh peran tersebut.

Untuk memeriksa apakah peran tertaut layanan memiliki sesi aktif menggunakan konsol

1. Masuk ke Konsol Manajemen AWS dan buka konsol IAM dihttps://console.aws.amazon.com/iam/.
2. Di panel navigasi konsol IAM, pilih Peran, lalu pilih nama (bukan kotak centang) peran. AWSServiceRoleForRDS
3. Di halaman Ringkasan untuk peran yang dipilih, pilih tab Penasihat Akses.
4. Di tab Penasihat Akses, tinjau aktivitas terbaru untuk peran tertaut layanan.

> **Note**
>
> Jika Anda tidak yakin apakah Amazon DocumentDB AWSServiceRoleForRDS menggunakan peran tersebut, Anda dapat mencoba menghapus peran tersebut. Jika layanan ini menggunakan peran tersebut, peran tidak dapat dihapus dan Anda dapat melihat Wilayah tempat peran tersebut digunakan. Jika peran tersebut sedang digunakan, Anda harus menunggu hingga sesi ini berakhir sebelum dapat menghapus peran tersebut. Anda tidak dapat mencabut sesi untuk peran terkait layanan.

Jika Anda ingin menghapus AWSServiceRoleForRDS peran, Anda harus terlebih dahulu menghapus semua instance dan cluster Anda. Untuk informasi tentang menghapus instans dan klaster, lihat topik berikut ini:

- Menghapus instans Amazon DocumentDB
- Menghapus cluster Amazon DocumentDB

Wilayah yang didukung untuk peran terkait layanan Amazon DocumentDB

Amazon DocumentDB mendukung menggunakan peran tertaut layanan di semua Wilayah tempat layanan tersedia. Lihat informasi yang lebih lengkap di https://docs.aws.amazon.com/documentdb/latest/devguide/regions-and-azs.html#regions-and-azs-availability.

Menggunakan aliran perubahan dengan Amazon DocumentDB

Fitur aliran perubahan di Amazon DocumentDB (dengan kompatibilitas MongoDB) menyediakan urutan waktu peristiwa perubahan yang terjadi dalam koleksi klaster Anda. Anda dapat membaca peristiwa dari aliran perubahan untuk menerapkan banyak kasus penggunaan yang berbeda, termasuk yang berikut:

- Notifikasi perubahan
- Full-text pencarian dengan Amazon OpenSearch Service (OpenSearch Layanan)
- Analitik dengan Amazon Redshift

Aplikasi dapat menggunakan aliran perubahan untuk berlangganan perubahan data pada koleksi individu. Peristiwa aliran perubahan diurutkan saat terjadi di klaster dan disimpan selama 3 jam (secara default) setelah peristiwa dicatat. Periode retensi dapat diperpanjang hingga 7 hari menggunakan parameter `change_stream_log_retention_duration`. Untuk mengubah periode retensi aliran perubahan, lihat Memodifikasi Durasi Retensi Log Aliran Ubah.

Topik

- Operasi yang didukung
- Penagihan
- Batasan
- Mengaktifkan aliran perubahan
- Contoh: menggunakan aliran perubahan dengan Python
- Pencarian dokumen lengkap
- Melanjutkan aliran perubahan
- Melanjutkan aliran perubahan dengan start AtOperationTime
- Melanjutkan aliran perubahan dengan posting BatchResumeToken

- Transaksi di aliran perubahan
- Memodifikasi durasi retensi log aliran perubahan
- Menggunakan aliran perubahan pada instance sekunder

Operasi yang didukung

Amazon DocumentDB mendukung operasi berikut untuk aliran perubahan:

- Semua peristiwa perubahan didukung di API `db.collection.watch()`, `db.watch()` dan `client.watch()` MongoDB.
- Pencarian dokumen lengkap untuk pembaruan.
- Tahap agregasi: `$match`, `$project`, `$redact`, serta `$addFields` dan `$replaceRoot`.
- Melanjutkan aliran perubahan dari token resume
- Melanjutkan aliran perubahan dari stempel waktu menggunakan `startAtOperation` (berlaku untuk Amazon DocumentDB 4.0+)

Penagihan

Fitur aliran perubahan Amazon DocumentDB dinonaktifkan secara default dan tidak dikenakan biaya tambahan apa pun hingga fitur tersebut diaktifkan. Menggunakan aliran perubahan dalam klaster menimbulkan tambahan IO baca dan tulis serta tambahan biaya penyimpanan. Anda dapat menggunakan operasi API `modifyChangeStreams` untuk mengaktifkan fitur ini untuk klaster Anda. Untuk informasi selengkapnya tentang penentuan harga, lihat Penentuan harga Amazon DocumentDB.

Batasan

Aliran perubahan memiliki batasan berikut di Amazon DocumentDB:

- Di Amazon DocumentDB 3.6. dan Amazon DocumentDB 4.0, aliran perubahan hanya dapat dibuka dari koneksi ke instance utama cluster Amazon DocumentDB. Membaca dari aliran perubahan pada instance replika tidak didukung di Amazon DocumentDB 3.6. dan Amazon DocumentDB 4.0. Ketika memanggil operasi API `watch()`, Anda harus menentukan preferensi baca `primary` untuk memastikan bahwa semua pembacaan diarahkan ke instans primer (lihat bagian Contoh).
- Di Amazon DocumentDB 5.0, aliran perubahan dapat dibuka dari instans primer dan instance sekunder, termasuk klaster global. Anda dapat menentukan preferensi baca sekunder untuk

mengarahkan aliran perubahan ke instance sekunder. Lihat Menggunakan aliran perubahan pada instance sekunder untuk praktik dan batasan terbaik tambahan.

- Peristiwa yang ditulis ke aliran perubahan untuk koleksi tersedia hingga 7 hari (defaultnya adalah 3 jam). Data aliran perubahan dihapus setelah jendela durasi retensi log, meskipun tidak ada perubahan baru yang terjadi.
- Operasi tulis yang berjalan lama pada koleksi seperti `updateMany` atau `deleteMany` dapat menghentikan sementara penulisan peristiwa aliran perubahan hingga operasi tulis yang berjalan lama selesai.
- Amazon DocumentDB tidak mendukung log operasi MongoDB (`oplog`).
- Dengan Amazon DocumentDB, Anda harus secara eksplisit mengaktifkan aliran perubahan pada koleksi tertentu.
- Jika ukuran total peristiwa aliran perubahan (termasuk data perubahan dan dokumen lengkap, jika diminta) lebih besar dari `16 MB`, klien akan mengalami kegagalan baca pada aliran perubahan.
- Driver Ruby saat ini tidak didukung saat menggunakan `db.watch()` dan `client.watch()` dengan Amazon DocumentDB 3.6.
- Output dari `updateDescription` perintah dalam aliran perubahan berbeda di Amazon DocumentDB daripada di MongoDB ketika nilai bidang yang diperbarui sama dengan yang sebelumnya:
  - Amazon DocumentDB tidak mengembalikan bidang dalam `updateDescription` output jika bidang yang disediakan ditentukan dalam `$set` perintah dan nilai targetnya sudah sama dengan nilai sumber.
  - MongoDB mengembalikan bidang dalam output, bahkan jika nilai yang ditentukan sama dengan nilai saat ini.

Mengaktifkan aliran perubahan

Anda dapat mengaktifkan aliran perubahan Amazon DocumentDB untuk semua koleksi dalam basis data tertentu, atau hanya untuk koleksi yang dipilih. Berikut ini adalah contoh cara mengaktifkan aliran perubahan untuk kasus penggunaan yang berbeda menggunakan shell mongo. String kosong diperlakukan sebagai wildcard ketika menentukan nama basis data dan koleksi.

```
//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});
```

Aliran perubahan akan diaktifkan untuk koleksi jika apa pun dari berikut ini benar:

- Baik basis data dan koleksi diaktifkan secara eksplisit.
- Basis data yang berisi koleksi diaktifkan.
- Semua basis data diaktifkan.

Menjatuhkan koleksi dari basis data tidak menonaktifkan aliran perubahan untuk koleksi tersebut jika basis data induk juga mengaktifkan aliran perubahan, atau jika semua basis data di klaster diaktifkan. Jika koleksi baru dibuat dengan nama yang sama dengan koleksi yang dihapus, aliran perubahan akan diaktifkan untuk koleksi tersebut.

Anda dapat membuat daftar semua aliran perubahan yang diaktifkan klaster Anda dengan menggunakan tahap alur agregasi `$listChangeStreams`. Semua tahapan agregasi yang didukung oleh Amazon DocumentDB dapat digunakan dalam alur untuk pemrosesan tambahan. Jika koleksi yang sebelumnya diaktifkan telah dinonaktifkan, koleksi tersebut tidak akan muncul di output `$listChangeStreams`.

```
//List all databases and collections with change streams enabled
```

```
cursor = new DBCommandCursor(db,
    db.runCommand(
        {aggregate: 1,
        pipeline: [{$listChangeStreams: 1}],
        cursor:{}}));
```

```
//List of all databases and collections with change streams enabled
{ "database" : "test", "collection" : "foo" }
{ "database" : "bar", "collection" : "" }
{ "database" : "", "collection" : "" }
```

```
//Determine if the database “bar” or collection “bar.foo” have change streams enabled
cursor = new DBCommandCursor(db,
  db.runCommand(
      {aggregate: 1,
       pipeline: [{$listChangeStreams: 1},
                  {$match: {$or: [{database: "bar", collection: "foo"},
                                  {database: "bar", collection: ""},
                                  {database: "", collection: ""}]}}
                 ],
      cursor:{}}));
```

Contoh: menggunakan aliran perubahan dengan Python

Berikut ini adalah contoh penggunaan aliran perubahan Amazon DocumentDB dengan Python di tingkat koleksi.

```
import os
import sys
from pymongo import MongoClient, ReadPreference

username = "DocumentDBusername"
password = <Insert your password>

clusterendpoint = "DocumentDBClusterEndpoint”
client = MongoClient(clusterendpoint, username=username, password=password, tls='true',
 tlsCAFile='global-bundle.pem')

db = client['bar']

#While ‘Primary’ is the default read preference, here we give an example of
#how to specify the required read preference when reading the change streams
```

```
coll = db.get_collection('foo', read_preference=ReadPreference.PRIMARY)
#Create a stream object
stream = coll.watch()
#Write a new document to the collection to generate a change event
coll.insert_one({'x': 1})
#Read the next change event from the stream (if any)
print(stream.try_next())

"""
Expected Output:
{'_id': {'_data': '015daf94f600000002010000000200009025'},
'clusterTime': Timestamp(1571788022, 2),
'documentKey': {'_id': ObjectId('5daf94f6ea258751778163d6')},
'fullDocument': {'_id': ObjectId('5daf94f6ea258751778163d6'), 'x': 1},
'ns': {'coll': 'foo', 'db': 'bar'},
'operationType': 'insert'}
"""

#A subsequent attempt to read the next change event returns nothing, as there are no
 new changes
print(stream.try_next())

"""
Expected Output:
None
"""

#Generate a new change event by updating a document
result = coll.update_one({'x': 1}, {'$set': {'x': 2}})
print(stream.try_next())

"""
Expected Output:
{'_id': {'_data': '015daf99d400000001010000000100009025'},
'clusterTime': Timestamp(1571789268, 1),
'documentKey': {'_id': ObjectId('5daf9502ea258751778163d7')},
'ns': {'coll': 'foo', 'db': 'bar'},
'operationType': 'update',
'updateDescription': {'removedFields': [], 'updatedFields': {'x': 2}}}
"""
```

Berikut ini adalah contoh penggunaan aliran perubahan Amazon DocumentDB dengan Python di tingkat basis data.

```
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': '015daf94f60000000201000000200009025'},
'clusterTime': Timestamp(1571788022, 2),
'documentKey': {'_id': ObjectId('5daf94f6ea258751778163d6')},
'fullDocument': {'_id': ObjectId('5daf94f6ea258751778163d6'), 'x': 1},
'ns': {'coll': 'foo1', 'db': 'bar'},
'operationType': 'insert'}
"""
```

Pencarian dokumen lengkap

Peristiwa perubahan pembaruan tidak menyertakan dokumen lengkap; itu mencakup hanya perubahan yang telah dibuat. Jika kasus penggunaan Anda memerlukan dokumen lengkap yang terpengaruh oleh pembaruan, Anda dapat mengaktifkan pencarian dokumen lengkap saat membuka aliran.

Dokumen `fullDocument` untuk peristiwa aliran perubahan pembaruan mewakili versi terbaru dari dokumen yang diperbarui pada saat pencarian dokumen. Jika terjadi perubahan antara operasi pembaruan dan pencarian `fullDocument`, dokumen `fullDocument` mungkin tidak mewakili status dokumen pada waktu pembaruan.

Untuk membuat objek aliran dengan pencarian pembaruan diaktifkan, gunakan contoh ini:

```
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()
```

Output dari objek stream akan terlihat seperti ini:

```
{'_id': {'_data': '015daf9b7c00000001010000000100009025'},
'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}}}
```

Melanjutkan aliran perubahan

Anda dapat melanjutkan aliran perubahan nanti dengan menggunakan token resume, yang sama dengan bidang `_id` dari dokumen peristiwa perubahan yang terakhir diambil.

```
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}}
"""
```

Melanjutkan aliran perubahan dengan startAtOperationTime

Anda dapat melanjutkan aliran perubahan nanti dari stempel waktu tertentu dengan menggunakan `startAtOperationTime`.

> **Note**
>
> Kemampuan untuk menggunakan `startAtOperationTime` tersedia di Amazon DocumentDB 4.0+. Saat menggunakan `startAtOperationTime`, kursor aliran perubahan hanya akan mengembalikan perubahan yang terjadi pada atau setelah Stempel Waktu yang

ditentukan. Perintah `startAtOperationTime` dan `resumeAfter` saling eksklusif dan karenanya tidak dapat digunakan bersama.

```
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}}
"""
```

Melanjutkan aliran perubahan dengan posting BatchResumeToken

Aliran perubahan Amazon DocumentDB sekarang mengembalikan bidang tambahan yang disebut. `postBatchResumeToken` Bidang ini dikembalikan dari `$changestream` perintah dan `getMore` perintah.

Contoh `$changestream` perintah di Python:

```
db.command({"aggregate": "sales", "pipeline": [{ "$changeStream": {}}], "cursor":
 {"batchSize": 1}})
```

Keluaran yang diharapkan

```
cursor" : {
   "firstBatch" : [ ],
```

```
   "postBatchResumeToken" : {"_data" : "0167c8cbe60000000004"},
   "id" : NumberLong("9660788144470"),
   "ns" : "test.sales"
}
```

Contoh `getMore` perintah di Python:

```
db.command({"getMore": NumberLong(<cursor id>), "collection": "sales", "batchSize":
 1 })
```

Output yang diharapkan

```
cursor" : {
   "nextBatch" : [ ],
   "postBatchResumeToken" : {"_data" : "0167c8cbe60000000004"},
   "id" : NumberLong("9660788144470"),
   "ns" : "test.sales"
}
```

`postBatchResumeToken`Bidang ini dapat digunakan untuk membuka kursor aliran perubahan baru di `resumeAfter` lapangan, mirip dengan cara token resume digunakan.

Buka aliran mulai setelah dipilih`postBatchResumeToken`:

```
post_batch_resume_token = output['cursor']['postBatchResumeToken']
stream = db.watch(full_document='updateLookup', resume_after=post_batch_resume_token)
```

Tidak seperti token resume biasa yang selalu sesuai dengan entri log operasi (oplog) yang mencerminkan peristiwa aktual, `postBatchResumeToken` sesuai dengan entri oplog yang telah dipindai aliran perubahan di server, yang belum tentu merupakan perubahan yang cocok.

Mencoba melanjutkan dengan token resume reguler lama akan memaksa database untuk memindai semua entri oplog antara stempel waktu yang ditentukan dan waktu saat ini. Ini dapat menghasilkan banyak kueri secara internal dengan setiap pemindaian sub-kueri untuk jangka waktu yang singkat. Ini akan menyebabkan lonjakan penggunaan CPU dan menurunkan kinerja database. Melanjutkan dengan yang terakhir `postBatchResumeToken` melewatkan pemindaian entri oplog yang tak tertandingi.

Transaksi di aliran perubahan

Peristiwa aliran perubahan tidak akan berisi peristiwa dari transaksi yang and/or dibatalkan tanpa komitmen. Misalnya, jika Anda memulai transaksi dengan satu `INSERT` operasi dan satu `UPDATE` operasi, dan jika `INSERT` operasi Anda berhasil, tetapi `UPDATE` operasi gagal, transaksi akan dibatalkan. Karena transaksi ini digulung kembali, aliran perubahan Anda tidak akan berisi peristiwa apa pun untuk transaksi ini.

Memodifikasi durasi retensi log aliran perubahan

Anda dapat mengubah durasi retensi log aliran perubahan menjadi antara 1 jam dan 7 hari menggunakan Konsol Manajemen AWS atau AWS CLI.

Using the Konsol Manajemen AWS

Untuk memodifikasi durasi retensi log aliran perubahan

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Grup parameter.

   > **Tip**
   >
   > Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu (≡) di pojok kiri atas halaman.

3. Di panel Grup parameter, pilih grup parameter klaster yang terkait dengan klaster Anda. Untuk mengidentifikasi grup parameter klaster yang terkait dengan klaster Anda, lihat Menentukan grup parameter cluster Amazon DocumentDB.
4. Halaman yang dihasilkan menunjukkan parameter dan detailnya yang sesuai untuk grup parameter klaster Anda. Pilih parameter `change_stream_log_retention_duration`.
5. Di kanan atas halaman, pilih Edit untuk mengubah nilai parameter. `change_stream_log_retention_duration`Parameter dapat dimodifikasi menjadi antara 1 jam dan 7 hari.
6. Buat perubahan Anda, lalu pilih Modifikasi parameter klaster untuk menyimpan perubahan. Untuk membuang perubahan Anda, pilih Batal.

Using the AWS CLI

Untuk mengubah parameter `change_stream_log_retention_duration` grup parameter klaster Anda, gunakan operasi `modify-db-cluster-parameter-group` dengan parameter berikut:

- **`--db-cluster-parameter-group-name`** — Diperlukan. Nama grup parameter klaster yang Anda modifikasi. Untuk mengidentifikasi grup parameter klaster yang terkait dengan klaster Anda, lihat Menentukan grup parameter cluster Amazon DocumentDB.
- **`--parameters`** — Diperlukan. Parameter yang Anda modifikasi. Setiap entri parameter harus menyertakan hal berikut:
  - **`ParameterName`** — Nama parameter yang Anda modifikasi. Dalam hal ini, itu adalah `change_stream_log_retention_duration`
  - **`ParameterValue`** — Nilai baru untuk parameter ini.
  - **`ApplyMethod`** — Bagaimana Anda ingin perubahan pada parameter ini diterapkan. Nilai yang diizinkan adalah `immediate` dan `pending-reboot`.

> **Note**
>
> Parameter dengan `ApplyType` dari `static` harus memiliki `ApplyMethod` dari `pending-reboot`.

1. Untuk mengubah nilai parameter `change_stream_log_retention_duration`, jalankan perintah berikut dan ganti `parameter-value` dengan nilai yang ingin Anda modifikasi parameternya.

   Untuk Linux, macOS, atau 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"
   ```

   Untuk 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"
```

Output dari operasi ini terlihat seperti berikut (format JSON).

```
{
    "DBClusterParameterGroupName": "sample-parameter-group"
}
```

2. Tunggu paling tidak 5 menit.
3. Daftar nilai parameter dari `sample-parameter-group` untuk memastikan bahwa perubahan Anda telah dibuat.

   Untuk Linux, macOS, atau Unix:

```
aws docdb describe-db-cluster-parameters \
    --db-cluster-parameter-group-name sample-parameter-group
```

   Untuk Windows:

```
aws docdb describe-db-cluster-parameters ^
    --db-cluster-parameter-group-name sample-parameter-group
```

   Keluaran dari operasi ini terlihat seperti berikut ini (format 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**
>
> Mengubah retensi log aliran tidak akan menghapus log yang lebih lama dari `change_stream_log_retention_duration` nilai yang dikonfigurasi hingga ukuran log lebih besar dari (>) 51.200MB.

Menggunakan aliran perubahan pada instance sekunder

Untuk memulai menggunakan aliran perubahan pada instance sekunder, buka kursor aliran ubah dengan `readPreference` sebagai sekunder.

Anda dapat membuka kursor aliran perubahan untuk melihat peristiwa perubahan pada koleksi tertentu atau semua koleksi dalam cluster atau database. Anda dapat membuka kursor aliran perubahan pada instans Amazon DocumentDB apa pun dan mengambil dokumen aliran perubahan dari instance penulis dan pembaca. Anda dapat membagikan token aliran perubahan (seperti `resumeToken` atau`startOperationTime`) di berbagai kursor aliran perubahan yang dibuka pada instance penulis dan pembaca.

Contoh

```
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)
```

Pedoman dan batasan untuk aliran perubahan pada instance sekunder

- Peristiwa aliran perubahan perlu direplikasi dari instance primer ke instance sekunder. Anda dapat memantau lag dari `DBInstanceReplicaLag` metrik di Amazon CloudWatch.
- Stempel waktu pada instance sekunder mungkin tidak selalu sinkron dengan instance utama. Dalam hal ini, harapkan penundaan pada stempel waktu instance sekunder sehingga dapat mengejar ketinggalan. Sebagai praktik terbaik, kami sarankan `resumeToken` untuk menggunakan `startAtOperationTime` atau memulai jam tangan pada instance sekunder.
- Anda mungkin mengalami throughput yang lebih rendah pada instance sekunder dibandingkan dengan instance utama jika ukuran dokumen Anda besar, Anda melakukannya`fullDocumentLookup`, dan ada beban kerja penulisan bersamaan yang tinggi pada instance utama. Sebagai praktik terbaik, kami sarankan Anda memantau rasio hit cache buffer Anda di sekunder dan memastikan bahwa rasio hit cache buffer tinggi.

Menggunakan Collation di Amazon DocumentDB 8.0

Amazon DocumentDB 8.0 sekarang mendukung pemeriksaan. Collation memungkinkan Anda mengonfigurasi aturan khusus bahasa untuk perbandingan string. Dengan pemeriksaan, Anda dapat menentukan aturan untuk perbandingan peka huruf besar/kecil, atau menentukan bahasa lokal. Collation dapat dikonfigurasi pada tingkat koleksi atau tingkat indeks di DocumentDB 8.0. Saat Anda

menggunakan pemeriksaan di DocumentDB, secara internal dokumen pemeriksaan dibuat dengan parameter berikut.

```
{
locale: string,
caseLevel: boolean,
caseFirst: string,
strength: int,
numericOrdering: boolean,
alternate: string,
maxVariable: string,
backwards: boolean,
normalization: boolean
}
```

Batasan

Collation memiliki batasan berikut di Amazon DocumentDB:

- Collation kompatibel dengan planner v3 yang tersedia di Amazon DocumentDB 8.0. Beralih ke plannerv2 atau plannerv1 dapat menyebabkan perilaku yang tidak konsisten, termasuk kesalahan “Indeks tidak ditemukan”.
- Karena perbedaan pustaka yang melekat, jika koleksi mongodb dengan pemeriksaan diekspor, Anda perlu memperbarui file metadata.bson dan mengubah versinya dari 57.1 menjadi 60.2 sebelum migrasi.
- Dalam kasus yang jarang terjadi, pengaturan pemeriksaan Anda dapat melanggar batas internal jumlah karakter, menyebabkan kesalahan berikut. “Kesalahan: Dokumen pemeriksaan memiliki atribut non-default lebih dari yang didukung. Tolong kurangi jumlah opsi.” Dalam hal ini, cobalah untuk mengurangi opsi yang Anda berikan dalam dokumen pemeriksaan, atau sebagai alternatif Anda dapat mencoba dan menggunakan nilai default sedapat mungkin.

Menggunakan Tampilan di Amazon DocumentDB 8.0

- Amazon DocumentDB 8.0 sekarang mendukung tampilan. Tampilan berfungsi sebagai koleksi virtual yang menyajikan data berdasarkan operasi agregasi tertentu. Saat membuat tampilan, Anda menentukan kueri yang mengubah data dari satu atau beberapa koleksi sumber. Amazon

DocumentDB 8.0 mengeksekusi kueri ini setiap kali tampilan diakses, tanpa menghabiskan sumber daya penyimpanan tambahan. Tidak seperti koleksi standar, tampilan di Amazon DocumentDB 8.0 tidak menyimpan dokumen pada disk, menjadikannya solusi yang efisien untuk menyajikan data yang diubah atau difilter ke aplikasi. Untuk membuat tampilan di Amazon DocumentDB, Anda dapat menggunakan perintah createView atau metode pembantu db.createView ():

```
db.createView("viewName","sourceCollection",
[
    { $match: { status: "active" } },
    { $project: { _id: 1, name: 1, email: 1 } }
]
)
```

Ini membuat tampilan bernama “ViewName” berdasarkan “sourceCollection” yang hanya menyertakan dokumen aktif dan memproyeksikan hanya bidang _id, nama, dan email. Tampilan di Amazon DocumentDB hanya bisa dibaca. Operasi tulis pada tampilan akan mengembalikan kesalahan. Untuk kinerja optimal dengan kumpulan data yang besar, Anda dapat menyusun jalur pipa tampilan untuk memaksimalkan efisiensi. Untuk pipeline agregasi yang kompleks, disarankan untuk menggunakan tahap $match sebagai tahap pertama atau awal pipeline untuk mengurangi jumlah dokumen yang perlu diproses oleh tahap selanjutnya, sehingga meningkatkan kinerja kueri.

Praktik Terbaik

Beberapa praktik terbaik untuk diikuti dengan Tampilan tercantum di bawah ini.

- Filter Awal: Gunakan tahap $match di awal pipeline tampilan untuk mengurangi jumlah data yang diproses.
- Hindari Agregasi Kompleks: Untuk tampilan yang sering diakses dengan agregasi kompleks, pertimbangkan untuk membuat koleksi terpisah dengan hasil yang telah dihitung sebelumnya yang diperbarui secara berkala.
- Perencanaan Indeks: Pastikan bahwa bidang yang digunakan dalam pipeline tampilan, terutama dalam operasi $match dan $sort, diindeks dengan benar pada koleksi sumber.
- Pengoptimalan Kueri: Gunakan perintah jelaskan untuk memahami bagaimana kueri tampilan Anda dijalankan dan mengoptimalkannya.

- Alternatif untuk Tampilan: Mengingat perbedaan fungsional antara tampilan Amazon DocumentDB dan MongoDB, pertimbangkan untuk menggunakan koleksi reguler dengan pembaruan terjadwal sebagai alternatif tampilan saat menghadapi batasan.

Kompatibilitas Operator Agregator

Amazon DocumentDB mendukung banyak operator agregasi dalam definisi tampilan sambil terus memperluas kompatibilitas. Saat menggunakan tampilan, fokuslah pada operator yang didukung ini:

- $ match untuk memfilter dokumen
- $project untuk seleksi lapangan dan transformasi
- $addFields untuk menambahkan bidang yang dihitung
- $ sort untuk memesan hasil
- $limit dan $skip untuk pagination

Beberapa operator khusus seperti $currentTop, $replaceRoot, dan $geonear saat ini bekerja dalam kueri agregasi langsung daripada definisi tampilan.

Memanfaatkan Indeks dan Tampilan

Tampilan di Amazon DocumentDB 8.0 menggunakan indeks koleksi yang mendasarinya. Akibatnya, Anda tidak dapat membuat, menjatuhkan, atau membangun kembali indeks pada tampilan secara langsung. Namun, indeks yang dirancang dengan baik pada koleksi sumber dapat secara signifikan meningkatkan kinerja kueri tampilan Di bawah ini adalah beberapa langkah untuk mengoptimalkan kinerja kueri pada tampilan:

- Pastikan indeks yang sesuai ada pada kolom koleksi sumber yang digunakan dalam pipeline tampilan, terutama dalam operasi $match dan $sort
- Gunakan metode explain () untuk menganalisis rencana eksekusi kueri dan memverifikasi penggunaan indeks. E.g., `db.viewName.find({...}).explain()`

Penggunaan AWS Lambda dengan aliran perubahan

Amazon DocumentDB terintegrasi AWS Lambda sehingga Anda dapat menggunakan fungsi Lambda untuk memproses catatan dalam aliran perubahan. Pemetaan sumber peristiwa Lambda adalah

sumber daya yang dapat digunakan untuk menjalankan fungsi Lambda untuk memproses peristiwa Amazon DocumentDB yang tidak langsung memanggil Lambda. Dengan aliran perubahan Amazon DocumentDB sebagai sumber peristiwa, Anda dapat membuat aplikasi berbasis peristiwa yang merespons perubahan dalam data Anda. Misalnya, Anda dapat menggunakan fungsi Lambda untuk memproses dokumen baru, melacak pembaruan ke dokumen yang ada, atau mencatat dokumen yang dihapus.

Anda dapat mengonfigurasi pemetaan sumber peristiwa untuk mengirim catatan dari aliran perubahan Amazon DocumentDB ke fungsi Lambda. Acara dapat dikirim satu per satu atau batch untuk meningkatkan efisiensi dan akan diproses secara berurutan. Anda dapat mengonfigurasi perilaku batching pemetaan sumber peristiwa berdasarkan durasi jendela waktu tertentu (0 - 300 detik) atau jumlah catatan batch (batas maksimal 10.000 catatan). Anda dapat membuat beberapa pemetaan sumber peristiwa untuk memproses data yang sama dengan beberapa fungsi Lambda, atau untuk memproses item yang berbeda dari beberapa aliran dengan satu fungsi.

Jika fungsi Anda mengembalikan kesalahan, Lambda mencoba ulang batch hingga berhasil diproses. Jika peristiwa dalam aliran perubahan telah kedaluwarsa, Lambda akan menonaktifkan pemetaan sumber peristiwa. Dalam hal ini, Anda dapat membuat pemetaan sumber peristiwa baru dan mengonfigurasinya dengan posisi awal pilihan Anda. Pemetaan sumber peristiwa Lambda memproses peristiwa setidaknya sekali karena sifat pollernya yang terdistribusi. Akibatnya, fungsi Lambda Anda mungkin menerima kejadian duplikat dalam situasi yang jarang terjadi. Ikuti praktik terbaik untuk bekerja dengan AWS Lambda fungsi dan buat fungsi idempoten untuk menghindari masalah yang terkait dengan peristiwa duplikat. Untuk informasi selengkapnya, lihat Menggunakan AWS Lambda console dengan Amazon DocumentDB di AWS Lambda Panduan Pengembang.

Sebagai praktik terbaik kinerja, fungsi Lambda harus berumur pendek. Untuk menghindari penundaan pemrosesan yang tidak perlu, logika yang rumit juga tidak boleh dijalankan. Khususnya untuk aliran kecepatan tinggi, lebih baik memicu alur kerja fungsi langkah pasca-pemrosesan asinkron daripada Lambdas yang berjalan lama secara sinkron. Untuk informasi selengkapnya AWS Lambda, lihat Panduan AWS Lambda Pengembang.

Batasan

Berikut ini adalah batasan yang perlu dipertimbangkan saat bekerja dengan Amazon AWS Lambda DocumentDB dan:

- AWS Lambda saat ini hanya didukung di Amazon DocumentDB 4.0 dan 5.0.
- AWS Lambda saat ini tidak didukung pada cluster elastis atau cluster global.

- AWS Lambda ukuran muatan tidak boleh melebihi 6MB. Untuk informasi selengkapnya tentang ukuran batch Lambda, lihat “Perilaku pengelompokan” di bagian pemetaan sumber peristiwa Lambda di Panduan Pengembang.AWS Lambda

Menggunakan cluster elastis Amazon DocumentDB

Cluster elastis Amazon DocumentDB mendukung beban kerja dengan jutaan reads/writes per detik dan kapasitas penyimpanan petabyte. Cluster elastis juga menyederhanakan cara pengembang berinteraksi dengan Amazon DocumentDB dengan menghilangkan kebutuhan untuk memilih, mengelola, atau meningkatkan instance.

> **Note**
>
> Cluster elastis menggunakan protokol kawat yang kompatibel dengan MongoDB 5.0. Mereka tidak didukung di Amazon DocumentDB engine versi 8.0, dan tidak tersedia di wilayah Tiongkok atau. AWS GovCloud

Cluster elastis Amazon DocumentDB dibuat untuk:

- Berikan solusi bagi pelanggan yang mencari database yang menyediakan skala hampir tak terbatas dengan kemampuan kueri yang kaya dan kompatibilitas API MongoDB.
- Berikan pelanggan batas koneksi yang lebih tinggi, dan untuk mengurangi downtime dari patching.
- Terus berinvestasi dalam arsitektur cloud-native, elastis, dan terkemuka di kelas untuk beban kerja JSON.

Topik

- Kasus penggunaan cluster elastis
- Keuntungan dari cluster elastis
- Wilayah cluster elastis dan ketersediaan versi
- Batasan
- Cluster elastis Amazon DocumentDB: cara kerjanya
- Memulai dengan cluster elastis Amazon DocumentDB
- Praktik terbaik cluster elastis Amazon DocumentDB
- Mengelola cluster elastis Amazon DocumentDB
- Enkripsi data saat istirahat untuk cluster elastis Amazon DocumentDB
- Service-linked peran dalam kelompok elastis

Kasus penggunaan cluster elastis

Basis data dokumen berguna untuk beban kerja yang memerlukan skema fleksibel untuk pengembangan yang cepat dan berulang. Misalnya kasus penggunaan Amazon DocumentDB, lihat. Kasus penggunaan basis data dokumen

Berikut ini adalah beberapa contoh kasus penggunaan di mana cluster elastis dapat memberikan keuntungan yang signifikan:

Profil pengguna

Karena database dokumen memiliki skema yang fleksibel, mereka dapat menyimpan dokumen yang memiliki atribut dan nilai data yang berbeda dalam skala besar. Cluster elastis adalah solusi praktis untuk profil online di mana pengguna yang berbeda memberikan berbagai jenis informasi. Misalkan aplikasi Anda mendukung ratusan juta profil pengguna. Anda dapat menggunakan cluster elastis untuk mendukung aplikasi tersebut karena mereka dapat ditingkatkan dan keluar untuk mendukung jutaan penulisan dan pembacaan ke profil pengguna ini. Anda juga dapat menurunkan skala untuk jam sibuk untuk mengurangi biaya.

Manajemen konten dan catatan sejarah

Untuk mengelola konten secara efektif, Anda harus dapat mengumpulkan dan mengagregasikan konten dari berbagai sumber, dan kemudian mengirimkannya ke pelanggan. Akibat skema fleksibel mereka, basis data dokumen sempurna untuk mengumpulkan dan menyimpan jenis data apa pun. Anda dapat menggunakannya untuk membuat dan menggabungkan jenis konten baru, termasuk konten buatan pengguna seperti gambar, komentar, dan video. Seiring waktu, database Anda mungkin memerlukan lebih banyak penyimpanan. Dengan cluster elastis, Anda dapat mendistribusikan data Anda ke lebih banyak volume penyimpanan yang memungkinkan Anda menyimpan petabyte data dalam satu cluster.

Keuntungan dari cluster elastis

AWS Integrasi layanan

Cluster elastis Amazon DocumentDB terintegrasi dengan layanan AWS lain dengan cara yang sama seperti Amazon DocumentDB:

- Migrasi - Anda dapat menggunakan AWS Database Migration Service (DMS) untuk bermigrasi dari MongoDB dan database relasional lainnya ke cluster elastis Amazon DocumentDB.
- Pemantauan — Anda dapat memantau kesehatan dan kinerja cluster elastis Anda menggunakan Amazon CloudWatch.
- Keamanan — Anda dapat mengatur autentikasi dan otorisasi melalui AWS Identity and Access Management (IAM) untuk mengelola sumber daya cluster elastis Anda seperti membuat dan memperbarui cluster elastis, dan menggunakan Amazon VPC untuk koneksi yang aman. VPC-only Login otentikasi IAM untuk cluster elastis tidak didukung.
- Manajemen data - Anda dapat menggunakan AWS Glue untuk mengimpor dan mengekspor data AWS layanan from/to lain seperti Amazon S3, Amazon Redshift, dan OpenSearch Amazon Service.

Wilayah cluster elastis dan ketersediaan versi

Ketersediaan wilayah untuk cluster elastis

Tabel berikut menunjukkan AWS wilayah tempat cluster elastis Amazon DocumentDB saat ini tersedia dan titik akhir untuk setiap wilayah.

Nama wilayah	Region	Zona ketersediaan
AS Timur (Virginia Utara)	`us-east-1`	5
AS Timur (Ohio)	`us-east-2`	3
AS Barat (Oregon)	`us-west-2`	3
Asia Pasifik (Hong Kong)	`ap-east-1`	3
Asia Pasifik (Mumbai)	`ap-south-1`	3
Asia Pasifik (Seoul)	`ap-northeast-2`	3
Asia Pasifik (Singapura)	`ap-southeast-1`	3
Asia Pasifik (Sydney)	`ap-southeast-2`	3
Asia Pasifik (Tokyo)	`ap-northeast-1`	3

Nama wilayah	Region	Zona ketersediaan
Kanada (Pusat)	`ca-central-1`	3
Amerika Selatan (Sao Paulo)	`sa-east-1`	3
Eropa (Frankfurt)	`eu-central-1`	3
Eropa (Irlandia)	`eu-west-1`	3
Eropa (London)	`eu-west-2`	3
Europe (Milan)	`eu-south-1`	3
Eropa (Paris)	`eu-west-3`	3

Ketersediaan versi

Cluster elastis mendukung protokol kabel MongoDB 5.0-compatable. Untuk perbedaan antara cluster berbasis instans Amazon DocumentDB 4.0 dan cluster elastis, lihat. Perbedaan fungsional antara Amazon DocumentDB 4.0 dan cluster elastis Catatan: Cluster Elastis tidak didukung di Amazon DocumentDB 8.0

Batasan

Manajemen cluster elastis

Fitur dan kemampuan manajemen klaster berikut tidak didukung dalam rilis ini:

- Port khusus (hanya 27017 yang didukung)
- Kemampuan untuk menggunakan pengguna dan peran IAM untuk mengautentikasi ke dalam database
- Kemampuan untuk membuat cluster global
- Acara Amazon DocumentDB yang ada dan berlangganan acara
- Rentang sharding
- Koleksi Shard yang ada

- Multi-field kunci pecahan
- Ubah kunci pecahan
- Point-in-time mengembalikan
- Kloning
- Wawasan Performa

> **Note**
>
> Untuk informasi tentang batas cluster elastis, lihatKuota dan batas Amazon DocumentDB.

Operasi kueri dan tulis

Perintah dan kemampuan operasi query dan write berikut tidak didukung dalam rilis ini:

- Perintah DDL selama operasi penskalaan
- Profiler
- Grup parameter
- AWS Config
- AWS Backup

Pengumpulan dan manajemen indeks

Fitur manajemen koleksi dan indeks berikut tidak didukung dalam rilis ini:

- Indeks unik
- Indeks sebagian
- Indeks teks
- Indeks vektor
- Kompresi dokumen

Administrasi dan diagnostik

Perintah dan kemampuan administrasi dan diagnostik berikut tidak didukung dalam rilis ini:

- AWS Secrets Manager
- Role-based-access-control (RBAC) peran khusus.
- Saat menghubungkan, tulis kekhawatiran 0 tidak didukung.
- Mengubah subnet milik VPC yang saat ini tidak ditetapkan ke cluster elastis yang ada.

Opt-in fitur

Fitur keikutsertaan Amazon DocumentDB berikut tidak didukung dalam rilis ini:

- Transaksi ACID
- DDL/DML audit
- Mengelola aliran
- Perintah sesi

Cluster elastis Amazon DocumentDB: cara kerjanya

Topik di bagian ini memberikan informasi tentang mekanisme dan fungsi yang mendukung cluster elastis Amazon DocumentDB.

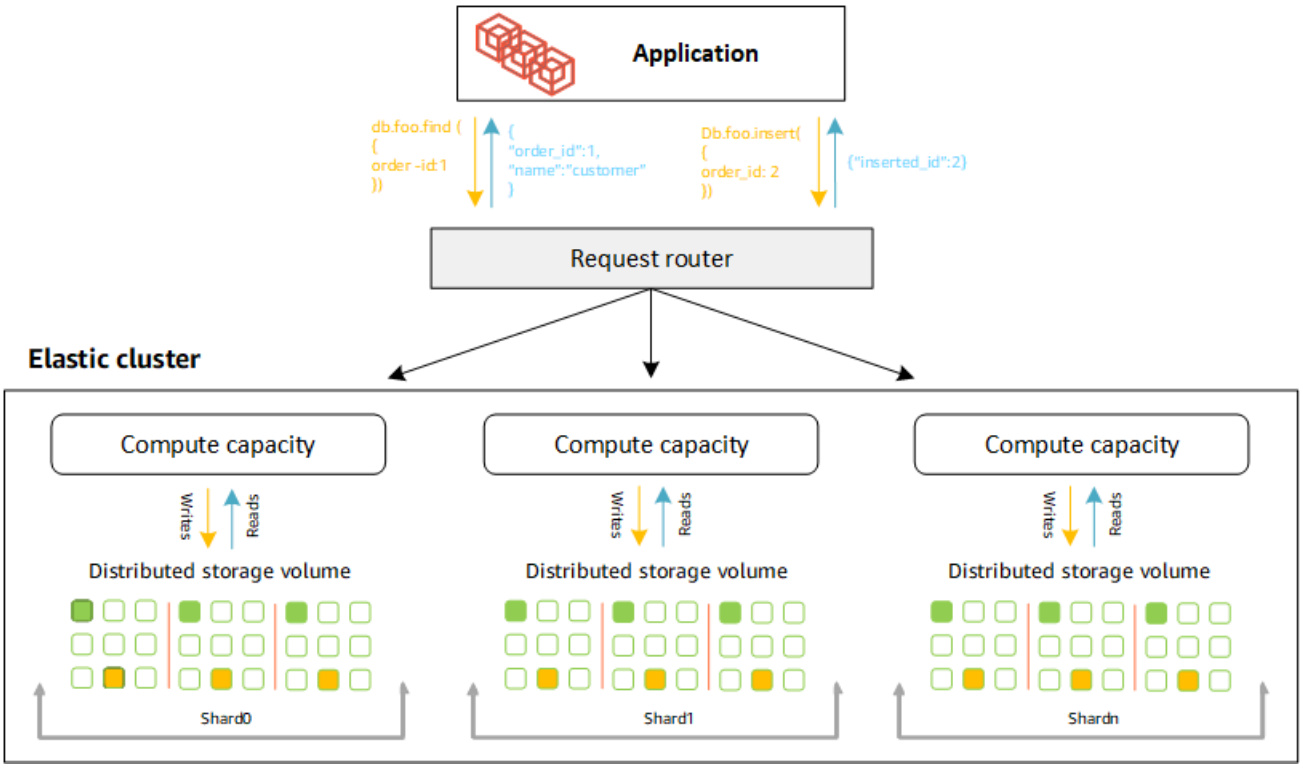
Topik

- Sharding cluster elastis Amazon DocumentDB
- Migrasi cluster elastis
- Penskalaan cluster elastis
- Keandalan cluster elastis
- Penyimpanan dan ketersediaan cluster elastis
- Perbedaan fungsional antara Amazon DocumentDB 4.0 dan cluster elastis

Sharding cluster elastis Amazon DocumentDB

Cluster elastis Amazon DocumentDB menggunakan sharding berbasis hash untuk mempartisi data di seluruh sistem penyimpanan terdistribusi. Sharding, juga dikenal sebagai partisi, membagi kumpulan data besar menjadi kumpulan data kecil di beberapa node yang memungkinkan Anda untuk skala database Anda di luar batas penskalaan vertikal. Cluster elastis menggunakan pemisahan, atau “decoupling,” komputasi dan penyimpanan di Amazon DocumentDB, memungkinkan Anda

untuk menskalakan secara independen satu sama lain. Alih-alih mempartisi ulang koleksi dengan memindahkan potongan kecil data antara node komputasi, cluster elastis menyalin data secara efisien dalam sistem penyimpanan terdistribusi.



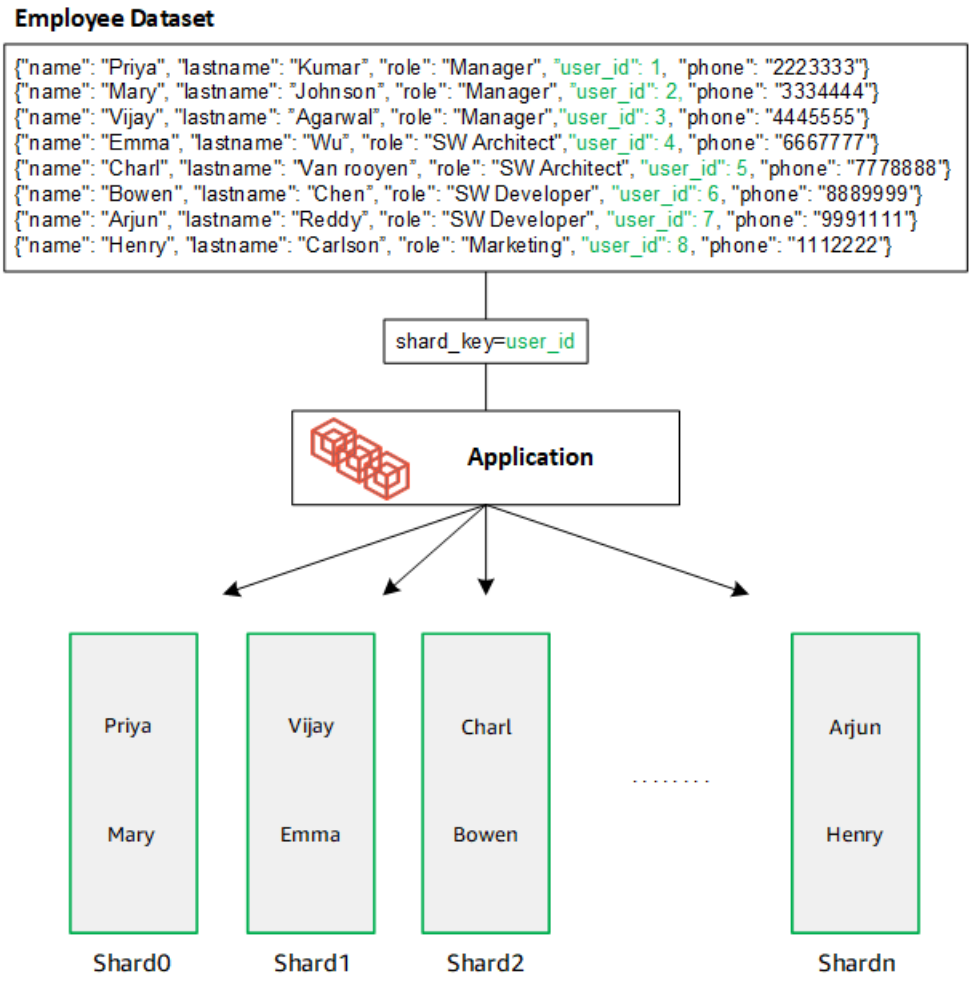
Definisi pecahan

Definisi nomenklatur pecahan:

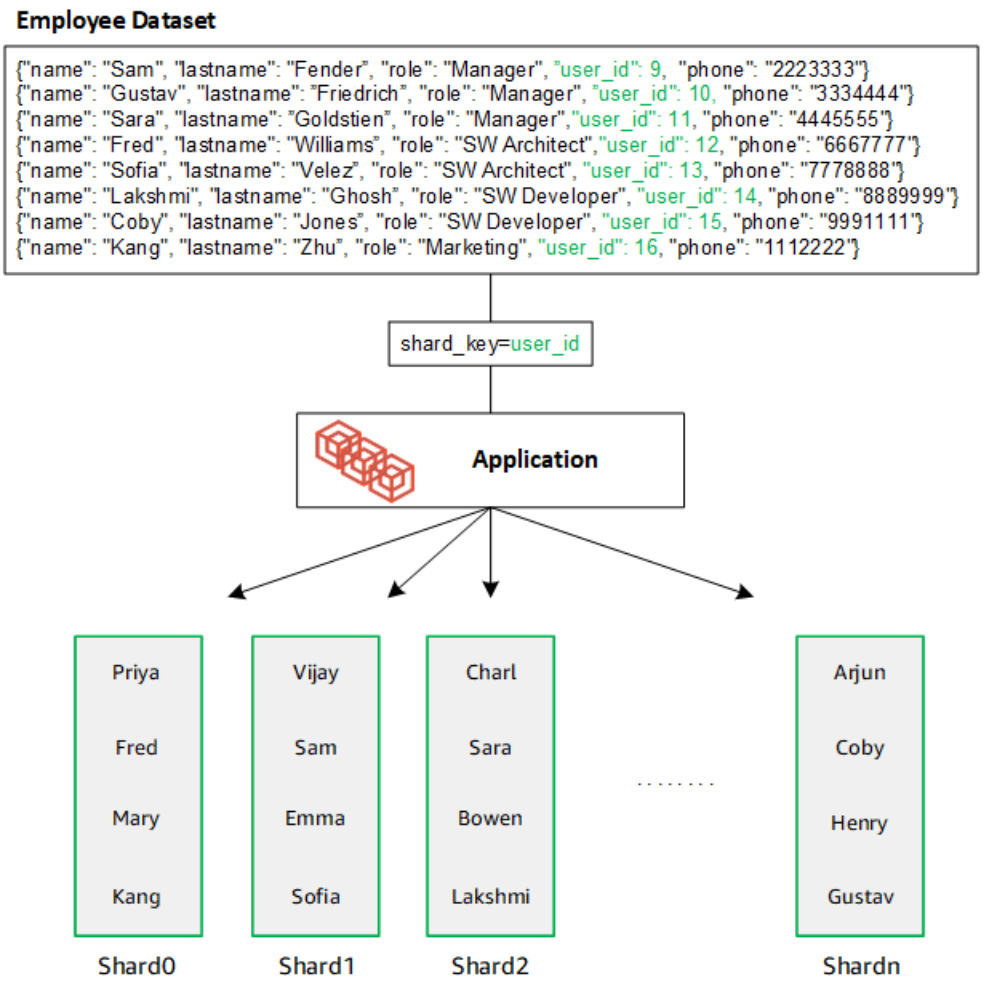
- Pecahan — Pecahan menyediakan komputasi untuk cluster elastis. Ini akan memiliki contoh penulis tunggal dan 0—15 replika baca. Secara default, pecahan akan memiliki dua contoh: penulis dan satu replika baca. Anda dapat mengonfigurasi maksimum 32 pecahan dan setiap instance shard dapat memiliki maksimum 64 vCPU.
- Kunci pecahan — Kunci pecahan adalah bidang wajib dalam dokumen JSON Anda dalam koleksi sharded yang digunakan cluster elastis untuk mendistribusikan lalu lintas baca dan tulis ke pecahan yang cocok.
- Koleksi sharded — Koleksi sharded adalah koleksi yang datanya didistribusikan di seluruh cluster elastis di partisi data.
- Partisi — Partisi adalah bagian logis dari data sharded. Saat Anda membuat koleksi sharded, data diatur ke dalam partisi dalam setiap shard secara otomatis berdasarkan kunci shard. Setiap pecahan memiliki beberapa partisi.

Mendistribusikan data di seluruh pecahan yang dikonfigurasi

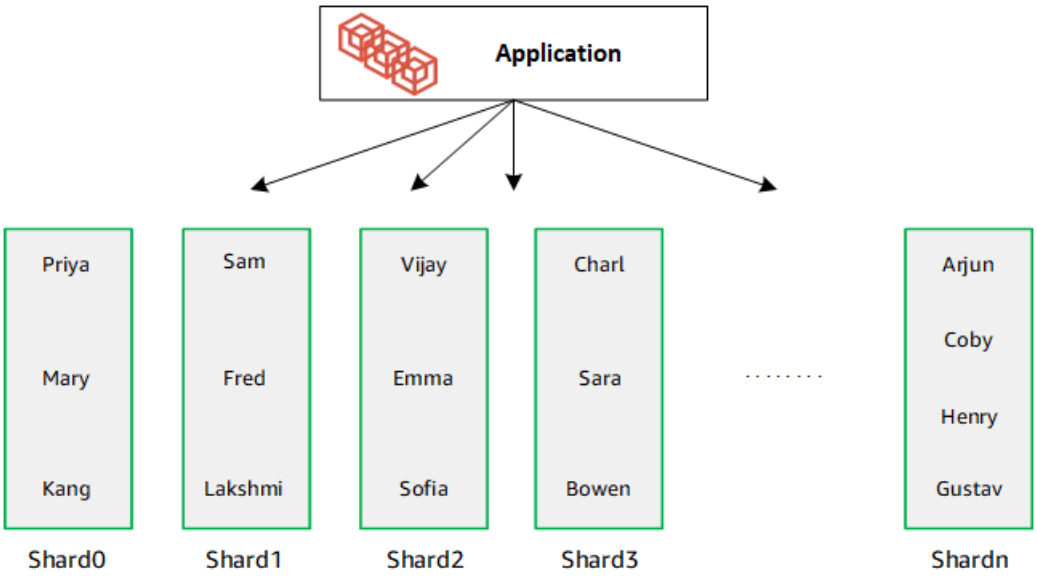
Buat kunci pecahan yang memiliki banyak nilai unik. Kunci shard yang baik akan mempartisi data Anda secara merata di seluruh pecahan yang mendasarinya, memberikan beban kerja Anda throughput dan kinerja terbaik. Contoh berikut adalah data nama karyawan yang menggunakan kunci shard bernama “user_id”:



DocumentDB menggunakan hash sharding untuk mempartisi data Anda di seluruh pecahan yang mendasarinya. Data tambahan dimasukkan dan didistribusikan dengan cara yang sama:



Saat Anda menskalakan database dengan menambahkan pecahan tambahan, Amazon DocumentDB secara otomatis mendistribusikan ulang data:



Migrasi cluster elastis

Amazon DocumentDB mendukung migrasi data sharded MongoDB ke cluster elastis. Metode migrasi offline, online, dan hibrida didukung. Untuk informasi selengkapnya, lihat Migrasi ke Amazon DocumentDB.

Penskalaan cluster elastis

Cluster elastis Amazon DocumentDB memberikan kemampuan untuk meningkatkan jumlah pecahan (skala keluar) di cluster elastis Anda, dan jumlah vCPU yang diterapkan pada setiap pecahan (skala naik). Anda juga dapat mengurangi jumlah pecahan dan kapasitas komputasi (vCPU) sesuai kebutuhan.

Untuk praktik terbaik penskalaan, lihatPenskalaan cluster elastis.

> **Note**
>
> Cluster-level penskalaan juga tersedia. Untuk informasi selengkapnya, lihat Menskalakan cluster Amazon DocumentDB.

Keandalan cluster elastis

Amazon DocumentDB dirancang agar andal, tahan lama, dan toleran terhadap kesalahan. Untuk meningkatkan ketersediaan, cluster elastis menyebarkan dua node per shard yang ditempatkan di Availability Zone yang berbeda. Amazon DocumentDB menyertakan beberapa fitur otomatis yang menjadikannya solusi basis data yang andal. Untuk informasi selengkapnya, lihat Keandalan Amazon DocumentDB.

Penyimpanan dan ketersediaan cluster elastis

Data Amazon DocumentDB disimpan dalam volume klaster, yang merupakan volume virtual tunggal yang menggunakan solid state drive (SSD). Volume cluster terdiri dari enam salinan data Anda, yang direplikasi secara otomatis di beberapa Availability Zone dalam satu AWS Region. Replikasi ini membantu memastikan bahwa data Anda sangat berdaya tahan, dengan kemungkinan kehilangan data yang lebih kecil. Ini juga membantu memastikan bahwa klaster Anda lebih tersedia selama failover karena salinan data Anda sudah ada di Availability Zone lainnya. Untuk detail lebih lanjut tentang penyimpanan, ketersediaan tinggi, dan replikasi lihatAmazon DocumentDB: cara kerjanya.

Perbedaan fungsional antara Amazon DocumentDB 4.0 dan cluster elastis

Perbedaan fungsional berikut ada antara Amazon DocumentDB 4.0 dan cluster elastis.

- Hasil dari `top` dan `collStats` dipartisi oleh pecahan. Untuk koleksi sharded, data didistribusikan di antara beberapa partisi dan `collStats` laporan yang dikumpulkan `collScans` dari partisi.
- Statistik koleksi dari `top` dan `collStats` untuk koleksi sharded diatur ulang saat jumlah pecahan cluster diubah.
- Peran bawaan cadangan sekarang mendukung`serverStatus`. Tindakan - Pengembang dan aplikasi dengan peran cadangan dapat mengumpulkan statistik tentang keadaan cluster Amazon DocumentDB.
- `SecondaryDelaySecs`Bidang menggantikan `slaveDelay` `replSetGetConfig` output.
- `hello`Perintah menggantikan `isMaster` - `hello` mengembalikan dokumen yang menggambarkan peran cluster elastis.
- `$elemMatch`Operator dalam cluster elastis hanya cocok dengan dokumen di tingkat bersarang pertama dari sebuah array. Di Amazon DocumentDB 4.0, operator melintasi semua tingkatan sebelum mengembalikan dokumen yang cocok. Contoh:

```
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 ] } ] ] }
```

- Proyeksi “$” di Amazon DocumentDB 4.0 mengembalikan semua dokumen dengan semua bidang. Dengan cluster elastis, `find` perintah dengan proyeksi “$” mengembalikan dokumen yang cocok dengan parameter kueri yang hanya berisi bidang yang cocok dengan proyeksi “$”.
- Dalam cluster elastis, `find` perintah dengan `$regex` dan parameter `$options` kueri mengembalikan kesalahan: “Tidak dapat mengatur opsi di $regex dan $options.”

- Dengan cluster elastis, `$indexOfCP` sekarang mengembalikan “-1" ketika:
  - substring tidak ditemukan di`string expression`, atau
  - `start`adalah angka yang lebih besar dari`end`, atau
  - `start`adalah angka yang lebih besar dari panjang byte string.

  Di Amazon DocumentDB 4.0`$indexOfCP`, mengembalikan “0" ketika `start` posisi adalah angka yang `end` lebih besar dari atau panjang byte string.
- Dengan cluster elastis, operasi proyeksi di`_id fields`, misalnya`{"_id.nestedField" : 1}`, mengembalikan dokumen yang hanya menyertakan bidang yang diproyeksikan. Sementara itu, di Amazon DocumentDB 4.0, perintah proyeksi bidang bersarang tidak menyaring dokumen apa pun.

Memulai dengan cluster elastis Amazon DocumentDB

Bagian memulai ini memandu Anda tentang bagaimana Anda dapat membuat dan menanyakan cluster elastis pertama Anda.

Ada banyak cara untuk terhubung dan memulai dengan Amazon DocumentDB. Prosedur berikut adalah cara tercepat, paling sederhana, dan termudah bagi pengguna untuk mulai menggunakan database dokumen kami yang kuat. Panduan ini digunakan AWS CloudShelluntuk menghubungkan dan menanyakan cluster Amazon DocumentDB Anda langsung dari. Konsol Manajemen AWS Pelanggan baru yang memenuhi syarat untuk Tingkat AWS Gratis dapat menggunakan Amazon DocumentDB CloudShell dan gratis. Jika AWS CloudShell lingkungan Anda atau cluster elastis Amazon DocumentDB menggunakan sumber daya di luar tingkat gratis, Anda akan dikenakan tarif AWS normal untuk sumber daya tersebut. Panduan ini akan membantu Anda memulai Amazon DocumentDB dalam waktu kurang dari 5 menit.

Topik

- Prasyarat
- Langkah 1: Buat cluster elastis
- Langkah 2: Connect ke cluster elastis Anda
- Langkah 3: Pecahkan koleksi, sisipkan, dan kueri data Anda
- Langkah 4: Jelajahi

Prasyarat

Sebelum membuat klaster Amazon DocumentDB pertama, Anda harus melakukan hal berikut:

Buatlah akun Amazon Web Services (AWS)

Sebelum Anda dapat mulai menggunakan Amazon DocumentDB, Anda harus memiliki akun Amazon Web Services (AWS). AWS Akun ini gratis. Anda hanya membayar untuk layanan dan sumber daya yang Anda gunakan.

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar untuk Akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.
2. Ikuti petunjuk online.

   Bagian dari prosedur pendaftaran melibatkan menerima panggilan telepon atau pesan teks dan memasukkan kode verifikasi pada keypad telepon.

   Saat Anda mendaftar untuk sebuah Akun AWS, sebuah Pengguna root akun AWSdibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya di akun. Sebagai praktik keamanan terbaik, tetapkan akses administratif ke pengguna, dan gunakan hanya pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

Siapkan izin yang diperlukan AWS Identity and Access Management (IAM).

Akses untuk mengelola sumber daya Amazon DocumentDB seperti cluster, instance, dan grup parameter cluster memerlukan kredensil yang dapat digunakan untuk AWS mengautentikasi permintaan Anda. Untuk informasi selengkapnya, lihat Identity and Access Management untuk Amazon DocumentDB.

1. Di bilah pencarian Konsol Manajemen AWS, ketik IAM dan pilih IAM di menu tarik-turun.
2. Setelah Anda berada di konsol IAM, pilih Pengguna dari panel navigasi.
3. Pilih nama pengguna Anda.
4. Pilih Tambahkan izin.
5. Pilih Lampirkan kebijakan secara langsung.
6. Ketik `AmazonDocDBElasticFullAccess` di bilah pencarian dan pilih itu setelah muncul di hasil pencarian.
7. Pilih Berikutnya.
8. Pilih Tambahkan izin.

> **Note**
>
> AWS Akun Anda menyertakan VPC default di setiap wilayah. Jika Anda memilih untuk menggunakan VPC Amazon, selesaikan langkah-langkah dalam topik Buat VPC Amazon di Panduan Pengguna Amazon VPC.

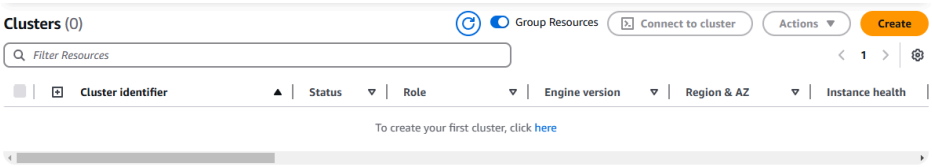
Langkah 1: Buat cluster elastis

Pada bagian ini kami menjelaskan cara membuat cluster elastis baru, menggunakan Konsol Manajemen AWS atau AWS CLI dengan instruksi berikut.

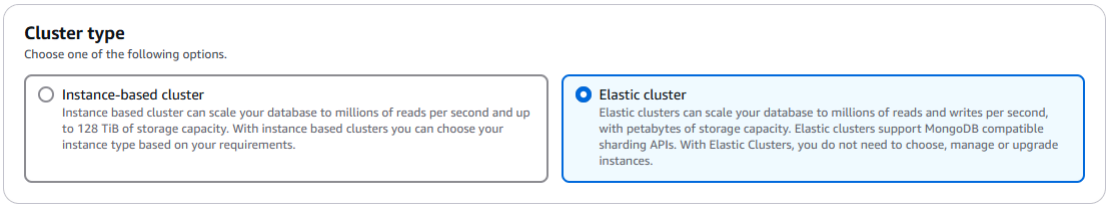
Using the Konsol Manajemen AWS

Untuk membuat konfigurasi cluster elastis menggunakan Konsol Manajemen AWS:

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di Amazon DocumentDB Management Console, di bawah Clusters, pilih Create.



3. Pada halaman cluster Create Amazon DocumentDB, di bagian tipe Cluster, pilih cluster Elastic.



4. Di bagian Konfigurasi, konfigurasikan yang berikut ini:

   a. Di bidang Nama cluster, masukkan pengidentifikasi cluster unik (mengikuti persyaratan penamaan di bawah bidang).

   b. Di bidang Shard count, masukkan jumlah pecahan yang Anda inginkan di cluster Anda. Jumlah maksimum pecahan per cluster adalah 32.

      > **Note**
      >
      > Dua node akan digunakan untuk setiap pecahan. Kedua node akan memiliki kapasitas shard yang sama.

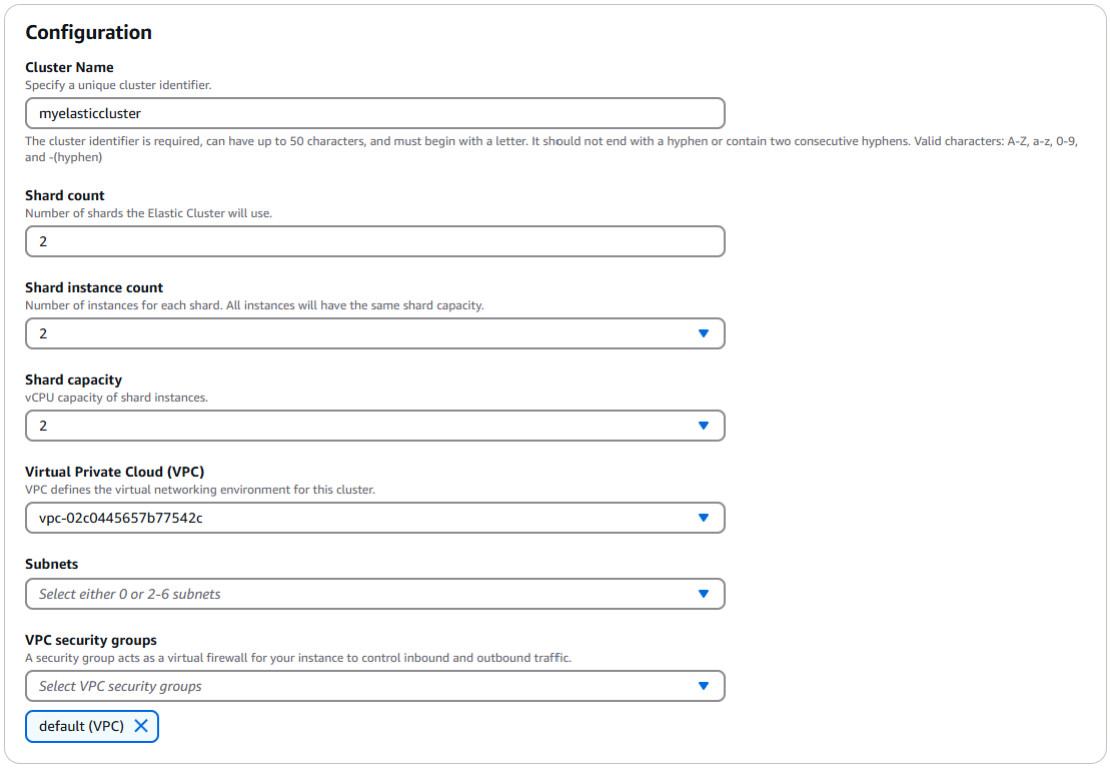
   c. Di bidang hitungan instance Shard, pilih jumlah instance replika yang ingin Anda kaitkan dengan setiap pecahan. Jumlah maksimum instance shard adalah 16, dengan penambahan 1. Semua contoh replika memiliki kapasitas pecahan yang sama seperti yang didefinisikan dalam bidang berikut. Untuk tujuan pengujian, nilai default 2 sudah cukup.

      > **Note**
      >
      > Jumlah contoh replika berlaku untuk semua pecahan di gugus elastis. Nilai hitungan instans pecahan 1 berarti ada satu instance penulis, dan setiap contoh tambahan adalah replika yang dapat digunakan untuk membaca dan untuk meningkatkan ketersediaan. Untuk tujuan pengujian, nilai default 2 sudah cukup.

   d. Di bidang kapasitas Shard, pilih jumlah CPU virtual (vCPU) yang ingin Anda kaitkan dengan setiap instance shard. Jumlah maksimum vCPU per instance shard adalah 64. Nilai yang diizinkan adalah 2, 4, 8, 16, 32, 64. Untuk tujuan pengujian, nilai default 2 sudah cukup.

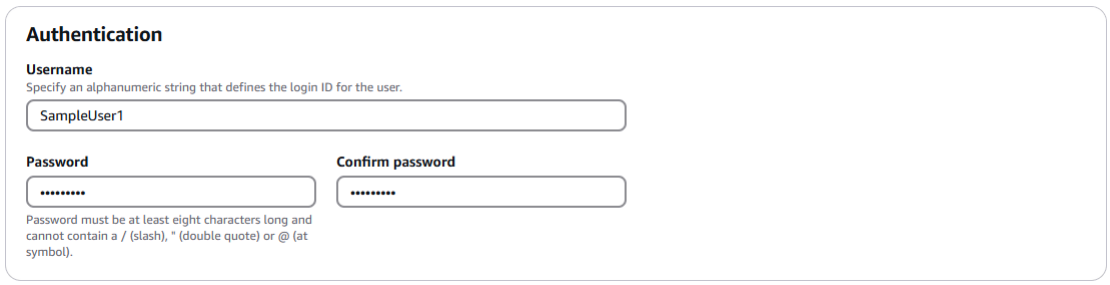
   e. Di bidang Virtual Private Cloud (VPC), pilih VPC dari daftar dropdown.

   f. Untuk grup keamanan Subnet dan VPC, Anda dapat menggunakan default atau memilih tiga subnet pilihan Anda dan hingga tiga grup keamanan VPC (minimal satu).



5. Di bagian Otentikasi, masukkan string yang mengidentifikasi nama login pengguna utama di bidang Nama Pengguna.

   Di bidang Kata Sandi, masukkan kata sandi unik yang sesuai dengan instruksi, lalu konfirmasikan.



6. Di bagian Enkripsi, pertahankan pengaturan default (Kunci Default).

   Secara opsional, Anda dapat memasukkan AWS KMS key ARN yang Anda buat. Untuk informasi selengkapnya, lihat Enkripsi data saat istirahat untuk cluster elastis Amazon DocumentDB.

> **Important**
>
> Enkripsi harus diaktifkan untuk cluster elastis.

7. Di bagian Backup, edit bidang sesuai dengan kebutuhan cadangan Anda. Untuk tujuan pengujian, Anda dapat mempertahankan pengaturan default.



   a. Periode retensi cadangan—Dalam daftar, pilih jumlah hari untuk menyimpan backup otomatis klaster ini sebelum menghapusnya.

   b. Jendela backup—Atur waktu harian dan durasi yang akan digunakan oleh Amazon DocumentDB untuk membuat backup dari klaster ini.

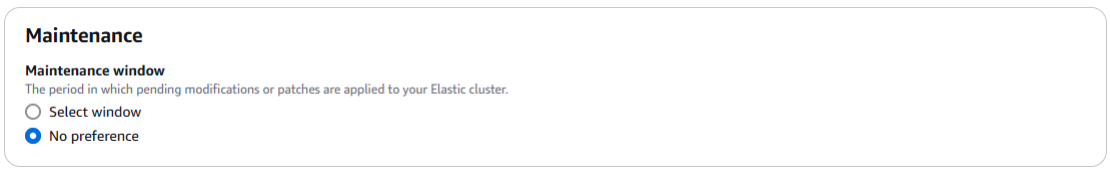
      i. Pilih Pilih jendela jika Anda ingin mengonfigurasi waktu dan durasi saat pencadangan dibuat.

         Waktu mulai—Pada daftar pertama, pilih jam waktu mulai (UTC) untuk memulai pencadangan otomatis. Dalam daftar kedua, pilih menit dari jam pilihan Anda untuk memulai backup otomatis.

         Durasi—Dalam daftar, pilih jumlah jam yang akan dialokasikan untuk membuat cadangan otomatis.

      ii. Pilih Tidak ada preferensi jika Anda ingin Amazon DocumentDB memilih waktu dan durasi saat pencadangan dibuat.

8. Di bagian Maintenance, pilih hari, waktu, dan durasi saat modifikasi atau patch diterapkan ke cluster Anda. Untuk tujuan pengujian, Anda dapat mempertahankan pengaturan default.



9. Pilih Buat klaster.

Cluster elastis sekarang sedang disediakan. Ini bisa memakan waktu hingga beberapa menit untuk menyelesaikannya. Anda dapat terhubung ke klaster Anda ketika status cluster elastis ditampilkan sebagai Tersedia dalam daftar Clusters.

Using the AWS CLI

Untuk membuat cluster elastis menggunakan AWS CLI, gunakan `create-cluster` operasi dengan parameter berikut:

- --cluster-name—Wajib. Nama saat ini dari cluster skala elastis seperti yang dimasukkan selama pembuatan atau terakhir dimodifikasi.
- --shard-capacity—Wajib. Jumlah vCPU yang ditetapkan untuk setiap pecahan. Maksimal adalah 64. Nilai yang diizinkan adalah 2, 4, 8, 16, 32, 64.
- --shard-count—Wajib. Jumlah pecahan yang ditugaskan ke cluster. Maksimal adalah 32.
- --shard-instance-count—Opsional. Jumlah contoh replika yang diterapkan ke semua pecahan di cluster ini. Maksimal adalah 16.
- --admin-user-name—Wajib. Nama pengguna yang terkait dengan pengguna admin.
- --admin-user-password—Wajib. Kata sandi yang terkait dengan pengguna admin.
- --auth-type—Wajib. Jenis otentikasi yang digunakan untuk menentukan di mana untuk mengambil password yang digunakan untuk mengakses cluster elastis. Jenis yang valid adalah `PLAIN_TEXT` atau`SECRET_ARN`.
- --vpc-security-group-ids—Opsional. Konfigurasikan daftar grup keamanan VPC EC2 untuk diasosiasikan dengan cluster ini.
- --preferred-maintenance-window—Opsional. Konfigurasikan rentang waktu mingguan selama pemeliharaan sistem dapat terjadi, di Universal Coordinated Time (UTC).

  Formatnya adalah:`ddd:hh24:mi-ddd:hh24:mi`. Hari yang berlaku (ddd): Senin, Sel, Rabu, Kam, Jum, Sabtu, Minggu

  Defaultnya adalah jendela 30 menit yang dipilih secara acak dari blok waktu 8 jam untuk setiap Wilayah Amazon Web Services, yang terjadi pada hari acak dalam seminggu.

  Jendela minimal 30 menit.
- --kms-key-id—Opsional. Konfigurasikan pengidentifikasi kunci KMS untuk cluster terenkripsi.

  Pengidentifikasi kunci KMS adalah Amazon Resource Name (ARN) untuk kunci enkripsi. AWS KMS Jika Anda membuat klaster menggunakan akun Amazon Web Services yang sama yang

memiliki kunci enkripsi KMS yang digunakan untuk mengenkripsi cluster baru, Anda dapat menggunakan alias kunci KMS alih-alih ARN untuk kunci enkripsi KMS.

Jika kunci enkripsi tidak ditentukan KmsKeyId dan jika `StorageEncrypted` parameternya benar, Amazon DocumentDB menggunakan kunci enkripsi default Anda.

- --preferred-backup-window—Opsional. Rentang waktu harian yang disukai selama pencadangan otomatis dibuat. Defaultnya adalah jendela 30 menit yang dipilih secara acak dari blok waktu 8 jam untuk masing-masing. Wilayah AWS
- --backup-retention-period—Opsional. Jumlah hari cadangan otomatis dipertahankan. Nilai default adalah 1.
- --storage-encrypted—Opsional. Mengkonfigurasikan apakah cluster dienkripsi atau tidak dienkripsi.

  `--no-storage-encrypted`menentukan bahwa cluster tidak dienkripsi.
- --subnet-ids—Opsional. Konfigurasikan ID subnet jaringan.

Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi Anda sendiri.

> **Note**
>
> Contoh berikut termasuk pembuatan kunci KMS tertentu. Untuk menggunakan kunci KMS default, jangan sertakan `--kms-key-id` parameter.

Untuk Linux, macOS, atau 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
```

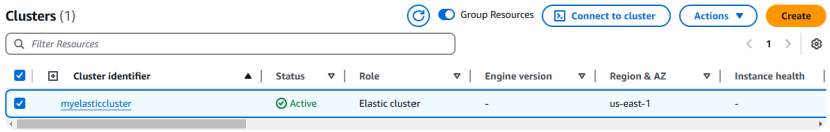
Untuk Windows:

```
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
```

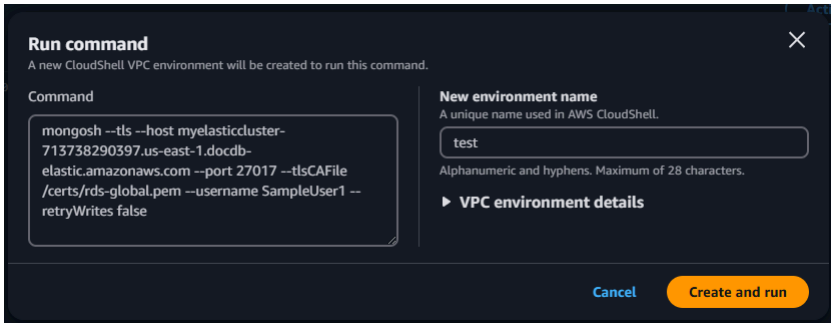
Langkah 2: Connect ke cluster elastis Anda

Connect ke cluster elastis Amazon DocumentDB Anda menggunakan. AWS CloudShell

1. Di konsol manajemen Amazon DocumentDB, di bawah Clusters, temukan cluster elastis yang Anda buat. Pilih klaster Anda dengan mengklik kotak centang di sebelahnya.



2. Pilih Connect to cluster (yang berada di sebelah menu dropdown Actions. Tombol ini diaktifkan hanya setelah Anda mengklik kotak centang di sebelah cluster Anda, dan status klaster ditampilkan sebagai Tersedia. Layar perintah CloudShell Jalankan muncul.
3. Di bidang Nama lingkungan baru, masukkan nama unik, seperti “uji” dan pilih Buat dan jalankan. Detail lingkungan VPC secara otomatis dikonfigurasi untuk database Amazon DocumentDB Anda.



4. Saat diminta, masukkan kata sandi yang Anda buat di Langkah 1: Buat cluster elastis Amazon DocumentDB (sub-langkah 5).



   Setelah Anda memasukkan kata sandi dan prompt Anda menjadi`direct: mongos] <env-name>>`, Anda berhasil terhubung ke cluster Amazon DocumentDB Anda

> **Note**
>
> Untuk informasi pemecahan masalah, lihat Memecahkan Masalah Amazon DocumentDB.

Langkah 3: Pecahkan koleksi, sisipkan, dan kueri data Anda

Cluster elastis menambahkan dukungan untuk sharding di Amazon DocumentDB. Sekarang Anda terhubung ke cluster Anda, Anda dapat menghancurkan cluster, menyisipkan data dan menjalankan beberapa kueri.

1. Untuk memecah koleksi, masukkan yang berikut ini:

```
sh.shardCollection("db.Employee1" , { "Employeeid" : "hashed" })
```

2. Untuk memasukkan satu dokumen, masukkan hal berikut ini:

```
db.Employee1.insertOne({"Employeeid":1, "Name":"Joe", "LastName": "Bruin", "level":
 1 })
```

Output berikut ditampilkan:

```
WriteResult({ "nInserted" : 1 })
```

3. Untuk membaca dokumen yang Anda tulis, masukkan `findOne()` perintah (mengembalikan satu dokumen):

```
db.Employee1.findOne()
```

Output berikut ditampilkan:

```
{
"_id" : ObjectId("61f344e0594fe1a1685a8151"),
"EmployeeID" : 1,
"Name" : "Joe",
"LastName" : "Bruin",
"level" : 1
}
```

4. Untuk melakukan beberapa pertanyaan lagi, pertimbangkan kasus penggunaan profil game. Pertama, masukkan beberapa entri ke dalam koleksi berjudul “Karyawan”. Masukkan yang berikut ini:

```
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 }
])
```

Output berikut ditampilkan:

```
{ acknowledged: true,
     insertedIds: {
        '0': ObjectId('679d02cd6b5a0581be78bcbd'),
        '1': ObjectId('679d02cd6b5a0581be78bcbe'),
```

```
        '2': ObjectId('679d02cd6b5a0581be78bcbf'),
        '3': ObjectId('679d02cd6b5a0581be78bcc0')
    }
}
```

5. Untuk mengembalikan semua dokumen dalam koleksi profil, masukkan perintah `find ()`:

```
db.Employee.find()
```

Data yang Anda masukkan pada langkah 4 ditampilkan.

6. Untuk menanyakan satu dokumen, sertakan filter (misalnya: “Katie”). Masukkan yang berikut ini:

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

Output berikut ditampilkan:

```
[

   {

     _id: ObjectId('679d02cd6b5a0581be78bcc0'),

     Employeeid: 4,

     name: 'Katie',

     lastname: 'Schaper',

     level: 3

   }

]
```

7. Untuk menemukan profil dan memodifikasinya, masukkan `findAndModify` perintah. Dalam contoh ini, karyawan “Matt” diberi level “14" yang lebih tinggi:

```
db.Employee.findAndModify({
    query: { "Employeeid" : 1, "name" : "Matt"},
    update: { "Employeeid" : 1, "name" : "Matt", "lastname" : "Winkle", "level" :
 14 }
```

```
})
```

Output berikut ditampilkan (perhatikan bahwa levelnya belum berubah):

```
{
   _id: ObjectId('679d02cd6b5a0581be78bcbd'),
   Employeeid: 1,
   name: 'Matt',
   lastname: 'Winkle',
   level: 12
}
```

8. Untuk memverifikasi peningkatan level, masukkan kueri berikut:

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

Output berikut ditampilkan:

```
[
   {
   _id: ObjectId('679d02cd6b5a0581be78bcbd'),
   Employeeid: 1,
   name: 'Matt',
   lastname: 'Winkle',
   level: 14
   }
]
```

Langkah 4: Jelajahi

Selamat! Anda telah berhasil menyelesaikan prosedur Memulai untuk cluster elastis Amazon DocumentDB.

Apa selanjutnya? Pelajari cara memanfaatkan basis data ini sepenuhnya dengan beberapa fitur populernya:

- Praktik terbaik cluster elastis Amazon DocumentDB
- Mengelola cluster elastis Amazon DocumentDB

> **Note**
>
> Cluster elastis yang Anda buat dari prosedur memulai ini akan terus bertambah biaya kecuali Anda menghapusnya. Untuk petunjuk arah, lihatMenghapus cluster elastis.

Praktik terbaik cluster elastis Amazon DocumentDB

Pelajari praktik terbaik untuk bekerja dengan klaster elastis Amazon DocumentDB. Semua praktik terbaik untuk cluster Amazon DocumentDB berbasis instans juga berlaku untuk cluster elastis. Bagian ini terus diperbarui saat praktik terbaik baru diidentifikasi.

Topik

- Memilih kunci pecahan
- Manajemen koneksi
- Koleksi unsharded
- Penskalaan cluster elastis
- Memantau cluster elastis

Memilih kunci pecahan

Daftar berikut menjelaskan pedoman untuk membuat kunci shard.

- Gunakan kunci hash yang didistribusikan secara merata untuk mendistribusikan data Anda di semua pecahan di cluster Anda (hindari tombol pintas).

- Gunakan kunci pecahan Anda di semua read/update permintaan/hapus untuk menghindari kueri pengumpulan pencar.
- Hindari kunci pecahan bersarang saat melakukan operasi read/update /delete.
- Saat membuat operasi batch, atur `ordered` ke false sehingga semua pecahan dapat berjalan secara paralel dan meningkatkan latensi.

Manajemen koneksi

Daftar berikut menjelaskan pedoman untuk mengelola koneksi Anda ke database Anda.

- Pantau jumlah koneksi Anda dan seberapa sering koneksi baru dibuka ditutup.
- Mendistribusikan koneksi Anda di semua subnet dalam konfigurasi aplikasi Anda. Jika cluster Anda dikonfigurasi dalam beberapa subnet tetapi Anda hanya menggunakan subset dari subnet, Anda mungkin terhambat pada koneksi maksimum Anda.

Koleksi unsharded

Berikut ini menjelaskan pedoman untuk koleksi unsharded.

- Saat bekerja dengan koleksi unsharded, untuk mendistribusikan beban, cobalah menyimpan koleksi unsharded yang sangat dimanfaatkan pada database yang berbeda. Cluster elastis Amazon DocumentDB menempatkan database di seluruh pecahan yang berbeda dan bersama-sama menempatkan koleksi unsharded untuk database yang sama pada pecahan yang sama.

Penskalaan cluster elastis

Daftar berikut menjelaskan pedoman untuk menskalakan cluster elastis Anda.

- Operasi penskalaan dapat menyebabkan periode singkat database intermiten dan kesalahan jaringan. Jika memungkinkan, hindari penskalaan selama jam sibuk. Cobalah untuk skala selama jendela pemeliharaan.
- Menskalakan kapasitas shard naik dan turun (mengubah jumlah vCPU per shard) untuk meningkatkan komputasi lebih disukai daripada menambah atau mengurangi shard-count karena lebih cepat dan memiliki durasi database intermiten dan kesalahan jaringan yang lebih pendek.

- Saat mengantisipasi pertumbuhan, lebih baik meningkatkan jumlah pecahan alih-alih menskalakan kapasitas pecahan. Ini memungkinkan Anda untuk menskalakan klaster Anda dengan meningkatkan kapasitas pecahan untuk skenario di mana Anda perlu menskalakan dengan cepat.
- Pantau kebijakan coba lagi sisi klien Anda dan coba lagi dengan backoff dan jitter eksponensial untuk menghindari kelebihan beban database Anda saat mendapatkan kesalahan saat melakukan penskalaan.
- Mengubah jumlah pecahan dapat memakan waktu karena data harus dipindahkan dengan aman dari satu pecahan ke pecahan lainnya. Untuk meminimalkan risiko kesalahan, kami sarankan untuk tidak menghentikan atau membuat perubahan lain pada status klaster saat perubahan sedang berlangsung.

Memantau cluster elastis

Daftar berikut menjelaskan pedoman untuk memantau kelompok elastis Anda.

- Lacak rasio puncak-ke-rata-rata metrik per shard Anda untuk menentukan apakah Anda mengarahkan lalu lintas yang tidak merata (memiliki hot- spot). key/hot Metrik utama untuk melacak rasio puncak-ke-rata-rata adalah:
  - `PrimaryInstanceCPUUtilization`
    - Ini dapat dipantau pada tingkat per-shard.
    - Pada tingkat cluster Anda dapat memantau rata-rata ke p99 miring.
  - `PrimaryInstanceFreeableMemory`
    - Ini dapat dipantau pada tingkat per-shard.
    - Pada tingkat cluster Anda dapat memantau rata-rata ke p99 miring.
  - `DatabaseCursorsMax`
    - Ini harus dipantau pada tingkat per-shard untuk menentukan kemiringan.
  - `Documents-Inserted/Updated/Returned/Deleted`
    - Ini harus dipantau pada tingkat per-shard untuk menentukan kemiringan.

Mengelola cluster elastis Amazon DocumentDB

Untuk mengelola cluster elastis Amazon DocumentDB, Anda harus memiliki kebijakan IAM dengan izin bidang kontrol Amazon DocumentDB yang sesuai. Izin ini memungkinkan Anda untuk membuat,

memodifikasi, dan menghapus cluster. DocumentDBFullAccess Kebijakan Amazon menyediakan semua izin yang diperlukan untuk mengelola klaster elastis Amazon DocumentDB.

Topik berikut menunjukkan cara melakukan berbagai tugas saat bekerja dengan cluster elastis Amazon DocumentDB.

Topik

- Memodifikasi konfigurasi cluster elastis
- Memantau cluster elastis
- Menghapus cluster elastis
- Mengelola snapshot cluster elastis
- Menghentikan dan memulai cluster elastis Amazon DocumentDB
- Mempertahankan cluster elastis Amazon DocumentDB

Memodifikasi konfigurasi cluster elastis

Pada bagian ini kami menjelaskan cara memodifikasi cluster elastis, menggunakan Konsol Manajemen AWS atau AWS CLI dengan instruksi berikut.

Penggunaan utama memodifikasi cluster adalah untuk menskalakan pecahan dengan meningkatkan atau mengurangi kapasitas komputasi shard count shard. and/or

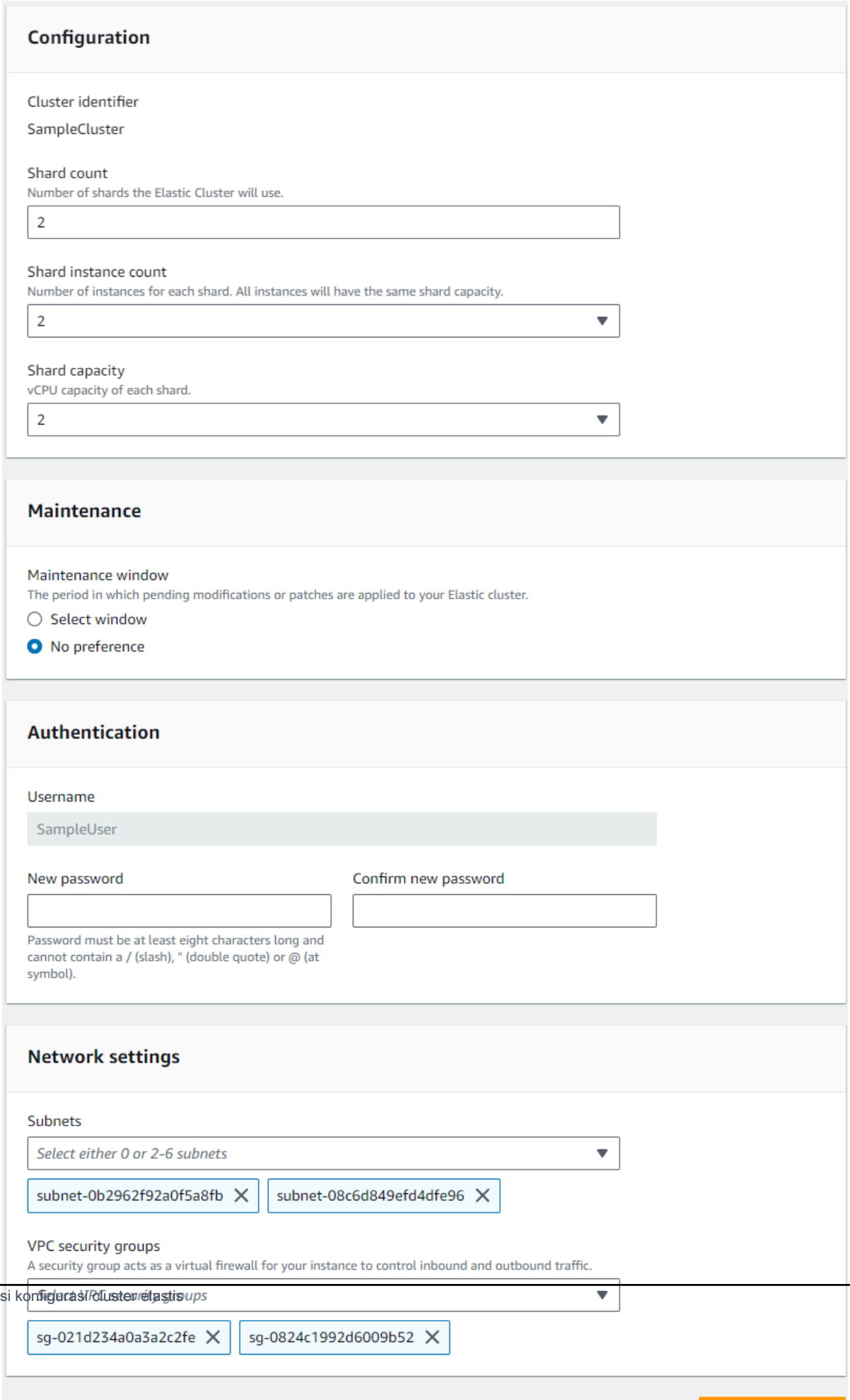
Using the Konsol Manajemen AWS

Untuk memodifikasi konfigurasi cluster elastis menggunakan Konsol Manajemen AWS:

1. Masuk ke Konsol Manajemen AWSdan buka konsol Amazon DocumentDB.
2. Pada panel navigasi, silakan pilih Klaster.

> **Tip**
>
> Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu di sudut kiri atas panel navigasi.

3. Pilih nama cluster yang ingin Anda ubah di kolom pengenal Cluster.
4. Pilih Ubah.
5. Edit bidang yang ingin diubah lalu pilih Modify 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**
>
> Atau, Anda dapat mengakses dialog Modify cluster dengan membuka halaman Clusters, mencentang kotak di sebelah cluster Anda, memilih Actions, lalu Modify.

Using the AWS CLI

Untuk memodifikasi konfigurasi cluster elastis menggunakan AWS CLI, gunakan `update-cluster` operasi dengan parameter berikut:

- **`--cluster-arn`**—Wajib. Pengidentifikasi ARN dari cluster yang ingin Anda modifikasi.
- **`--shard-capacity`**—Opsional. Jumlah vCPU yang ditetapkan untuk setiap pecahan. Maksimal adalah 64. Nilai yang diizinkan adalah 2, 4, 8, 16, 32, 64.
- **`--shard-count`**—Opsional. Jumlah pecahan yang ditugaskan ke cluster. Maksimal 32.
- **`--shard-instance`**-Hitung—opsional. Jumlah contoh replika yang diterapkan ke semua pecahan di cluster ini. Maksimal adalah 16.
- **`--auth-type`**—Opsional. Jenis otentikasi yang digunakan untuk menentukan di mana untuk mengambil password yang digunakan untuk mengakses cluster elastis. Jenis yang valid adalah `PLAIN_TEXT` atau`SECRET_ARN`.
- **`--admin-user-password`**—Opsional. Kata sandi yang terkait dengan pengguna admin.
- **`--vpc-security-group-ids`**—Opsional. Konfigurasikan daftar grup keamanan Amazon EC2 dan Amazon Virtual Private Cloud (VPC) untuk dikaitkan dengan klaster ini.
- **`--preferred-maintenance-window`**—Opsional. Konfigurasikan rentang waktu mingguan selama pemeliharaan sistem dapat terjadi, di Universal Coordinated Time (UTC)

  Formatnya adalah:`ddd:hh24:mi-ddd:hh24:mi`. Hari yang berlaku (ddd): Senin, Sel, Rabu, Kam, Jum, Sabtu, Minggu

  Defaultnya adalah jendela 30 menit yang dipilih secara acak dari blok waktu 8 jam untuk setiap Wilayah Amazon Web Services, yang terjadi pada hari acak dalam seminggu.

  Jendela minimal 30 menit.
- **`--subnet-ids`**—Opsional. Konfigurasikan ID subnet jaringan.

Dalam contoh berikut, ganti masing-masing *user input placeholder* dengan informasi Anda sendiri.

Untuk Linux, macOS, atau 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
```

Untuk 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
```

Untuk memantau status cluster elastis setelah modifikasi, lihat Memantau cluster elastis.

Memantau cluster elastis

Pada bagian ini, kami menjelaskan cara memantau cluster elastis Anda, menggunakan Konsol Manajemen AWS atau AWS CLI dengan instruksi berikut.

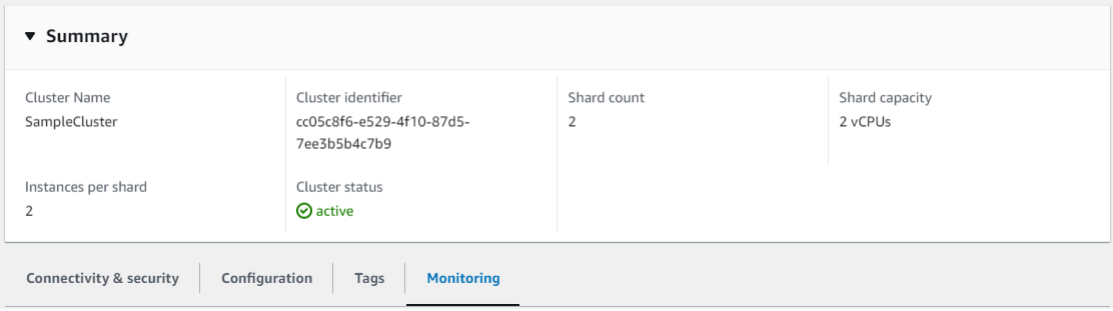
Using the Konsol Manajemen AWS

Untuk memantau konfigurasi cluster elastis menggunakan Konsol Manajemen AWS:

1. Masuk ke Konsol Manajemen AWSdan buka konsol Amazon DocumentDB.
2. Pada panel navigasi, silakan pilih Klaster.

> **Tip**
>
> Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu di sudut kiri atas panel navigasi.

3. Pilih nama cluster yang ingin Anda pantau di kolom pengenal Cluster.
4. Pilih tab Pemantauan.



Sejumlah grafik dari Amazon CloudWatch ditampilkan untuk kategori pemantauan berikut:

- Pemanfaatan Sumber Daya
- Throughput
- Operasi
- Sistem

Anda juga dapat mengakses Amazon CloudWatch melalui Konsol Manajemen AWS untuk mengatur lingkungan pemantauan Anda sendiri untuk cluster elastis Anda.

Using the AWS CLI

Untuk memantau konfigurasi cluster elastis tertentu menggunakan AWS CLI, gunakan `get-cluster` operasi dengan parameter berikut:

- **`--cluster-arn`**—Wajib. Pengidentifikasi ARN dari cluster yang Anda inginkan informasinya.

Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi Anda sendiri.

Untuk Linux, macOS, atau Unix:

```
aws docdb-elastic get-cluster \
    --cluster-arn arn:aws:docdb-elastic:us-west-2:123456789012:cluster:/68ffcdf8-
e3af-40a3-91e4-24736f2dacc9
```

Untuk Windows:

```
aws docdb-elastic get-cluster ^
    --cluster-arn arn:aws:docdb:-elastic:us-west-2:123456789012:cluster:/68ffcdf8-
e3af-40a3-91e4-24736f2dacc9
```

Output dari operasi ini terlihat seperti berikut:

```
"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",
        ...
 }
```

Untuk informasi selengkapnya, lihat `DescribeClusterSnapshot` di Referensi API Manajemen Sumber Daya Amazon DocumentDB.

Untuk melihat rincian semua cluster elastis menggunakan AWS CLI, gunakan `list-clusters` operasi dengan parameter berikut:

- **`--next-token`**—Opsional. Jika jumlah item output (`--max-results`) lebih sedikit dari jumlah total item yang dikembalikan oleh panggilan API yang mendasarinya, outputnya menyertakan perintah `NextToken` yang dapat Anda teruskan ke perintah berikutnya untuk mengambil set item berikutnya.

- **`--max-results`**—Opsional. Jumlah total item yang akan dikembalikan dalam output perintah. Jika lebih banyak hasil yang ada daripada `max-results` nilai yang ditentukan, token pagination (`next-token`) disertakan dalam respons sehingga hasil yang tersisa dapat diambil.
  - Default: 100
  - Minimum 20, maksimum 100

Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi Anda sendiri.

Untuk Linux, macOS, atau Unix:

```
aws docdb-elastic list-clusters \
    --next-token eyJNYXJrZXIiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAxfQ== \
    --max-results 2
```

Untuk Windows:

```
aws docdb-elastic list-clusters ^
    --next-token eyJNYXJrZXIiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAxfQ== ^
    --max-results 2
```

Output dari operasi ini terlihat seperti berikut:

```
{
   "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"
```

```
        }
    ]
}
```

Menghapus cluster elastis

Pada bagian ini kami menjelaskan cara menghapus cluster elastis, menggunakan Konsol Manajemen AWS atau AWS CLI dengan instruksi berikut.

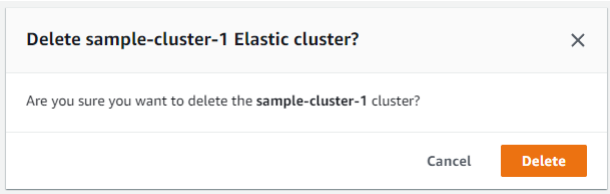
Using the Konsol Manajemen AWS

Untuk menghapus konfigurasi cluster elastis menggunakan Konsol Manajemen AWS:

1. Masuk ke Konsol Manajemen AWSdan buka konsol Amazon DocumentDB.
2. Pada panel navigasi, silakan pilih Klaster.

> **Tip**
>
> Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu di sudut kiri atas panel navigasi.

3. Dalam tabel daftar cluster, pilih kotak centang di sebelah kiri nama cluster yang ingin Anda hapus lalu pilih Tindakan. Dari menu pilihan menurun, pilih Hapus.
4. Di cluster elastis Delete “cluster-name”? kotak dialog, pilih Hapus.



Dibutuhkan beberapa menit untuk menghapus klaster. Untuk memantau status klaster, lihat Memantau Status Cluster Amazon DocumentDB.

Using the AWS CLI

Untuk menghapus cluster elastis menggunakan AWS CLI, gunakan `delete-cluster` operasi dengan parameter berikut::

- **`--cluster-arn`**—Wajib. Pengidentifikasi ARN dari cluster yang ingin Anda hapus.

- **`--no-skip-final-backup`**—Opsional. Jika Anda menginginkan cadangan akhir, Anda harus menyertakan parameter ini dengan nama untuk cadangan akhir. Anda harus menyertakan `--final-backup-identifier` atau `--skip-final-backup`.
- **`--skip-final-backup`**—Opsional. Gunakan parameter ini hanya jika Anda tidak ingin mengambil cadangan akhir sebelum menghapus klaster Anda. Pengaturan default adalah untuk mengambil snapshot akhir.

Contoh AWS CLI kode berikut menghapus cluster dengan ARN arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-cluster dengan cadangan akhir.

Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi Anda sendiri..

Untuk Linux, macOS, atau 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
```

Untuk 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
```

Contoh AWS CLI kode berikut menghapus cluster dengan ARN arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-cluster tanpa mengambil cadangan akhir.

Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi Anda sendiri.

Untuk Linux, macOS, atau Unix:

```
aws docdb-elastic delete-cluster \
```

```
    --cluster-arn arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-
cluster \
    --skip-final-backup \
```

Untuk Windows:

```
aws docdb-elastic delete-cluster ^
    --cluster-arn arn:aws:docdb:us-west-2:123456789012:sharded-cluster:sample-
cluster ^
    --skip-final-backup ^
```

Output dari `delete-cluster` operasi adalah tampilan cluster yang Anda hapus.

Dibutuhkan beberapa menit untuk menghapus klaster. Untuk memantau status klaster, lihat Memantau Status Cluster Amazon DocumentDB.

Mengelola snapshot cluster elastis

Snapshot manual dapat diambil setelah cluster elastis dibuat. Pencadangan otomatis dibuat saat snapshot cluster elastis dibuat.

> **Note**
>
> Cluster elastis Anda harus dalam `Available` keadaan agar snapshot manual diambil.

Bagian ini menjelaskan bagaimana Anda dapat membuat, melihat, memulihkan dari, dan menghapus snapshot cluster elastis.

Topik berikut menunjukkan cara melakukan berbagai tugas saat bekerja dengan snapshot cluster elastis Amazon DocumentDB.

Topik

- Membuat snapshot cluster elastis manual
- Melihat snapshot cluster elastis
- Memulihkan cluster elastis dari snapshot
- Menyalin snapshot cluster elastis

- Menghapus snapshot cluster elastis
- Mengelola cadangan otomatis snapshot cluster elastis

Membuat snapshot cluster elastis manual

Pada bagian ini kami menjelaskan cara membuat snapshot cluster elastis manual, menggunakan Konsol Manajemen AWS atau AWS CLI dengan instruksi berikut.

Using the Konsol Manajemen AWS

Untuk membuat snapshot cluster elastis manual menggunakan: Konsol Manajemen AWS

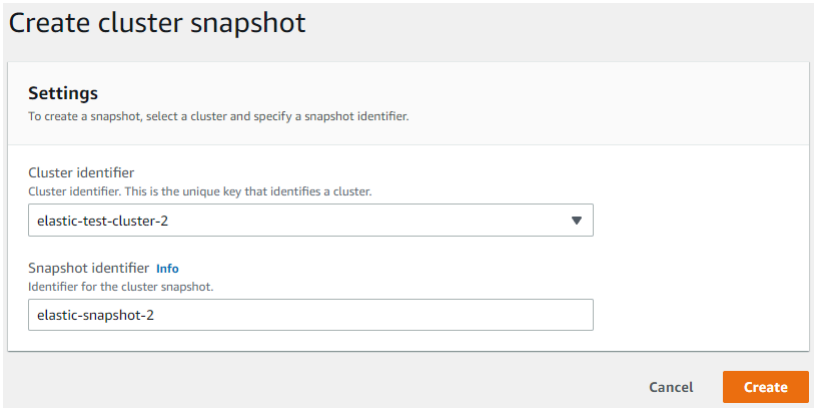
1. Masuk ke Konsol Manajemen AWSdan buka konsol Amazon DocumentDB.
2. Di panel navigasi, pilih Snapshot.

> **Tip**
>
> Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu di sudut kiri atas panel navigasi.

3. Pada halaman Snapshot, pilih Buat.
4. Pada halaman Create cluster snapshot, di kolom Cluster identifier, pilih cluster elastis Anda dari daftar drop-down.

   Di bidang pengenal Snapshot, masukkan pengenal unik untuk cluster elastis Anda.

   Pilih Buat.



> **Note**
>
> Atau, Anda dapat mengakses dialog Buat snapshot cluster dengan membuka halaman Clusters, mencentang kotak di sebelah cluster Anda, lalu memilih Tindakan, lalu Ambil snapshot.

Snapshot cluster elastis Anda sekarang sedang disediakan. Ini bisa memakan waktu hingga beberapa menit untuk menyelesaikannya. Anda dapat melihat dan memulihkan dari snapshot Anda ketika status ditampilkan seperti `Available` dalam daftar Snapshots.

Using the AWS CLI

Untuk membuat snapshot cluster elastis manual menggunakan AWS CLI, gunakan `create-cluster-snapshot` operasi dengan parameter berikut:

- **`--snapshot-name`**—Wajib. Nama snapshot cluster yang ingin Anda buat.
- **`--cluster-arn`**—Wajib. Pengidentifikasi ARN dari cluster yang ingin Anda buat snapshot.

Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi Anda sendiri.

Untuk Linux, macOS, atau 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
```

Untuk 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
```

Melihat snapshot cluster elastis

Pada bagian ini kami menjelaskan cara melihat informasi snapshot cluster elastis, menggunakan Konsol Manajemen AWS atau AWS CLI dengan instruksi berikut.

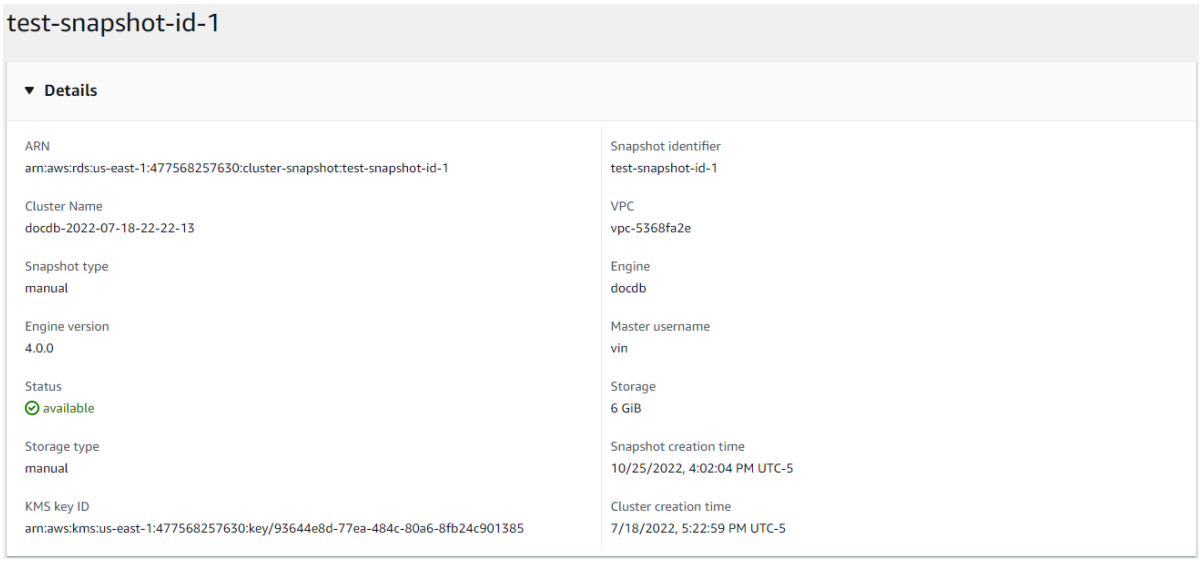
Using the Konsol Manajemen AWS

Untuk melihat informasi tentang snapshot cluster elastis tertentu menggunakan: Konsol Manajemen AWS

1. Masuk ke Konsol Manajemen AWSdan buka konsol Amazon DocumentDB.
2. Di panel navigasi, pilih Snapshot.

> **Tip**
>
> Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu di sudut kiri atas panel navigasi.

3. Pada halaman Snapshots, pilih snapshot Anda dari daftar dengan mengklik nama di kolom pengenal Snapshot.
4. Lihat informasi snapshot Anda di Detail.



Using the AWS CLI

Untuk melihat informasi tentang snapshot cluster elastis tertentu menggunakan AWS CLI, gunakan `get-cluster-snapshot` operasi dengan parameter berikut:

- **`--snapshot-arn`**—Wajib. Pengidentifikasi ARN dari snapshot yang Anda inginkan informasinya.

Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi Anda sendiri.

Untuk Linux, macOS, atau Unix:

```
aws docdb-elastic get-cluster-snapshot \
    --snapshot-arn sampleResourceName
```

Untuk Windows:

```
aws docdb-elastic get-cluster-snapshot ^
    --snapshot-arn sampleResourceName
```

Untuk melihat informasi tentang snapshot cluster elastis tertentu menggunakan AWS CLI, gunakan `get-cluster-snapshot` operasi dengan parameter berikut:

- **`--snapshot-arn`**—Wajib. Pengidentifikasi ARN dari snapshot yang Anda inginkan informasinya.

Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi Anda sendiri.

Untuk Linux, macOS, atau Unix:

```
aws docdb-elastic get-cluster-snapshot \
    --snapshot-arn sampleResourceName
```

Untuk Windows:

```
aws docdb-elastic get-cluster-snapshot ^
    --snapshot-arn sampleResourceName
```

Untuk melihat informasi tentang semua snapshot cluster elastis menggunakan AWS CLI, gunakan `list-cluster-snapshots` operasi dengan parameter berikut:

- **`--snapshot-type`**—Opsional. Jenis snapshot klaster untuk dikembalikan. Anda dapat menentukan salah satu nilai berikut:
  - `automated`- Kembalikan semua snapshot cluster yang Amazon DocumentDB telah secara otomatis dibuat untuk akun Anda. AWS
  - `manual`- Kembalikan semua snapshot cluster yang telah Anda buat secara manual untuk AWS akun Anda.
  - `shared`- Kembalikan semua snapshot cluster manual yang telah dibagikan ke AWS akun Anda.
  - `public` - Mengembalikan semua snapshot klaster yang telah ditandai sebagai publik.
- **`--next-token`**—Opsional. Token pagination opsional yang disediakan oleh permintaan sebelumnya. Jika parameter ini ditentukan, respons hanya mencakup catatan di luar token ini, hingga nilai yang ditentukan oleh`max-results`.
- **`--max-results`**—Opsional. Jumlah maksimum hasil untuk dimasukkan dalam respons. Jika lebih banyak hasil yang ada daripada `max-results` nilai yang ditentukan, token pagination (`next-token`) disertakan dalam respons sehingga hasil yang tersisa dapat diambil.
  - Default: 100
  - Minimum 20, maksimum 100

Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi Anda sendiri.

Untuk Linux, macOS, atau Unix:

```
aws docdb-elastic list-cluster-snapshots \
    --snapshot-type  value \
    --next-token  value \
    --max-results 50
```

Untuk Windows:

```
aws docdb-elastic list-cluster-snapshots ^
    --snapshot-type  value ^
    --next-token  value ^
    --max-results 50
```

Memulihkan cluster elastis dari snapshot

Pada bagian ini kami menjelaskan cara mengembalikan cluster elastis dari snapshot, menggunakan Konsol Manajemen AWS atau AWS CLI dengan instruksi berikut.

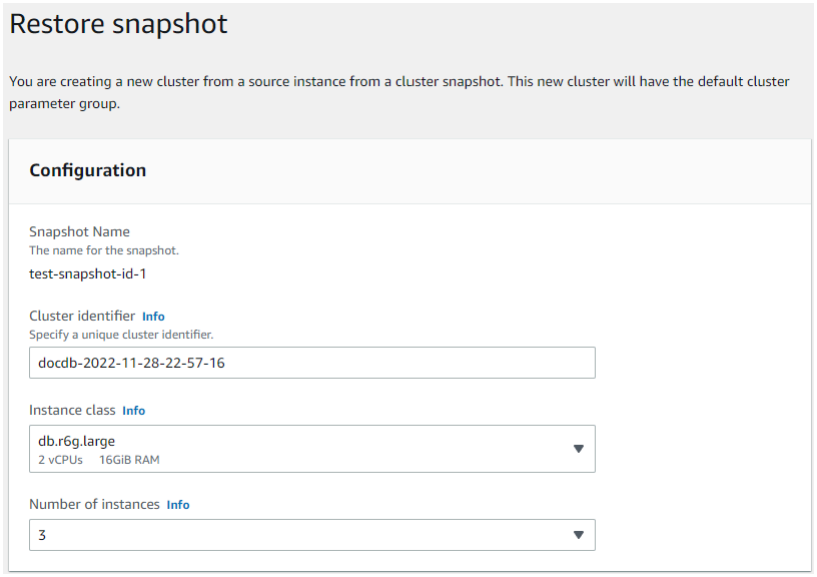
Using the Konsol Manajemen AWS

Untuk mengembalikan cluster elastis dari snapshot menggunakan: Konsol Manajemen AWS

1. Masuk ke Konsol Manajemen AWSdan buka konsol Amazon DocumentDB.
2. Di panel navigasi, pilih Snapshot.

> **Tip**
>
> Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu di sudut kiri atas panel navigasi.

3. Pilih tombol di sebelah kiri snapshot, yang ingin Anda gunakan untuk memulihkan cluster, di kolom pengenal Snapshot.
4. Pilih Tindakan, lalu Pulihkan.



5. Pada halaman Restore snapshot, masukkan nama untuk cluster baru di kolom Cluster identifier.

> **Note**
>
> Untuk restorasi snapshot manual, Anda harus membuat cluster baru.

6. Di bidang Virtual Private Cloud (VPC), pilih VPC dari daftar drop-down.
7. Untuk grup keamanan Subnet dan VPC, Anda dapat menggunakan default atau memilih tiga subnet pilihan Anda dan hingga tiga grup keamanan VPC (minimal satu).
8. Jika Anda puas dengan konfigurasi klaster, pilih Pulihkan klaster dan tunggu sementara klaster Anda dipulihkan.

Using the AWS CLI

Untuk mengembalikan cluster elastis dari snapshot menggunakan AWS CLI, gunakan `restore-cluster-from-snapshot` operasi dengan parameter berikut:

- **`--cluster-name`**—Wajib. Nama cluster elastis saat ini seperti yang dimasukkan selama pembuatan atau terakhir dimodifikasi.
- **`--snapshot-arn`**—Wajib. Pengidentifikasi ARN dari snapshot yang digunakan untuk memulihkan cluster.
- **`--vpc-security-group-ids`**—Opsional. Satu atau beberapa grup keamanan Amazon EC2 dan Amazon Virtual Private Cloud (VPC) untuk diasosiasikan dengan cluster.
- **`--kms-key-id`**—Opsional. Konfigurasikan pengidentifikasi kunci KMS untuk cluster terenkripsi.

  Pengidentifikasi kunci KMS adalah Amazon Resource Name (ARN) untuk kunci enkripsi. AWS KMS Jika Anda membuat cluster menggunakan akun Amazon Web Services yang sama yang memiliki kunci enkripsi KMS yang digunakan untuk mengenkripsi cluster baru, Anda dapat menggunakan alias kunci KMS alih-alih ARN untuk kunci enkripsi KMS.

  Jika kunci enkripsi tidak ditentukan KmsKeyId dan jika `StorageEncrypted` parameternya benar, Amazon DocumentDB menggunakan kunci enkripsi default Anda.
- **`--subnet-ids`**—Opsional. Id subnet jaringan.

Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi Anda sendiri..

Untuk Linux, macOS, atau 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
```

Untuk 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
```

Menyalin snapshot cluster elastis

Di Amazon DocumentDB, Anda dapat menyalin snapshot cluster elastis manual dan otomatis dalam wilayah yang sama dan dalam akun yang sama. Pada bagian ini kami menjelaskan cara menyalin snapshot cluster elastis, menggunakan Konsol Manajemen AWS atau AWS CLI.

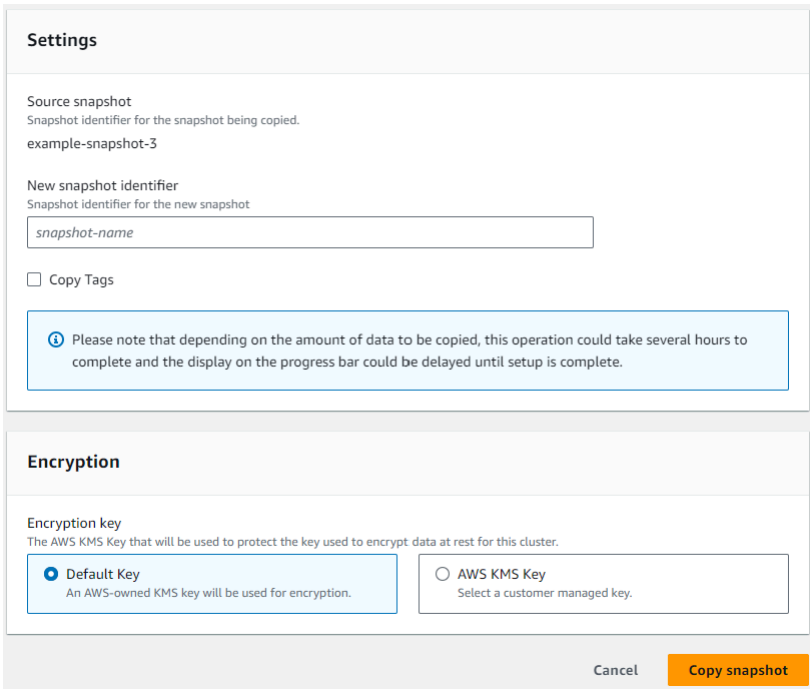
Using the Konsol Manajemen AWS

Untuk menyalin snapshot cluster elastis menggunakan: Konsol Manajemen AWS

1. Masuk ke Konsol Manajemen AWSdan buka konsol Amazon DocumentDB.
2. Di panel navigasi, pilih Snapshot.

   > **Tip**
   >
   > Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu di sudut kiri atas panel navigasi.

3. Pilih tombol di sebelah kiri snapshot, yang ingin Anda salin, di kolom pengenal Snapshot.
4. Pilih Tindakan, lalu Salin.



5. Untuk pengenal snapshot baru, masukkan nama snapshot baru.
6. Untuk Copy Tags, centang kotak jika Anda ingin menyalin semua tag dari snapshot cluster elastis sumber ke snapshot cluster elastis target.
7. Untuk Enkripsi, pilih kunci AWS KMS default atau kunci KMS yang Anda pilih. Opsi kedua memungkinkan Anda untuk memilih kunci KMS yang sudah Anda buat atau memungkinkan Anda membuat yang baru.
8. Pilih Salin snapshot saat selesai.

Using the AWS CLI

Untuk menyalin snapshot cluster elastis menggunakan AWS CLI, gunakan `copy-cluster-snapshot` operasi dengan parameter berikut:

- **`--source-db-cluster-snapshot-identifier`**—Wajib. Pengidentifikasi snapshot cluster elastis yang ada sedang disalin. Snapshot cluster elastis harus ada dan berada dalam keadaan tersedia. Parameter ini tidak peka huruf besar kecil.
- **`--target-db-cluster-snapshot-identifier`**—Wajib. Pengidentifikasi snapshot cluster elastis baru untuk dibuat dari snapshot cluster yang ada. Parameter ini tidak peka huruf besar kecil.

  Kendala nama snapshot target:

- Tidak dapat berupa nama snapshot yang sudah ada.
- Panjangnya adalah [1—63] huruf, angka, atau tanda hubung.
- Karakter pertama harus berupa huruf.
- Tidak dapat diakhiri dengan tanda hubung atau berisi dua tanda hubung berurutan.

Dalam contoh berikut, ganti masing-masing *user input placeholder* dengan informasi Anda sendiri.

Untuk Linux, macOS, atau Unix:

```
aws docdb-elastic copy-cluster-snapshot \
     --source-cluster-snapshot-arn <sample ARN> \
     --target-cluster-snapshot-name my-target-copied-snapshot
```

Untuk Windows:

```
aws docdb-elastic copy-cluster-snapshot ^
     --source-cluster-snapshot-arn <sample ARN> ^
     --target-cluster-snapshot-name my-target-copied-snapshot
```

Menghapus snapshot cluster elastis

Pada bagian ini kami menjelaskan cara menghapus snapshot cluster elastis, menggunakan Konsol Manajemen AWS atau AWS CLI.

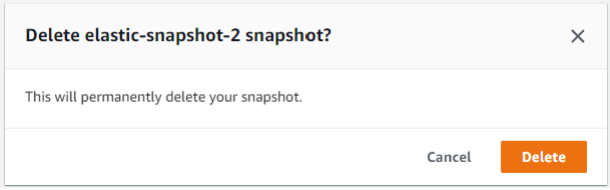
Using the Konsol Manajemen AWS

Untuk mengembalikan cluster elastis dari snapshot menggunakan: Konsol Manajemen AWS

1. Masuk ke Konsol Manajemen AWSdan buka konsol Amazon DocumentDB.
2. Di panel navigasi, pilih Snapshot.

> **Tip**
>
> Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu di sudut kiri atas panel navigasi.

3. Pilih tombol di sebelah kiri snapshot, yang ingin Anda gunakan untuk memulihkan cluster, di kolom pengenal Snapshot.
4. Pilih Actions (Tindakan), lalu Delete (Hapus).



5. Pada dialog Hapus snapshot “snapshot-name”, pilih Hapus.

Using the AWS CLI

Untuk menghapus snapshot cluster elastis menggunakan AWS CLI, gunakan `delete-cluster-snapshot` operasi dengan parameter berikut:

- **`--snapshot-arn`**—Wajib. Pengidentifikasi ARN dari snapshot yang digunakan untuk memulihkan cluster.

Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi Anda sendiri..

Untuk Linux, macOS, atau Unix:

```
aws docdb-elastic delete-cluster-snapshot \
     --snapshot-arn sampleResourceName
```

Untuk Windows:

```
aws docdb-elastic delete-cluster-snapshot ^
     --snapshot-arn sampleResourceName
```

Mengelola cadangan otomatis snapshot cluster elastis

Amazon DocumentDB mengambil snapshot harian dari cluster elastis Anda. Anda dapat menentukan jendela cadangan pilihan dan periode retensi cadangan dalam konfigurasi snapshot cluster elastis baru atau yang sudah ada. Pada bagian ini kami menjelaskan cara mengatur parameter cadangan otomatis dalam snapshot cluster elastis, menggunakan Konsol Manajemen AWS atau AWS CLI.

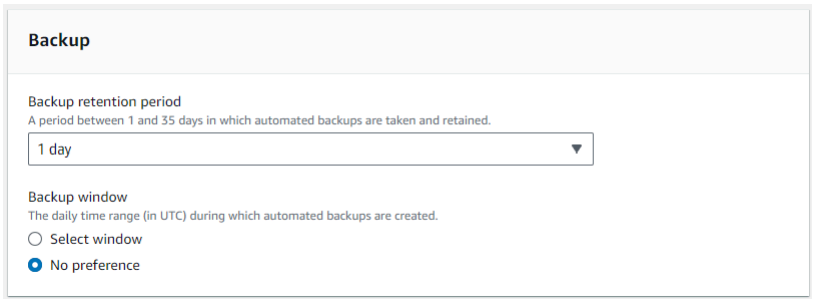
Using the Konsol Manajemen AWS

Untuk mengatur cadangan otomatis untuk snapshot cluster elastis baru menggunakan: Konsol Manajemen AWS

1. Masuk ke Konsol Manajemen AWSdan buka konsol Amazon DocumentDB.
2. Pada panel navigasi, silakan pilih Klaster.

> **Tip**
>
> Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu di sudut kiri atas panel navigasi.

3. Pilih tombol di sebelah kiri cluster, yang ingin Anda ubah pengaturan cadangan, di kolom pengenal Cluster.
4. Pilih Tindakan, lalu Ubah.
5. Di bagian Backup, edit bidang sesuai dengan kebutuhan cadangan Anda.



   a. Periode retensi cadangan—Dalam daftar, pilih jumlah hari untuk menyimpan backup otomatis klaster ini sebelum menghapusnya.
   b. Jendela backup—Atur waktu harian dan durasi yang akan digunakan oleh Amazon DocumentDB untuk membuat backup dari klaster ini.

      i. Pilih Pilih jendela jika Anda ingin mengonfigurasi waktu dan durasi saat cadangan dibuat.

         Waktu mulai—Pada daftar pertama, pilih jam waktu mulai (UTC) untuk memulai pencadangan otomatis. Dalam daftar kedua, pilih menit dari jam pilihan Anda untuk memulai backup otomatis.

         Durasi—Dalam daftar, pilih jumlah jam yang akan dialokasikan untuk membuat cadangan otomatis.

ii. Pilih Tidak ada preferensi jika Anda ingin Amazon DocumentDB memilih waktu dan durasi saat backup dibuat.

6. Pilih Modify cluster saat selesai.

Using the AWS CLI

Untuk mengatur cadangan otomatis untuk snapshot cluster elastis baru menggunakan AWS CLI, gunakan `create-cluster-snapshot` operasi dengan parameter berikut:

- **`--preferred-backup-window`**—Opsional. Rentang waktu harian yang disukai selama pencadangan otomatis dibuat. Defaultnya adalah jendela 30 menit yang dipilih secara acak dari blok waktu 8 jam untuk masing-masing. Wilayah AWS

  Batas:

  - Harus dalam format `hh24:mi-hh24:mi`.
  - Harus dalam Waktu Universal Terkoordinasi (UTC).
  - Tidak boleh bertentangan dengan jendela pemeliharaan yang diinginkan.
  - Harus setidaknya 30 menit.
- **`--backup-retention-period`**—Opsional. Jumlah hari cadangan otomatis dipertahankan. Nilai default adalah 1.

  Batas:

  - Harus menentukan nilai minimum 1.
  - Rentang adalah dari 1 hingga 35.

> **Note**
>
> Pencadangan otomatis hanya dilakukan saat klaster berada dalam keadaan 'aktif'.

> **Note**
>
> Anda juga dapat memodifikasi `preferred-backup-window` dan `backup-retention-period` parameter cluster elastis yang ada menggunakan `aws docdb-elastic update-cluster` perintah.

Dalam contoh berikut, ganti masing-masing *user input placeholder* dengan informasi Anda sendiri.

`create-cluster`Contoh berikut membuat *sample-cluster* cluster elastis Amazon DocumentDB dengan periode retensi untuk pencadangan otomatis hari, dan *7* jendela cadangan pilihan. *18:00-18:30 UTC*

Untuk Linux, macOS, atau 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
```

Untuk 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
```

Menghentikan dan memulai cluster elastis Amazon DocumentDB

Menghentikan dan memulai cluster elastis Amazon DocumentDB dapat membantu Anda mengelola biaya untuk lingkungan pengembangan dan pengujian. Alih-alih membuat dan menghapus cluster elastis setiap kali Anda menggunakan Amazon DocumentDB, Anda dapat menghentikan sementara klaster Anda saat tidak diperlukan. Anda kemudian dapat memulainya lagi ketika Anda melanjutkan pengujian Anda.

Topik

- Ikhtisar menghentikan dan memulai cluster elastis
- Operasi yang dapat Anda lakukan pada cluster elastis yang berhenti

Ikhtisar menghentikan dan memulai cluster elastis

Selama periode di mana Anda tidak memerlukan cluster elastis Amazon DocumentDB, Anda dapat menghentikan cluster. Anda lalu dapat memulai klaster lagi kapan saja Anda memerlukannya. Memulai dan menghentikan menyederhanakan proses penyiapan dan pembongkaran untuk cluster elastis yang digunakan untuk pengembangan, pengujian, atau aktivitas serupa yang tidak memerlukan ketersediaan berkelanjutan. Anda dapat berhenti dan memulai cluster elastis menggunakan Konsol Manajemen AWS atau AWS CLI dengan satu tindakan.

Sementara cluster elastis Anda dihentikan, volume penyimpanan cluster tetap tidak berubah. Anda hanya dikenakan biaya untuk penyimpanan, snapshot manual, dan penyimpanan cadangan otomatis dalam jendela penyimpanan yang Anda tentukan. Amazon DocumentDB secara otomatis memulai cluster elastis Anda setelah tujuh hari sehingga tidak ketinggalan pembaruan pemeliharaan yang diperlukan. Ketika cluster Anda dimulai setelah tujuh hari, Anda akan mulai dikenakan biaya untuk penggunaan cluster elastis lagi. Saat klaster dihentikan, Anda tidak dapat melakukan kueri volume penyimpanan karena kueri mengharuskan klaster dalam status tersedia.

Ketika cluster elastis Amazon DocumentDB dihentikan, cluster tidak dapat dimodifikasi dengan cara apa pun. Ini termasuk menghapus cluster.

Using the Konsol Manajemen AWS

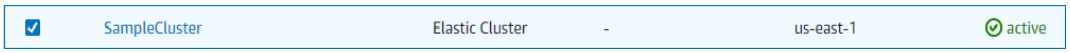
Prosedur berikut menunjukkan kepada Anda cara menghentikan gugus elastis dalam keadaan tersedia, atau memulai gugus elastis yang berhenti.

Untuk menghentikan atau memulai cluster elastis Amazon DocumentDB

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Klaster.

> **Tip**
>
> Jika panel navigasi tidak terlihat di sisi kiri layar Anda, pilih ikon menu (≡) di sudut kiri atas halaman.

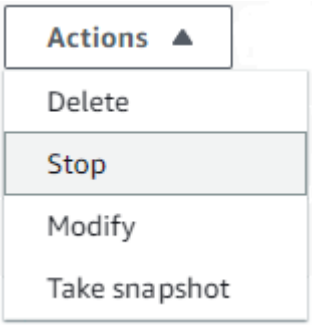
3. Dalam daftar klaster, pilih tombol di sebelah kiri pada klaster yang ingin Anda mulai atau tingkatkan.



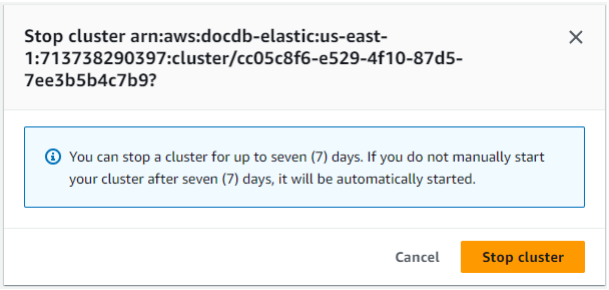
4. Pilih Tindakan, lalu pilih tindakan yang ingin Anda lakukan pada klaster.

   - Jika Anda ingin menghentikan klaster dan klaster tersedia:

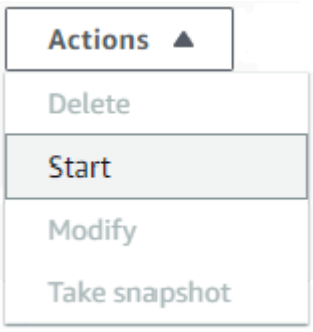
     a. Pilih Berhenti.



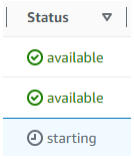
     b. Pada dialog konfirmasi, konfirmasikan bahwa Anda ingin menghentikan cluster elastis dengan memilih Stop cluster, atau agar cluster tetap berjalan, pilih Batal.



   - Jika Anda ingin memulai klaster, dan klaster berhenti, pilih Mulai.



5. Pantau status cluster elastis. Jika Anda memulai cluster, Anda dapat melanjutkan menggunakan cluster saat cluster tersedia. Untuk informasi selengkapnya, lihat Menentukan status klaster.



Using the AWS CLI

Contoh kode berikut menunjukkan cara menghentikan gugus elastis dalam keadaan aktif atau tersedia, atau memulai gugus elastis yang berhenti.

Untuk menghentikan cluster elastis menggunakan AWS CLI, gunakan `stop-cluster` operasi. Untuk memulai klaster yang telah dihentikan, gunakan operasi `start-cluster`. Kedua operasi menggunakan parameter `--cluster-arn`.

Parameter:

- **`--cluster-arn`**—Diperlukan. Pengidentifikasi ARN dari cluster elastis yang ingin Anda hentikan atau mulai.

Example— Untuk menghentikan cluster elastis menggunakan AWS CLI

Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi Anda sendiri.

Kode berikut menghentikan cluster elastis dengan ARN dari. `arn:aws:docdb-elastic:us-east-1:477568257630:cluster/b9f1d489-6c3e-4764-bb42-da62ceb7bda2`

> **Note**
>
> Cluster elastis harus dalam keadaan aktif atau tersedia.

Untuk Linux, macOS, atau Unix:

```
aws docdb-elastic stop-cluster \
   --cluster-arn arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2
```

Untuk Windows:

```
aws docdb-elastic stop-cluster ^
   --cluster-arn arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2
```

Example— Untuk memulai cluster elastis menggunakan AWS CLI

Dalam contoh berikut, ganti masing-masing *user input placeholder* dengan informasi Anda sendiri.

Kode berikut memulai cluster elastis dengan ARN dari. `arn:aws:docdb-elastic:us-east-1:477568257630:cluster/b9f1d489-6c3e-4764-bb42-da62ceb7bda2`

> **Note**
>
> Cluster elastis saat ini harus dihentikan.

Untuk Linux, macOS, atau Unix:

```
aws docdb-elastic start-cluster \
   --cluster-arn arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2
```

Untuk Windows:

```
aws docdb-elastic start-cluster ^
```

```
    --cluster-arn arn:aws:docdb-elastic:us-east-1:477568257630:cluster/
b9f1d489-6c3e-4764-bb42-da62ceb7bda2
```

Operasi yang dapat Anda lakukan pada cluster elastis yang berhenti

Anda tidak dapat mengubah konfigurasi cluster elastis Amazon DocumentDB saat cluster dihentikan. Anda harus memulai klaster sebelum melakukan tindakan administratif tersebut.

Amazon DocumentDB menerapkan pemeliharaan terjadwal apa pun ke cluster elastis Anda yang berhenti hanya setelah dimulai lagi. Setelah tujuh hari, Amazon DocumentDB secara otomatis memulai cluster elastis yang berhenti sehingga tidak tertinggal terlalu jauh dalam status pemeliharaannya. Saat cluster elastis dimulai ulang, Anda akan mulai dikenakan biaya untuk pecahan di cluster lagi.

Sementara cluster elastis dihentikan, Amazon DocumentDB tidak melakukan pencadangan otomatis apa pun juga tidak memperpanjang periode retensi cadangan.

Mempertahankan cluster elastis Amazon DocumentDB

Topik

- Melihat tindakan pemeliharaan cluster elastis yang tertunda
- Pembaruan mesin cluster elastis
- Pembaruan sistem operasi cluster elastis

Secara berkala, Amazon DocumentDB melakukan pemeliharaan pada sumber daya cluster elastis Amazon DocumentDB. Pemeliharaan paling sering melibatkan pembaruan ke mesin database (pemeliharaan cluster elastis) atau sistem operasi dasar cluster elastis (pembaruan OS). Pembaruan mesin database adalah tambalan yang diperlukan dan mencakup perbaikan keamanan, perbaikan bug, dan penyempurnaan pada mesin database. Meskipun sebagian besar tambalan sistem operasi bersifat opsional, jika Anda tidak menerapkannya untuk sementara waktu, tambalan mungkin diperlukan dan diterapkan secara otomatis untuk menjaga postur keamanan Anda. Jadi, kami menyarankan Anda menerapkan pembaruan sistem operasi ke cluster elastis Amazon DocumentDB Anda segera setelah tersedia.

Tambalan mesin database mengharuskan Anda membuat cluster elastis Amazon DocumentDB offline untuk waktu yang singkat. Setelah tersedia, tambalan ini secara otomatis dijadwalkan untuk

diterapkan selama jendela pemeliharaan terjadwal yang akan datang dari cluster elastis Amazon DocumentDB Anda.

Cluster elastis memiliki jendela perawatan masing-masing. Modifikasi cluster elastis yang telah Anda pilih untuk tidak segera diterapkan, diterapkan selama jendela pemeliharaan. Secara default, saat Anda membuat cluster elastis, Amazon DocumentDB menetapkan jendela pemeliharaan untuk cluster elastis Anda. Anda dapat memilih jendela pemeliharaan saat membuat cluster elastis. Anda juga dapat mengubah jendela pemeliharaan kapan saja agar sesuai dengan jadwal atau praktik bisnis Anda. Umumnya disarankan untuk memilih jendela pemeliharaan yang meminimalkan dampak pemeliharaan pada aplikasi Anda (misalnya, pada malam hari atau akhir pekan).

Melihat tindakan pemeliharaan cluster elastis yang tertunda

Anda dapat melihat apakah pembaruan pemeliharaan tersedia untuk cluster elastis Anda dengan menggunakan AWS CLI.

Jika pembaruan tersedia, Anda dapat melakukan salah satu hal berikut:

- Tunda tindakan pemeliharaan yang saat ini dijadwalkan untuk jendela pemeliharaan berikutnya (hanya untuk patch OS).
- Terapkan tindakan pemeliharaan segera.
- Jadwalkan tindakan pemeliharaan untuk dimulai selama periode pemeliharaan berikutnya.
- Jadwalkan tindakan pemeliharaan untuk memulai selama jendela penerapan yang Anda pilih.

Jendela pemeliharaan menentukan kapan operasi yang tertunda dimulai, tetapi tidak membatasi total waktu pelaksanaan operasi ini.

Gunakan AWS CLI operasi berikut untuk menentukan tindakan pemeliharaan apa yang tertunda. Daftar semua tindakan pemeliharaan yang tertunda:

```
aws docdb-elastic list-pending-maintenance-actions
```

Output dari operasi ini terlihat seperti berikut (format JSON):

```
{
'ResourcePendingMaintenanceActions': [
    {
        'ResourceArn': 'string-arn',
```

```
        'PendingMaintenanceActionDetails': [
            {
                'Action': 'ENGINE_UPDATE',
                'AutoAppliedAfterDate': 'string',
                'ForcedApplyDate': 'string',
                'OptInStatus': 'string',
                'CurrentApplyDate': 'string',
                'Description': 'string'
            },
        ]
    },
],
'NextToken': 'string'
}
```

Dapatkan tindakan pemeliharaan yang tertunda (jika ada) pada yang diberikan`resourceArn`:

```
aws docdb-elastic get-pending-maintenance-action --resource-arn string-arn
```

Keluaran dari operasi ini terlihat seperti berikut ini (format JSON).

```
{
    'ResourcePendingMaintenanceAction': {
        'ResourceArn': 'string-arn',
        'PendingMaintenanceActionDetails': [
            {
                'Action': 'ENGINE_UPDATE',
                'AutoAppliedAfterDate': 'string',
                'ForcedApplyDate': 'string',
                'OptInStatus': 'string',
                'CurrentApplyDate': 'string',
                'Description': 'string'
            }
        ]
    }
}
```

Parameter:

- `ResourceArn`—Nama Sumber Daya Amazon DocumentDB Amazon (ARN) dari sumber daya tempat tindakan pemeliharaan tertunda berlaku.
- `Action`—Tindakan pemeliharaan yang tertunda diterapkan ke sumber daya.

Nilai valid:

- `ENGINE_UPDATE`
- `ENGINE_UPGRADE`
- `SECURITY_UPDATE`
- `OS_UPDATE`
- `MASTER_USER_PASSWORD_UPDATE`

- `AutoAppliedAfterDate`—Jendela pemeliharaan pertama setelah tanggal ini. `NEXT_MAINTENANCE OPT_IN`diabaikan dalam kasus ini.
- `ForcedApplyDate`—Diterapkan terlepas dari jendela pemeliharaan. `IMMEDIATE OPT_IN`diabaikan dalam kasus ini.
- `OptInStatus`—Nilai yang menentukan jenis permintaan keikutsertaan, atau membatalkan permintaan keikutsertaan. Permintaan keikutsertaan tipe `IMMEDIATE` tidak dapat diurungkan.

  Nilai valid:

  - `IMMEDIATE`—Terapkan tindakan pemeliharaan segera.
  - `NEXT_MAINTENANCE`—Terapkan tindakan pemeliharaan selama jendela pemeliharaan berikutnya untuk sumber daya.
  - `APPLY_ON`—Terapkan tindakan pemeliharaan pada tanggal penerapan yang ditentukan terlepas dari jendela pemeliharaan berikutnya untuk sumber daya.
  - `UNDO_OPT_IN`—Batalkan permintaan yang ada `NEXT_MAINTENANCE` atau `APPLY_ON` ikut serta.
- `CurrentApplyDate`—Ditampilkan jika opt-in-type adalah. `APPLY_ON`
- `Description`—Deskripsi opsi untuk tindakan pemeliharaan.

Pembaruan mesin cluster elastis

Dengan Amazon DocumentDB, Anda dapat memilih kapan menerapkan operasi pemeliharaan. Anda dapat memutuskan kapan Amazon DocumentDB menerapkan pembaruan menggunakan. AWS CLI

Terapkan tindakan pemeliharaan yang tertunda:

```
aws docdb-elastic apply-pending-maintenance-action
--resource-arn string-arn
--apply-action string-enum
--opt-in-type string-enum
```

```
[--apply-on string-date-range]
```

Parameter:

- **`--resource-arn`**—Nama Sumber Daya Amazon DocumentDB Amazon (ARN) dari sumber daya tempat tindakan pemeliharaan tertunda berlaku.
- **`--apply-action`**—Tindakan pemeliharaan yang tertunda untuk diterapkan ke sumber daya ini.

  Nilai valid:

  - `ENGINE_UPDATE`
  - `ENGINE_UPGRADE`
  - `SECURITY_UPDATE`
  - `OS_UPDATE`
  - `MASTER_USER_PASSWORD_UPDATE`
- **`--opt-in-type`**—Nilai yang menentukan jenis permintaan keikutsertaan, atau membatalkan permintaan keikutsertaan. Permintaan keikutsertaan tipe `IMMEDIATE` tidak dapat diurungkan.

  Nilai valid:

  - `IMMEDIATE`—Terapkan tindakan pemeliharaan segera.
  - `NEXT_MAINTENANCE`—Terapkan tindakan pemeliharaan selama jendela pemeliharaan berikutnya untuk sumber daya.
  - `APPLY_ON`—Terapkan tindakan pemeliharaan pada tanggal penerapan yang ditentukan terlepas dari jendela pemeliharaan berikutnya untuk sumber daya.
  - `UNDO_OPT_IN`—Batalkan permintaan yang ada `NEXT_MAINTENANCE` atau `APPLY_ON` ikut serta.
- **`[--apply-on]`**—Diperlukan jika opt-in-type adalah. `APPLY_ON` Format: `yyyy/MM/dd HH:mm-yyyy/MM/dd HH:mm` (Opsi ini menggunakan waktu UTC. Waktu mulai bisa kapan saja di masa depan dari minimal 30 menit dan maksimal 14 hari, atau force/apply tanggal tindakan tertunda, mana yang lebih awal. Jendela waktu mulai hingga akhir bisa minimal 30 menit hingga maksimal 8 jam.)

Output dari operasi ini terlihat seperti berikut (format JSON):

```
{
 'ResourcePendingMaintenanceAction': {
        'ResourceArn': 'string-arn',
        'PendingMaintenanceActionDetails': [
```

```
            {
                'Action': 'SECURITY_UPDATE',
                'AutoAppliedAfterDate': 'string',
                'ForcedApplyDate': 'string',
                'OptInStatus': 'IMMEDIATE',
                'CurrentApplyDate': 'string',
                'Description': 'string'
            },
        ]
 }
}
```

Parameter:

- `ResourceArn`—Nama Sumber Daya Amazon DocumentDB Amazon (ARN) dari sumber daya tempat tindakan pemeliharaan tertunda berlaku.
- `Action`—Tindakan pemeliharaan yang tertunda diterapkan ke sumber daya.

  Nilai valid:

  - `ENGINE_UPDATE`
  - `ENGINE_UPGRADE`
  - `SECURITY_UPDATE`
  - `OS_UPDATE`
  - `MASTER_USER_PASSWORD_UPDATE`
- `AutoAppliedAfterDate`—Jendela pemeliharaan pertama setelah tanggal ini. `NEXT_MAINTENANCE OPT_IN`diabaikan dalam kasus ini.
- `ForcedApplyDate`—Diterapkan terlepas dari jendela pemeliharaan. `IMMEDIATE OPT_IN`diabaikan dalam kasus ini.
- `OptInStatus`—Nilai yang menentukan jenis permintaan keikutsertaan, atau membatalkan permintaan keikutsertaan. Permintaan keikutsertaan tipe `IMMEDIATE` tidak dapat diurungkan.

  Nilai valid:

  - `IMMEDIATE`—Terapkan tindakan pemeliharaan segera.
  - `NEXT_MAINTENANCE`—Terapkan tindakan pemeliharaan selama jendela pemeliharaan berikutnya untuk sumber daya.
  - `APPLY_ON`—Terapkan tindakan pemeliharaan pada tanggal penerapan yang ditentukan terlepas dari jendela pemeliharaan berikutnya untuk sumber daya.

  - UNDO_OPT_IN—Batalkan permintaan yang ada NEXT_MAINTENANCE atau APPLY_ON ikut serta.
- CurrentApplyDate—Ditampilkan jika opt-in-type adalah. APPLY_ON
- Description—Deskripsi opsi untuk tindakan pemeliharaan.

Terapkan tanggal

Setiap tindakan pemeliharaan memiliki tanggal penerapan masing-masing yang dapat Anda temukan saat menjelaskan tindakan pemeliharaan yang tertunda. Saat Anda membaca output tindakan pemeliharaan yang tertunda dari AWS CLI, tiga tanggal dicantumkan:

- CurrentApplyDate—Tanggal tindakan pemeliharaan akan diterapkan baik segera atau selama jendela pemeliharaan berikutnya. Jika pemeliharaan opsional, nilai ini bisa nol.
- ForcedApplyDate—Tanggal saat pemeliharaan akan diterapkan secara otomatis, terlepas dari jangka waktu pemeliharaan Anda.
- AutoAppliedAfterDate—Tanggal setelah pemeliharaan akan diterapkan selama jendela pemeliharaan klaster.

User-created tindakan pemeliharaan

Sebagai pengguna cluster Amazon DocumentdBelastic, Anda dapat memulai pembaruan ke konfigurasi cluster Anda.

Memperbarui kata sandi utama klaster

```
aws docdb-elastic update-cluster
--cluster-arn string-arn
[--admin-user-password string]
[--auth-type string-enum]
[--apply-method string-enum]
[--apply-on string-date-range]
#... other parameters of the API that follow here are not relevant for this
 configuration
```

Parameter:

- **--cluster-arn**—Nama Sumber Daya Amazon DocumentDB Amazon (ARN) dari sumber daya tempat tindakan pemeliharaan akan diterapkan.

- **[--admin-user-password]**—Kata sandi yang terkait dengan pengguna admin.
- **[--auth-type]**—Jenis otentikasi yang digunakan untuk menentukan di mana mengambil kata sandi yang digunakan untuk mengakses cluster elastis. Jenis yang valid adalah `PLAIN_TEXT` atau`SECRET_ARN`.
- **[--apply-method]**—Nilai yang menentukan jenis metode yang diterapkan. Nilai yang diizinkan adalah `IMMEDIATE` dan `APPLY_ON`. Default-nya adalah `IMMEDIATE`.
- **[--apply-on]**—Diperlukan jika `apply-method` ada`APPLY_ON`. Format: `yyyy/MM/dd HH:mm-yyyy/MM/dd HH:mm` (Opsi ini menggunakan waktu UTC. Waktu mulai bisa kapan saja di masa depan dari minimal 30 menit dan maksimal 14 hari. Jendela waktu mulai hingga akhir bisa minimal 30 menit hingga maksimal 8 jam.)

Output dari operasi ini terlihat seperti berikut (format JSON):

```
{
 'ResourcePendingMaintenanceAction': {
        'ResourceArn': 'string-arn',
        'PendingMaintenanceActionDetails': [
            {
                'Action': 'MASTER_USER_PASSWORD_UPDATE',
                'OptInStatus': 'APPLY_ON',
                'CurrentApplyDate': 'string',
                'Description': 'string'
            },
        ]
 }
}
```

Mengubah jendela pemeliharaan Amazon DocumentDB Anda

Waktu pemeliharaan harus jatuh pada saat penggunaan terendah dan dengan demikian mungkin perlu diubah dari waktu ke waktu. Cluster elastis Anda tidak tersedia selama waktu ini hanya jika perubahan sistem (seperti perubahan operasi penyimpanan skala) diterapkan dan memerlukan pemadaman. Ini tidak tersedia hanya untuk jumlah waktu minimum yang diperlukan untuk membuat perubahan yang diperlukan.

Defaultnya adalah jendela 30 menit yang dipilih secara acak dari blok waktu 8 jam untuk setiap Wilayah Amazon Web Services, yang terjadi pada hari acak dalam seminggu.

Untuk mengubah jendela pemeliharaan, lihatMemodifikasi konfigurasi cluster elastis.

Pembaruan sistem operasi cluster elastis

Cluster elastis Amazon DocumentDB terkadang memerlukan pembaruan sistem operasi. Amazon DocumentDB meningkatkan sistem operasi ke versi yang lebih baru untuk meningkatkan kinerja database dan postur keamanan pelanggan secara keseluruhan. Pembaruan sistem operasi tidak mengubah versi mesin cluster dari cluster elastis Amazon DocumentDB.

Sebagian besar pembaruan sistem operasi untuk cluster elastis Amazon DocumentDB bersifat opsional dan tidak memiliki tanggal yang ditetapkan untuk menerapkannya. Namun, jika Anda tidak menerapkan pembaruan ini untuk sementara waktu, mereka mungkin pada akhirnya menjadi diperlukan dan diterapkan secara otomatis selama jendela pemeliharaan cluster Anda. Ini untuk membantu menjaga postur keamanan database Anda. Untuk menghindari waktu henti yang tidak terduga, sebaiknya Anda menerapkan pembaruan sistem operasi ke cluster elastis Amazon DocumentDB Anda segera setelah tersedia dan mengatur jendela pemeliharaan klaster Anda pada saat yang sesuai dengan kebutuhan bisnis Anda.

Enkripsi data saat istirahat untuk cluster elastis Amazon DocumentDB

Topik berikut membantu Anda mempelajari, membuat, dan memantau kunci AWS Key Management Service enkripsi untuk cluster elastis Amazon DocumentDB:

Topik

- Bagaimana cluster elastis Amazon DocumentDB menggunakan hibah di AWS KMS
- Buat kunci yang dikelola pelanggan
- Memantau kunci enkripsi Anda untuk cluster elastis Amazon DocumentDB
- Pelajari selengkapnya

Cluster elastis Amazon DocumentDB secara otomatis terintegrasi AWS Key Management Service dengan AWS KMS() untuk manajemen kunci dan menggunakan metode yang dikenal sebagai enkripsi amplop untuk melindungi data Anda. Untuk informasi selengkapnya tentang enkripsi amplop, lihat Enkripsi amplop di Panduan Developer AWS Key Management Service .

An AWS KMS key adalah representasi logis dari sebuah kunci. Kunci KMS mencakup metadata, seperti ID kunci, tanggal pembuatan, deskripsi, dan status kunci. Kunci KMS juga berisi bahan kunci yang digunakan untuk mengenkripsi dan mendekripsi data. Untuk informasi selengkapnya tentang kunci KMS, lihat AWS KMS keys di Panduan Developer AWS Key Management Service .

Cluster elastis Amazon DocumentDB mendukung enkripsi dengan dua jenis kunci:

- AWS kunci yang dimiliki - Cluster elastis Amazon DocumentDB menggunakan kunci ini secara default untuk secara otomatis mengenkripsi data yang dapat diidentifikasi secara pribadi. Anda tidak dapat melihat, mengelola, atau menggunakan kunci AWS milik, atau mengaudit penggunaannya. Namun, Anda tidak perlu mengambil tindakan apa pun atau mengubah program apa pun untuk melindungi kunci yang mengenkripsi data Anda. Untuk informasi selengkapnya, lihat kunci yang AWS dimiliki di Panduan AWS Key Management Service Pengembang.
- Customer-managed keys — Simetris AWS KMS keys yang Anda buat, miliki, dan kelola. Karena Anda memiliki kontrol penuh atas lapisan enkripsi ini, Anda dapat melakukan tugas-tugas seperti:
  - Menetapkan dan memelihara kebijakan utama
  - Menetapkan dan memelihara kebijakan dan hibah IAM
  - Mengaktifkan dan menonaktifkan kebijakan utama
  - Memutar bahan kriptografi kunci
  - Menambahkan tanda
  - Membuat alias kunci
  - Kunci penjadwalan untuk penghapusan

  Untuk informasi selengkapnya, lihat Kunci terkelola pelanggan di Panduan AWS Key Management Service Pengembang.

> **Important**
>
> Anda harus menggunakan kunci KMS enkripsi simetris untuk mengenkripsi klaster Anda karena Amazon DocumentDB hanya mendukung kunci KMS enkripsi simetris. Jangan gunakan kunci KMS asimetris untuk mencoba mengenkripsi data di cluster elastis Amazon DocumentDB Anda. Untuk informasi selengkapnya, lihat Kunci asimetris AWS KMS di Panduan AWS Key Management Service Pengembang.
> Jika Amazon DocumentDB tidak bisa lagi mendapatkan akses ke kunci enkripsi untuk sebuah klaster — misalnya, saat akses ke kunci dicabut — klaster terenkripsi masuk ke status terminal. Dalam hal ini, Anda hanya dapat memulihkan klaster dari backup. Untuk Amazon DocumentDB, backup selalu diaktifkan selama 1 hari. Selain itu, jika Anda menonaktifkan kunci untuk cluster Amazon DocumentDB terenkripsi, Anda akhirnya akan kehilangan akses baca dan tulis ke cluster itu. Ketika Amazon DocumentDB menemukan klaster yang dienkripsi dengan kunci yang tidak dapat diaksesnya, klaster akan dimasukkan ke status terminal.

Dalam kondisi ini, klaster DB tidak lagi tersedia, dan status basis data saat ini tidak dapat dipulihkan. Untuk memulihkan klaster, Anda harus mengaktifkan kembali akses ke kunci enkripsi untuk Amazon DocumentDB, lalu memulihkan klaster dari backup.

Important

Anda tidak dapat mengubah kunci KMS untuk cluster terenkripsi setelah Anda membuatnya. Pastikan untuk menentukan persyaratan kunci enkripsi Anda sebelum Anda membuat cluster elastis terenkripsi Anda.

Bagaimana cluster elastis Amazon DocumentDB menggunakan hibah di AWS KMS

Cluster elastis Amazon DocumentDB memerlukan hibah untuk menggunakan kunci yang dikelola pelanggan Anda.

Saat Anda membuat klaster yang dienkripsi dengan kunci yang dikelola pelanggan, klaster elastis Amazon DocumentDB membuat hibah atas nama Anda dengan mengirimkan permintaan ke. `CreateGrant` AWS KMS Hibah AWS KMS digunakan untuk memberi Amazon DocumentDB cluster elastis akses ke kunci KMS di akun pelanggan.

Cluster elastis Amazon DocumentDB memerlukan hibah untuk menggunakan kunci terkelola pelanggan Anda untuk operasi internal berikut:

- Kirim `DescribeKey` permintaan AWS KMS untuk memverifikasi bahwa ID kunci KMS yang dikelola pelanggan simetris, yang dimasukkan saat membuat pelacak atau koleksi geofence, valid.
- Kirim `GenerateDataKey` permintaan AWS KMS untuk menghasilkan kunci data yang dienkripsi oleh kunci terkelola pelanggan Anda.
- Kirim `Decrypt` permintaan AWS KMS ke untuk mendekripsi kunci data terenkripsi sehingga mereka dapat digunakan untuk mengenkripsi data Anda.
- Anda dapat mencabut akses ke hibah, atau menghapus akses layanan ke kunci yang dikelola pelanggan kapan saja. Jika Anda melakukannya, cluster elastis Amazon DocumentDB tidak akan dapat mengakses data apa pun yang dienkripsi oleh kunci yang dikelola pelanggan, yang memengaruhi operasi yang bergantung pada data tersebut.

Buat kunci yang dikelola pelanggan

Anda dapat membuat kunci terkelola pelanggan simetris dengan menggunakan Konsol Manajemen AWS atau AWS KMS API.

Pembuatan kunci yang dikelola pelanggan simetris

Ikuti langkah-langkah untuk Membuat kunci terkelola pelanggan simetris di Panduan AWS Key Management Service Pengembang.

Kebijakan utama

Kebijakan utama mengontrol akses ke kunci yang dikelola pelanggan Anda. Setiap kunci yang dikelola pelanggan harus memiliki persis satu kebijakan utama, yang berisi pernyataan yang menentukan siapa yang dapat menggunakan kunci dan bagaimana mereka dapat menggunakannya. Saat membuat kunci terkelola pelanggan, Anda dapat menentukan kebijakan kunci. Untuk informasi selengkapnya, lihat informasi akses kunci KMS yang terdapat di AWS Key Management Service ikhtisar Panduan AWS Key Management Service Pengembang.

Untuk menggunakan kunci terkelola pelanggan Anda dengan resource cluster elastis Amazon DocumentDB, operasi API berikut harus diizinkan dalam kebijakan kunci:

- `kms:CreateGrant`— Menambahkan hibah ke kunci yang dikelola pelanggan. Memberikan akses kontrol ke kunci KMS tertentu, yang memungkinkan akses untuk memberikan operasi yang diperlukan Amazon Location Service. Untuk informasi selengkapnya tentang penggunaan hibah, lihat Hibah AWS KMS di Panduan AWS Key Management Service Pengembang.
- `kms:DescribeKey`— Memberikan detail kunci yang dikelola pelanggan untuk memungkinkan Docdb Elastic memvalidasi kunci.
- `kms:Decrypt`— Memungkinkan Docdb Elastic menggunakan kunci data terenkripsi yang disimpan untuk mengakses data terenkripsi.
- `kms:GenerateDataKey`— Memungkinkan Docdb Elastic untuk menghasilkan kunci data terenkripsi dan menyimpannya karena kunci data tidak segera digunakan untuk mengenkripsi.

Untuk informasi selengkapnya, lihat Izin untuk AWS layanan dalam kebijakan utama dan akses kunci pemecahan masalah di Panduan PengembangAWS Key Management Service .

Membatasi akses kunci yang dikelola pelanggan melalui kebijakan IAM

Selain kebijakan kunci KMS, Anda juga dapat membatasi izin kunci KMS dalam kebijakan IAM.

Anda dapat membuat kebijakan IAM lebih ketat dalam berbagai cara. Misalnya, untuk mengizinkan kunci terkelola pelanggan hanya digunakan untuk permintaan yang berasal dari klaster elastis Amazon DocumentDB, Anda dapat `kms:ViaService`menggunakan kunci kondisi dengan nilainya. `docdb-elastic.<region-name>.amazonaws.com`

Untuk informasi selengkapnya, lihat Mengizinkan pengguna di akun lain untuk menggunakan kunci KMS di Panduan Developer AWS Key Management Service .

Memantau kunci enkripsi Anda untuk cluster elastis Amazon DocumentDB

Saat Anda menggunakan kunci terkelola AWS KMS key pelanggan dengan sumber daya Docdb Elastic, Anda dapat menggunakan AWS CloudTrail atau Amazon CloudWatch Logs untuk melacak permintaan yang dikirimkan oleh Docdb Elastic. AWS KMS

Contoh berikut adalah AWS CloudTrail peristiwa untuk`CreateGrant`,, `GenerateDataKeyWithoutPlainTextDecrypt`, dan `DescribeKey` untuk memantau AWS KMS key operasi yang dipanggil oleh Amazon DocumentDB cluster elastis untuk mengakses data yang dienkripsi oleh kunci terkelola pelanggan Anda:

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"
}
```


Pelajari selengkapnya

Sumber daya berikut memberikan informasi lebih lanjut tentang enkripsi data saat istirahat:

- Untuk informasi selengkapnya tentang AWS KMS konsep, lihat konsep AWS Key Management Service dasar di Panduan AWS Key Management Service Pengembang.
- Untuk informasi selengkapnya tentang AWS KMS keamanan, lihat Praktik terbaik keamanan AWS Key Management Service di Panduan AWS Key Management Service Pengembang.

Service-linked peran dalam kelompok elastis

Cluster elastis Amazon DocumentDB AWS Identity and Access Management menggunakan peran terkait layanan (IAM). Peran terkait layanan adalah jenis peran IAM unik yang ditautkan langsung ke cluster elastis Amazon DocumentDB. Service-linked peran telah ditentukan sebelumnya oleh cluster elastis Amazon DocumentDB dan menyertakan semua izin yang diperlukan layanan untuk AWS memanggil layanan lain atas nama Anda.

Peran terkait layanan membuat penggunaan cluster elastis Amazon DocumentDB lebih mudah karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. Cluster elastis Amazon DocumentDB mendefinisikan izin peran terkait layanannya, dan kecuali ditentukan lain, hanya cluster elastis Amazon DocumentDB yang dapat mengambil perannya. Izin yang ditentukan mencakup kebijakan kepercayaan dan kebijakan izin, serta bahwa kebijakan izin tidak dapat dilampirkan ke entitas IAM lainnya. Anda dapat menghapus peran hanya setelah terlebih dahulu menghapus sumber daya terkaitnya. Ini melindungi sumber daya cluster elastis Amazon DocumentDB karena Anda tidak dapat secara tidak sengaja menghapus izin untuk mengakses sumber daya.

Untuk informasi tentang layanan lain yang mendukung peran terkait layanan, lihat AWS layanan yang bekerja dengan IAM dan cari layanan yang ditandai dengan Ya di kolom Service-Linked Peran. Pilih Ya dengan sebuah tautan untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Service-linked izin peran untuk cluster elastis

Cluster elastis Amazon DocumentDB menggunakan peran terkait layanan yang diberi nama untuk `AWS ServiceRoleForDocDB-Elastic` memungkinkan klaster elastis Amazon DocumentDB memanggil layanan atas nama kluster Anda. AWS

Peran terkait layanan ini memiliki kebijakan izin yang menyertainya bernama `AmazonDocDB-ElasticServiceRolePolicy` yang memberikannya izin untuk beroperasi di akun Anda. Kebijakan izin peran memungkinkan klaster elastis Amazon DocumentDB menyelesaikan tindakan berikut pada sumber daya yang ditentukan:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": [
                        "AWS/DocDB-Elastic"
                    ]
                }
            }
        }
    ]
}
```

> **Note**
>
> Anda harus mengonfigurasi izin agar entitas IAM (seperti pengguna, grup, atau peran) dapat membuat, mengedit, atau menghapus peran terkait layanan. Jika Anda menemukan pesan galat berikut: “Tidak dapat membuat sumber daya. Verifikasi bahwa Anda memiliki izin untuk membuat peran terkait layanan. Kalau tidak, tunggu dan coba lagi nanti.” , pastikan Anda mengaktifkan izin berikut:

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

Untuk informasi selengkapnya, lihat izin Service-linked peran di Panduan Pengguna AWS Identity and Access Management.

Membuat peran terkait layanan untuk cluster elastis Amazon DocumentDB

Anda tidak perlu membuat peran terkait layanan secara manual. Saat Anda membuat instans DB, cluster elastis Amazon DocumentDB membuat peran terkait layanan untuk Anda.

Mengedit peran terkait layanan untuk cluster elastis Amazon DocumentDB

Cluster elastis Amazon DocumentDB tidak memungkinkan Anda mengedit peran terkait layanan. `AWS ServiceRoleForDocDB-Elastic` Setelah membuat peran terkait layanan, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin merujuk peran tersebut. Namun, Anda dapat mengedit penjelasan peran menggunakan IAM. Untuk informasi selengkapnya, lihat Mengedit peran terkait layanan di Panduan Pengguna AWS Identity and Access Management.

Menghapus peran terkait layanan untuk cluster elastis Amazon DocumentDB

Jika Anda tidak perlu lagi menggunakan fitur atau layanan yang memerlukan peran terkait layanan, kami merekomendasikan Anda menghapus peran tersebut. Dengan begitu, Anda tidak memiliki entitas yang tidak digunakan yang tidak dipantau atau dipelihara secara aktif. Namun demikian, Anda harus menghapus semua klaster Anda sebelum Anda dapat menghapus peran tertaut layanan.

Membersihkan peran terkait layanan

Sebelum dapat menggunakan IAM untuk menghapus peran tertaut layanan, Anda harus mengonfirmasi terlebih dahulu bahwa peran tersebut tidak memiliki sesi aktif dan menghapus sumber daya yang digunakan oleh peran tersebut.

Untuk memeriksa apakah peran terkait layanan memiliki sesi aktif di konsol IAM:

1. Masuk ke Konsol Manajemen AWSdan buka konsol IAM.

2. Di panel navigasi konsol IAM, pilih Peran. Lalu pilih nama (bukan kotak centang) dari peran `AWS ServiceRoleForDocDB-Elastic`.
3. Di halaman Ringkasan untuk peran yang dipilih, pilih tab Penasihat Akses.

> **Note**
>
> Jika Anda tidak yakin apakah cluster elastis Amazon DocumentDB `AWS ServiceRoleForDocDB-Elastic` menggunakan peran tersebut, Anda dapat mencoba menghapus peran tersebut. Jika layanan menggunakan peran, maka penghapusan gagal dan Anda dapat melihat Wilayah AWS di mana peran tersebut digunakan. Jika peran tersebut sedang digunakan, Anda harus menunggu hingga sesi ini berakhir sebelum dapat menghapus peran tersebut. Anda tidak dapat mencabut sesi untuk peran terkait layanan. Jika Anda ingin menghapus `AWS ServiceRoleForDocDB-Elastic` peran, Anda harus terlebih dahulu menghapus semua cluster Anda.

Menghapus semua klaster Anda

Untuk menghapus cluster di konsol Amazon DocumentDB:

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon DocumentDB.
2. Di panel navigasi, pilih Klaster.
3. Pilih cluster yang ingin Anda hapus.
4. Untuk Tindakan, pilih Hapus.
5. Jika Anda diminta untuk Membuat Snapshot akhir? , pilih Ya atau Tidak.
6. Jika Anda memilih Ya pada langkah sebelumnya, untuk Nama snapshot akhir masukkan nama snapshot akhir Anda.
7. Pilih Hapus.

> **Note**
>
> Anda dapat menggunakan konsol IAM, IAM CLI, atau IAM API untuk menghapus `AWS ServiceRoleForDocDB-Elastic`-peran terkait layanan. Untuk informasi selengkapnya, lihat Menghapus peran terkait layanan di Panduan Pengguna AWS Identity and Access Management.

Memantau Amazon DocumentDB

Memantau AWS layanan Anda adalah bagian penting untuk menjaga sistem Anda tetap sehat dan berfungsi secara optimal. Sebaiknya kumpulkan data pemantauan dari semua bagian solusi AWS Anda sehingga Anda dapat lebih mudah men-debug dan memperbaiki kegagalan atau degradasi, jika itu terjadi. Sebelum Anda mulai memantau solusi AWS Anda, kami merekomendasikan Anda mempertimbangkan dan merumuskan jawaban untuk pertanyaan berikut:

- Apa saja sasaran pemantauan Anda?
- Sumber daya apa yang akan Anda pantau?
- Seberapa sering Anda akan memantau sumber daya ini?
- Alat pemantauan apa yang akan Anda gunakan?
- Siapa yang bertanggung jawab untuk melakukan pemantauan?
- Siapa yang akan diberitahu dan dengan cara apa jika terjadi kesalahan?

Untuk memahami pola performa Anda saat ini, mengidentifikasi anomali performa, dan merumuskan metode untuk mengatasi masalah, Anda harus menetapkan metrik performa dasar untuk berbagai waktu dan dalam kondisi beban yang berbeda. Saat Anda memantau solusi AWS Anda, kami merekomendasikan Anda menyimpan data pemantauan historis Anda untuk referensi di masa mendatang dan untuk menetapkan garis dasar Anda.

Secara umum, nilai yang dapat diterima untuk metrik performa bergantung pada seperti apa garis dasar Anda dan apa yang dilakukan aplikasi Anda. Selidiki variansi yang konsisten atau sedang tren dari acuan Anda. Berikut ini adalah saran tentang jenis metrik tertentu:

- Penggunaan CPU atau RAM yang tinggi — Nilai tinggi untuk penggunaan CPU atau RAM mungkin tepat, asalkan sesuai dengan tujuan Anda untuk aplikasi Anda (seperti throughput atau konkurensi) dan diharapkan.
- Konsumsi volume penyimpanan — Selidiki konsumsi penyimpanan (`VolumeBytesUsed`) jika ruang yang digunakan secara konsisten pada atau di atas 85 persen dari total ruang volume penyimpanan. Tentukan apakah Anda dapat menghapus data dari volume penyimpanan atau mengarsipkan data ke sistem lain untuk mengosongkan ruang. Untuk informasi selengkapnya, lihat Penyimpanan Amazon DocumentDB dan Kuota dan batas Amazon DocumentDB.

- Lalu lintas jaringan — Untuk lalu lintas jaringan, bicarakan dengan administrator sistem Anda untuk memahami throughput yang diharapkan untuk jaringan domain dan koneksi internet Anda. Selidiki lalu lintas jaringan jika throughput selalu di bawah yang diharapkan.
- Koneksi basis data — Pertimbangkan untuk membatasi koneksi basis data jika Anda melihat jumlah koneksi pengguna yang tinggi bersamaan dengan penurunan performa instans dan waktu respons. Jumlah koneksi pengguna terbaik untuk instans Anda akan bervariasi berdasarkan kelas instans Anda dan kompleksitas operasi yang dilakukan.
- Metrik IOPS—Nilai yang diharapkan untuk metrik IOPS bergantung pada spesifikasi disk dan konfigurasi server, jadi gunakan garis dasar Anda untuk mengetahui apa yang khusus. Selidiki apakah nilainya secara konsisten berbeda dari garis dasar Anda. Untuk performa IOPS terbaik, pastikan bahwa set kerja Anda yang biasa sesuai dengan memori untuk meminimalkan operasi baca dan tulis.

Amazon DocumentDB (dengan kompatibilitas MongoDB) menyediakan berbagai metrik CloudWatch Amazon yang dapat Anda pantau untuk menentukan kesehatan dan kinerja cluster dan instans Amazon DocumentDB Anda. Anda dapat melihat metrik Amazon DocumentDB menggunakan berbagai alat, termasuk konsol Amazon DocumentDB, API, dan Performance Insights. AWS CLI CloudWatch

Topik

- Memantau status klaster Amazon DocumentDB
- Memantau status instans Amazon DocumentDB
- Melihat rekomendasi Amazon DocumentDB
- Menggunakan langganan acara Amazon DocumentDB
- Memantau Amazon DocumentDB dengan CloudWatch
- Mencatat panggilan API Amazon DocumentDB dengan AWS CloudTrail
- Membuat profil operasi Amazon DocumentDB
- Memantau dengan Wawasan Kinerja

Memantau status klaster Amazon DocumentDB

Status klaster menunjukkan kondisi klaster. Anda dapat melihat status cluster dengan menggunakan konsol Amazon DocumentDB atau perintah. AWS CLI `describe-db-clusters`

Topik

- Nilai status cluster
- Memantau status klaster

Nilai status cluster

Tabel berikut mencantumkan nilai yang valid untuk status klaster.

status klaster	Deskripsi
`active`	Cluster ini aktif. Status ini hanya berlaku untuk cluster elastis.
`available`	Klaster sehat dan tersedia. Status ini hanya berlaku untuk cluster berbasis instans.
`backing-up`	Klaster saat ini sedang dicadangkan.
`creating`	Klaster sedang dibuat. Klaster tidak dapat diakses ketika sedang dibuat.
`deleting`	Klaster sedang dihapus. Klaster tidak dapat diakses ketika sedang dihapus.
`failing-over`	Sebuah failover dari instans primer ke replika Amazon DocumentDB sedang dilakukan .
`inaccessible-encryption-credentials`	AWS KMS Kunci yang digunakan untuk mengenkripsi atau mendekripsi cluster tidak dapat diakses.

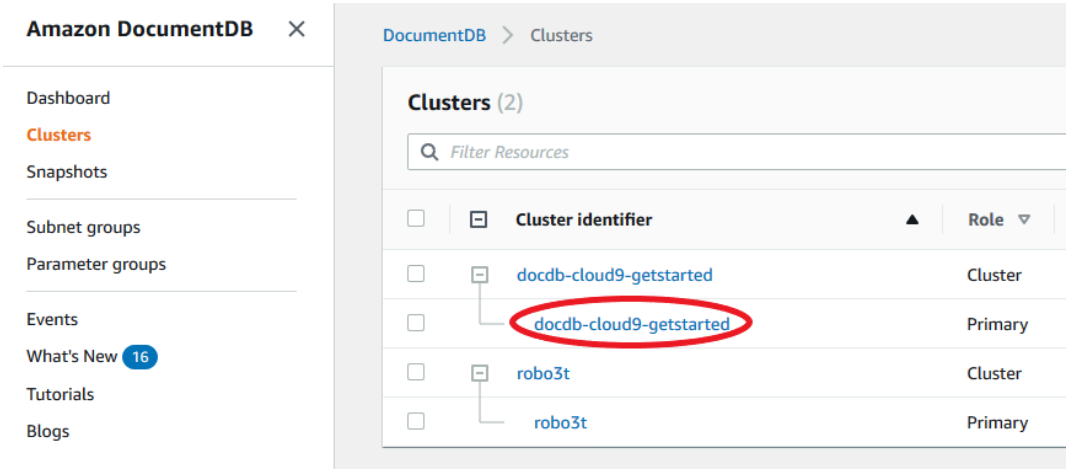
status klaster	Deskripsi
`maintenance`	Pembaruan pemelihar aan sedang diterapkan ke klaster. Status ini digunakan untuk pemeliharaan tingkat klaster yang dijadwalkan jauh sebelumnya oleh Amazon DocumentDB.
`migrating`	Snapshot klaster sedang dipulihkan ke sebuah klaster.
`migration-failed`	Migrasi gagal.
`modifying`	Klaster sedang dimodifikasi karena permintaan pelanggan untuk memodifikasi klaster.
`renaming`	Klaster sedang diganti namanya karena permintaan pelanggan untuk mengganti namanya.
`resetting-master-credentials`	Kredensial utama untuk klaster sedang diatur ulang karena permintaan pelanggan untuk mengatur ulang.
`upgrading`	Versi mesin klaster sedang ditingkatkan.

Memantau status klaster

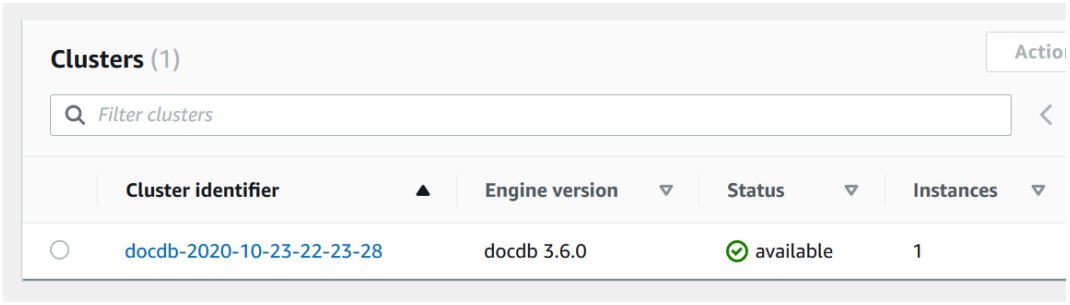
Using the Konsol Manajemen AWS

Saat menggunakan Konsol Manajemen AWS untuk menentukan status cluster, gunakan prosedur berikut.

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Klaster.
3. Di kotak navigasi Klaster, Anda akan melihat kolom Pengidentifikasi klaster. Instance Anda tercantum di bawah cluster, mirip dengan tangkapan layar berikut.



4. Dalam kolom Pengidentifikasi klaster, temukan nama instans yang Anda minati. Kemudian, untuk menemukan status instans, baca baris tersebut ke kolom Status, seperti yang ditunjukkan di bawah ini.



Using the AWS CLI

Saat menggunakan AWS CLI untuk menentukan status cluster, gunakan `describe-db-clusters` operasi. Kode berikut menemukan status klaster `sample-cluster`.

Untuk Linux, macOS, atau Unix:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster  \
    --query 'DBClusters[*].[DBClusterIdentifier,Status]'
```

Untuk Windows:

```
aws docdb describe-db-clusters ^
    --db-cluster-identifier sample-cluster  ^
    --query 'DBClusters[*].[DBClusterIdentifier,Status]'
```

Output dari operasi ini terlihat seperti berikut ini.

```
[
    [
        "sample-cluster",
        "available"
    ]
]
```

Memantau status instans Amazon DocumentDB

Amazon DocumentDB menyediakan informasi tentang kondisi saat ini dari setiap instance yang dikonfigurasi dalam database.

Ada tiga jenis status yang dapat Anda lihat untuk instance Amazon DocumentDB:

- Status instans: Status ini ditampilkan di kolom Status tabel cluster di Konsol Manajemen AWS dan menunjukkan kondisi siklus hidup saat ini dari instance. Nilai yang ditampilkan di kolom Status berasal dari `Status` bidang respons `DescribeDBCluster` API.
- Status kesehatan instans: Status ini ditampilkan di kolom kesehatan Instance dari tabel cluster di Konsol Manajemen AWS dan menunjukkan apakah mesin database, komponen yang bertanggung jawab untuk mengelola dan mengambil data, sedang berjalan. Nilai yang ditampilkan di kolom Instance health didasarkan pada metrik CloudWatch `EngineUptime` sistem Amazon.
- Status pemeliharaan: Status ini ditampilkan di kolom Maintenance dari tabel cluster di Konsol Manajemen AWS dan menunjukkan status dari setiap peristiwa pemeliharaan yang perlu diterapkan ke sebuah instance. Status pemeliharaan independen dari status instance lain' dan berasal dari API. `PendingMaintenanceAction` Untuk informasi selengkapnya tentang status pemeliharaan, lihat Memelihara Amazon DocumentDB.

Topik

- Nilai status instans

- Memantau status instans menggunakan Konsol Manajemen AWS atau AWS CLI
- Nilai status kesehatan contoh
- Memantau status kesehatan contoh menggunakan Konsol Manajemen AWS

Nilai status instans

Tabel berikut mencantumkan kemungkinan nilai status untuk instans dan cara Anda ditagih untuk setiap status. Ini menunjukkan apakah Anda akan ditagih untuk instans dan penyimpanan, hanya penyimpanan, atau tidak ditagih. Untuk semua status instans, Anda selalu ditagih untuk penggunaan pencadangan.

Status instans	Ditagih	Deskripsi
`available`	Ditagih	Instans sehat dan tersedia.
`backing-up`	Ditagih	Instans saat ini sedang dicadangkan.
`configuring-log-exports`	Ditagih	Menerbitkan file log ke Amazon CloudWatch Logs sedang diaktifkan atau dinonaktifkan untuk instance ini.
`creating`	Tidak ditagih	Instans sedang dibuat. Instans tidak dapat diakses saat sedang dibuat.
`deleting`	Tidak ditagih	Instans sedang dihapus.
`failed`	Tidak ditagih	Instans gagal dan Amazon DocumentDB tidak dapat memulihkannya. Untuk memulihkan data, lakukan pemulihan titik waktu ke waktu terbaru yang dapat dipulihkan dari instans.
`inaccessible-encryption-credentials`	Tidak ditagih	AWS KMS Kunci yang digunakan untuk mengenkripsi atau mendekripsi instance tidak dapat diakses.
`incompatible-network`	Tidak ditagih	Amazon DocumentDB mencoba melakukan tindakan pemulihan pada instans tetapi tidak

Status instans	Ditagih	Deskripsi
		dapat melakukannya karena VPC berada dalam status yang mencegah penyelesaian tindakan. Status ini dapat terjadi jika, misalnya, semua alamat IP yang tersedia di subnet sedang digunakan dan Amazon DocumentDB tidak bisa mendapatkan alamat IP untuk instans tersebut.
`maintenance`	Ditagih	Amazon DocumentDB menerapkan pembaruan pemeliharaan ke instans. Status ini digunakan untuk pemeliharaan tingkat instans yang dijadwalkan jauh sebelumnya oleh Amazon DocumentDB. Kami sedang mengevaluasi cara untuk mengekspos tindakan pemeliharaan tambahan kepada pelanggan melalui status ini.
`modifying`	Ditagih	Instans sedang dimodifikasi karena permintaan untuk memodifikasi instans.
`rebooting`	Ditagih	Instans sedang dinyalakan ulang karena permintaan atau proses Amazon DocumentDB yang memerlukan penyalaan ulang instans.
`renaming`	Ditagih	Instans sedang diganti namanya karena ada permintaan untuk mengganti namanya.
`resetting-master-credentials`	Ditagih	Kredensial utama untuk instans sedang diatur ulang karena permintaan untuk mengatur ulang.
`restore-error`	Ditagih	Instans mengalami kesalahan saat mencoba memulihkan ke titik waktu atau dari snapshot.

Status instans	Ditagih	Deskripsi
`starting`	Ditagih untuk penyimpanan	Instans dimulai.
`stopped`	Ditagih untuk penyimpanan	Instans dihentikan.
`stopping`	Ditagih untuk penyimpanan	Instans sedang dihentikan.
`storage-full`	Ditagih	Instans telah mencapai alokasi kapasitas penyimpanannya. Ini adalah status kritis dan harus segera diperbaiki; tingkatkan penyimpanan Anda dengan memodifikasi instans. Atur CloudWatch alarm Amazon untuk memperingatkan Anda saat ruang penyimpanan semakin rendah sehingga Anda tidak mengalami situasi ini.

Memantau status instans menggunakan Konsol Manajemen AWS atau AWS CLI

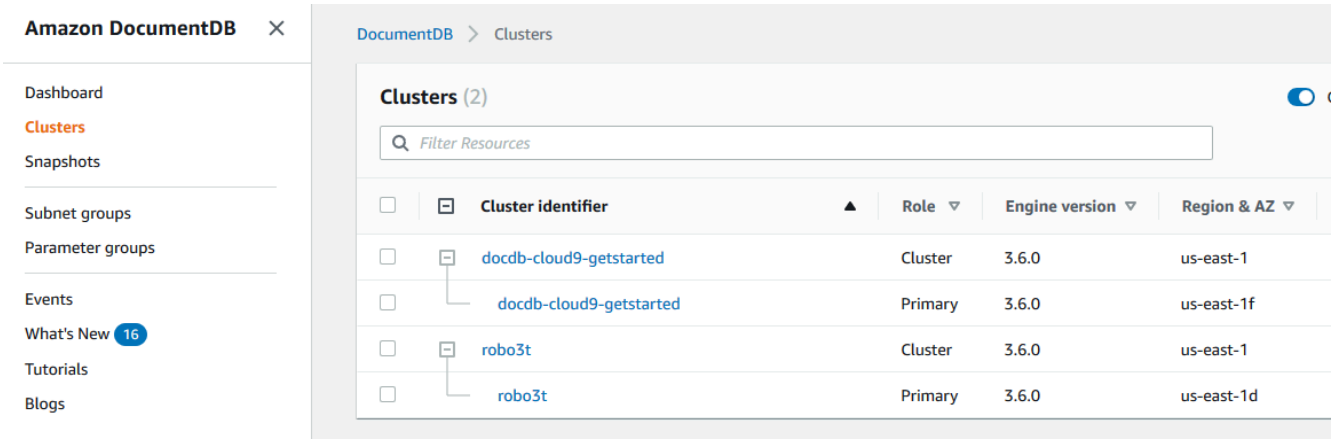
Gunakan Konsol Manajemen AWS atau AWS CLI untuk memantau status instance Anda.

Using the Konsol Manajemen AWS

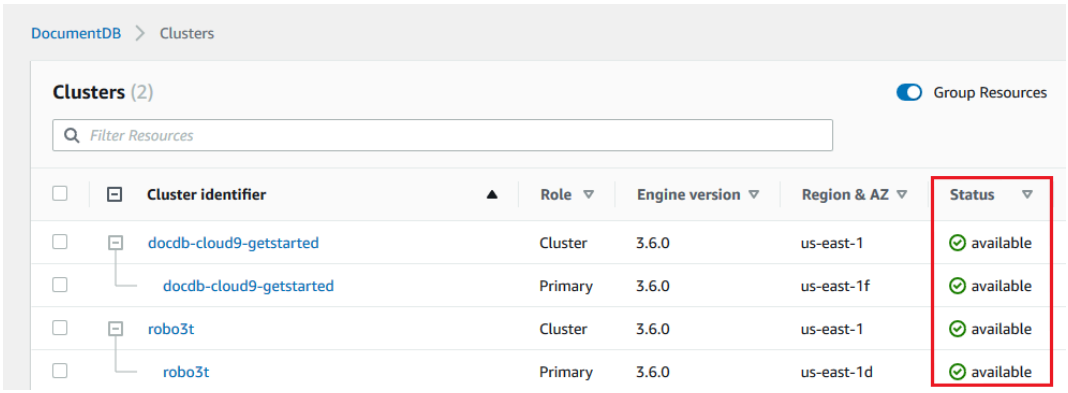
Saat menggunakan Konsol Manajemen AWS untuk menentukan status cluster, gunakan prosedur berikut.

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Pada panel navigasi, silakan pilih Klaster.

> **Note**
>
> Perhatikan bahwa di kotak navigasi Clusters, kolom Cluster identifier menunjukkan cluster dan instance. Contoh tercantum di bawah cluster, mirip dengan gambar berikut.



3. Temukan nama instans yang Anda minati. Kemudian, untuk menemukan status instans, baca sepanjang baris tersebut ke kolom Status, seperti yang ditunjukkan berikut ini.



Using the AWS CLI

Saat menggunakan AWS CLI untuk menentukan status cluster, gunakan `describe-db-instances` operasi. Kode berikut menemukan status instans `sample-cluster-instance-01`.

Untuk Linux, macOS, atau Unix:

```
aws docdb describe-db-instances \
          --db-instance-identifier sample-cluster-instance-01  \
```

```
          --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceStatus]'
```

Untuk Windows:

```
aws docdb describe-db-instances ^
          --db-instance-identifier sample-cluster-instance-01  ^
          --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceStatus]'
```

Output dari operasi ini terlihat seperti berikut.

```
[
          [
              "sample-cluster-instance-01",
              "available"
          ]
      ]
```

Nilai status kesehatan contoh

Tabel berikut mencantumkan nilai status kesehatan yang mungkin untuk instance. Kolom kesehatan Instance, yang terletak di tabel Clusters di Konsol Manajemen AWS, menunjukkan apakah mesin database, komponen yang bertanggung jawab untuk menyimpan, mengelola, dan mengambil data, beroperasi secara normal. Kolom ini juga menunjukkan apakah metrik `EngineUptime` sistem CloudWatch, tersedia di, menunjukkan status kesehatan setiap instance.

Status kesehatan instance	Deskripsi
sehat	Mesin database berjalan di instance Amazon DocumentDB.
tidak sehat	Mesin database tidak berjalan atau telah restart kurang dari satu menit yang lalu.

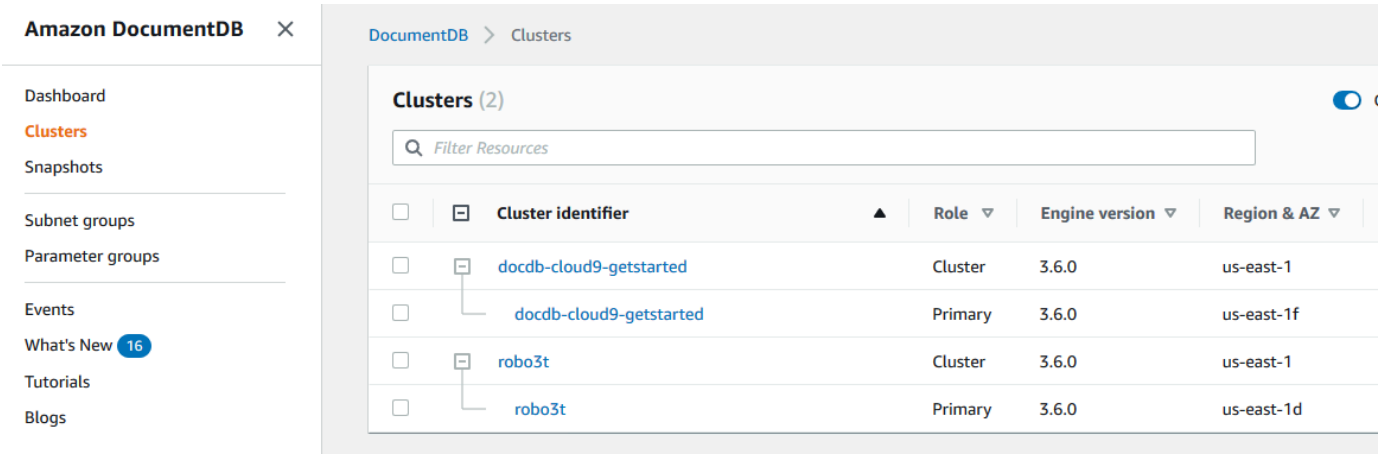
Memantau status kesehatan contoh menggunakan Konsol Manajemen AWS

Gunakan Konsol Manajemen AWS untuk memantau status kesehatan contoh Anda.

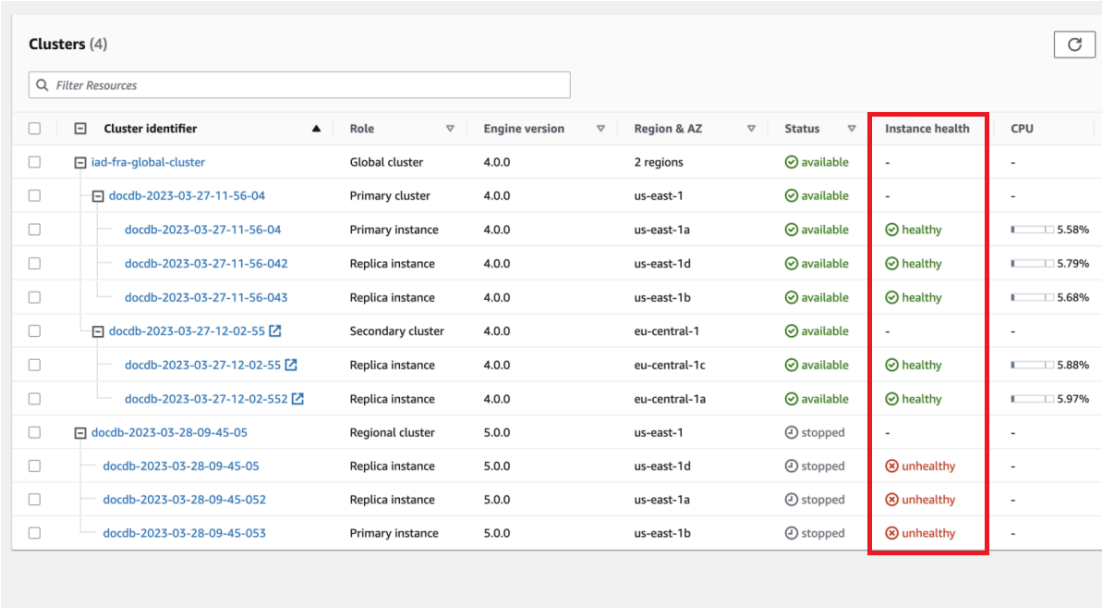
Saat menggunakan Konsol Manajemen AWS, gunakan langkah-langkah berikut untuk memahami status kesehatan instans.

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Pada panel navigasi, silakan pilih Klaster.

> **Note**
>
> Dalam kotak navigasi Clusters, kolom Cluster identifier menunjukkan kedua cluster dan instance. Contoh tercantum di bawah cluster, mirip dengan gambar berikut.



3. Temukan nama instans yang Anda minati. Kemudian, untuk menemukan status instance, baca baris itu ke kolom kesehatan Instance, seperti yang ditunjukkan pada gambar berikut:



> **Note**
>
> Polling status kesehatan contoh terjadi setiap 60 detik dan didasarkan pada metrik CloudWatch `EngineUptime` sistem. Nilai dalam kolom kesehatan Instance diperbarui secara otomatis.

Melihat rekomendasi Amazon DocumentDB

Amazon DocumentDB menyediakan daftar rekomendasi otomatis untuk sumber daya database, seperti instance dan cluster. Rekomendasi ini memberikan panduan praktik terbaik dengan menganalisis konfigurasi cluster dan instans Anda.

Sebagai contoh rekomendasi ini, lihat yang berikut ini:

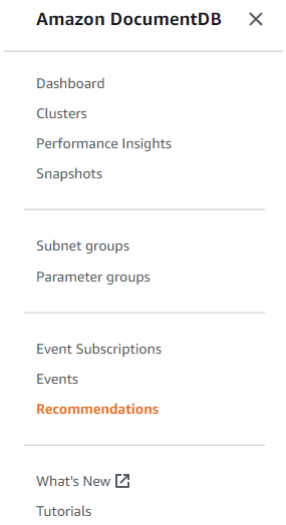
Tipe	Deskripsi	Rekomendasi	Informasi tambahan
Satu instans	Cluster hanya berisi satu contoh	Performa dan ketersediaan: sebaiknya tambahkan instance lain dengan kelas instance yang	Amazon DocumentDB Ketersediaan dan Replikasi Tinggi

Tipe	Deskripsi	Rekomendasi	Informasi tambahan
		sama di Availability Zone yang berbeda.	

Amazon DocumentDB menghasilkan rekomendasi untuk sumber daya saat sumber daya dibuat atau dimodifikasi. Amazon DocumentDB juga memindai sumber daya Anda secara berkala dan menghasilkan rekomendasi.

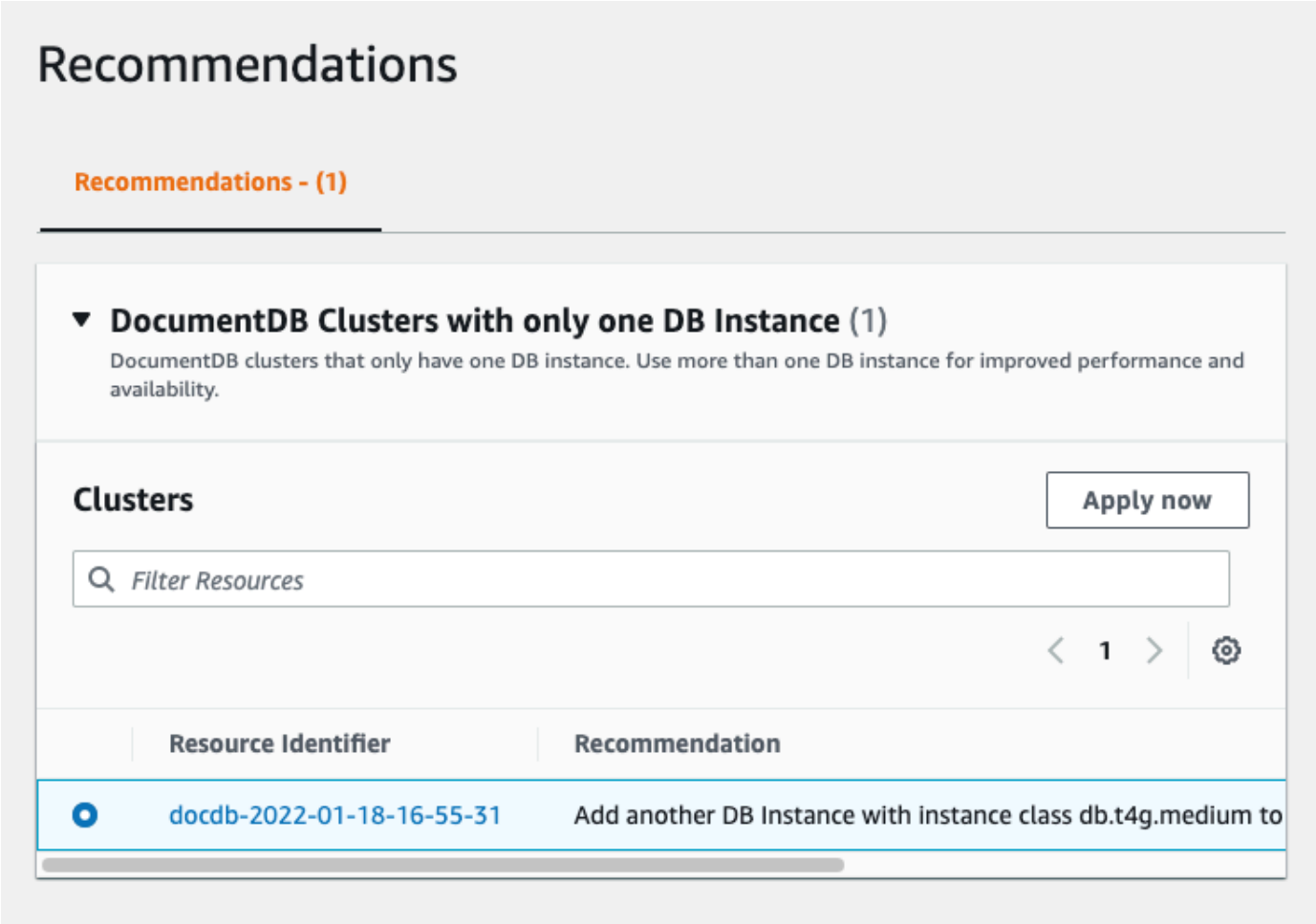
Untuk melihat dan mengambil tindakan pada rekomendasi Amazon DocumentDB

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Rekomendasi:



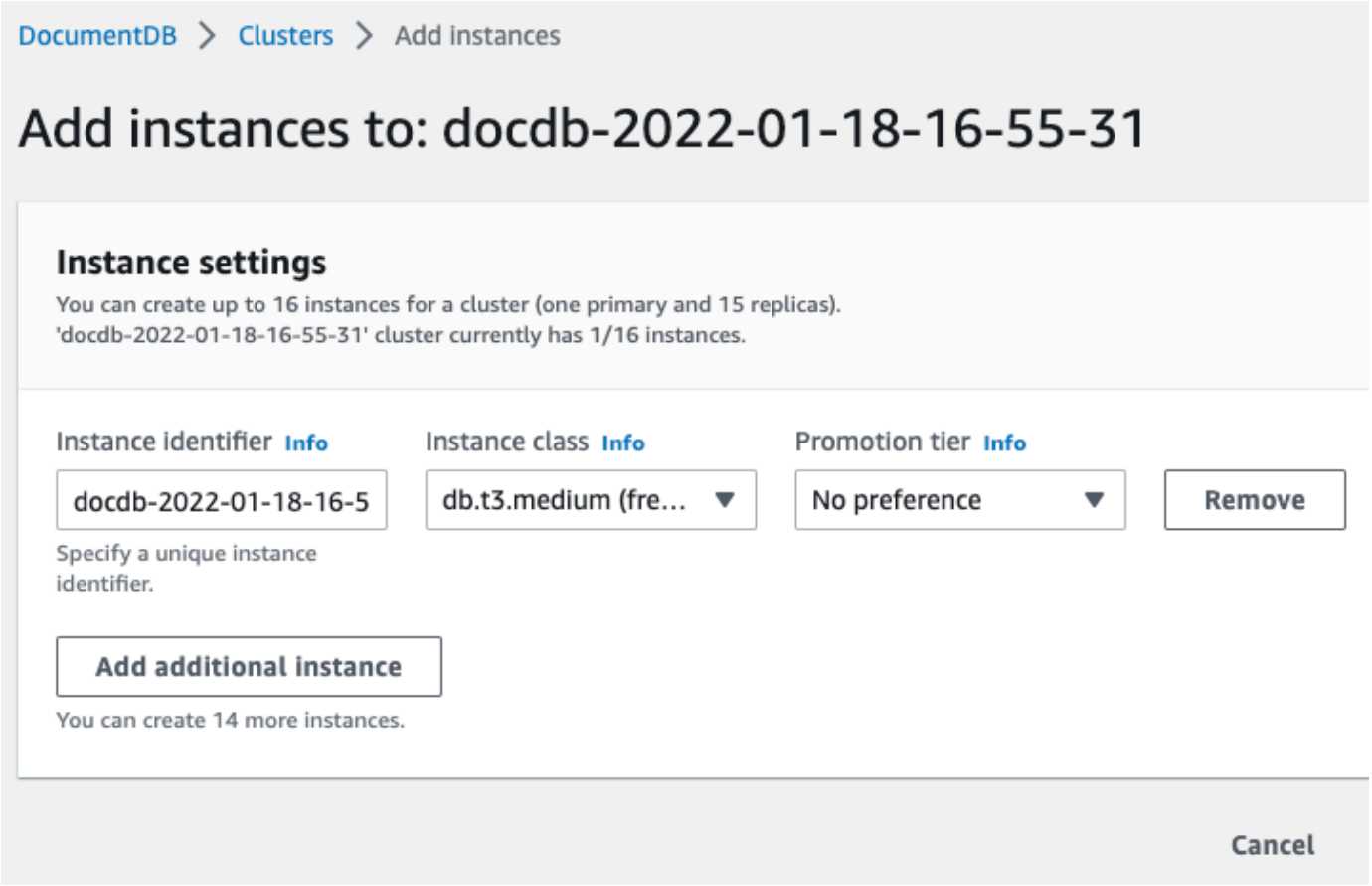
3. Dalam dialog Rekomendasi, perluas bagian yang menarik dan pilih tugas yang disarankan.

   Dalam contoh berikut, tugas yang disarankan berlaku untuk cluster Amazon DocumentDB dengan hanya satu instance. Rekomendasinya adalah menambahkan contoh lain untuk meningkatkan kinerja dan ketersediaan.



4. Pilih Terapkan sekarang.

   Untuk contoh ini, dialog Add instance muncul:



5. Ubah pengaturan instans baru Anda dan pilih Buat.

Menggunakan langganan acara Amazon DocumentDB

Amazon DocumentDB menggunakan Amazon Simple Notification Service (Amazon SNS) untuk memberikan pemberitahuan saat peristiwa Amazon DocumentDB terjadi. Pemberitahuan ini dapat dalam bentuk apa pun yang didukung oleh Amazon SNS untuk Wilayah AWS, seperti email, pesan teks, atau panggilan ke titik akhir HTTP.

Amazon DocumentDB mengelompokkan kejadian ini ke dalam kategori yang dapat Anda langgani sehingga Anda dapat menerima pemberitahuan saat suatu kejadian dalam kategori tersebut terjadi. Anda dapat berlangganan kategori peristiwa untuk instans, klaster, snapshot, snapshot klaster, atau untuk grup parameter. Misalnya, jika Anda berlangganan kategori Backup untuk instans tertentu, Anda akan diberi tahu setiap kali ada kejadian terkait pencadangan yang memengaruhi instans. Anda juga menerima notifikasi saat langganan peristiwa berubah.

Peristiwa terjadi di tingkat klaster dan instans, sehingga Anda dapat menerima peristiwa jika berlangganan ke klaster atau instans.

Langganan peristiwa dikirim ke alamat yang Anda berikan saat membuat langganan. Anda mungkin ingin membuat beberapa langganan berbeda, seperti satu langganan yang menerima semua notifikasi peristiwa dan langganan lain yang hanya mencakup peristiwa penting untuk instans produksi Anda. Anda dapat dengan mudah mematikan notifikasi tanpa menghapus langganan. Untuk melakukannya, atur tombol radio Diaktifkan ke pilihan Tidak di konsol Amazon DocumentDB.

> **Important**
>
> Amazon DocumentDB tidak menjamin urutan acara yang dikirim dalam alur kejadian. Urutan peristiwa dapat berubah sewaktu-waktu.

Amazon DocumentDB menggunakan Amazon Resource Name (ARN) dari topik Amazon SNS untuk mengidentifikasi setiap langganan. Konsol Amazon DocumentDB membuat ARN untuk Anda saat Anda membuat langganan.

Penagihan untuk langganan peristiwa Amazon DocumentDB dilakukan melalui Amazon SNS. Biaya Amazon SNS berlaku saat menggunakan pemberitahuan peristiwa. Untuk informasi selengkapnya, lihat Harga Amazon Simple Notification Service. Selain biaya Amazon SNS, Amazon DocumentDB tidak menagih langganan peristiwa.

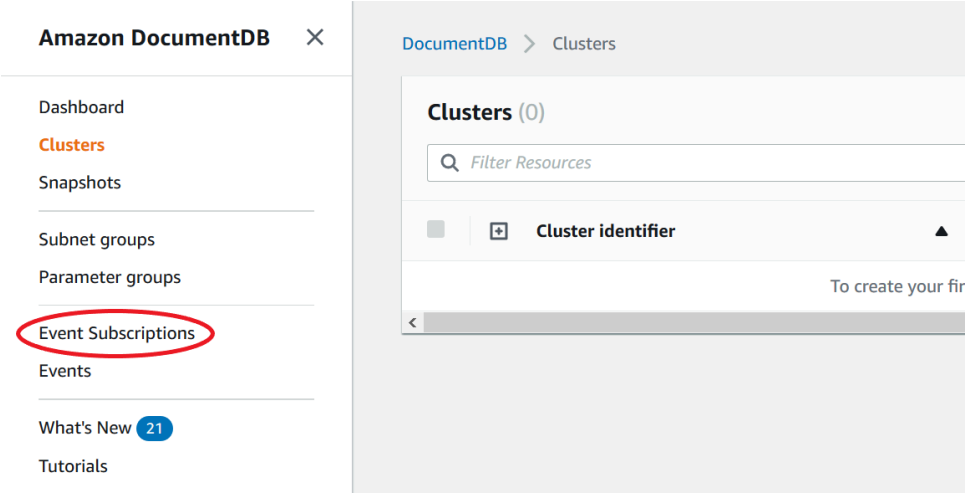
Topik

- Berlangganan acara Amazon DocumentDB
- Mengelola langganan pemberitahuan acara Amazon DocumentDB
- Kategori dan pesan acara Amazon DocumentDB

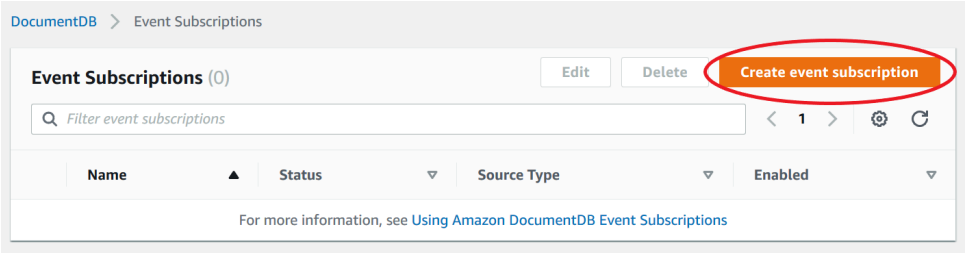
Berlangganan acara Amazon DocumentDB

Anda dapat menggunakan konsol Amazon DocumentDB untuk berlangganan langganan peristiwa, sebagai berikut:

1. Masuk ke Konsol Manajemen AWS athttps://console.aws.amazon.com/docdb.
2. Di panel navigasi, pilih Langganan peristiwa.

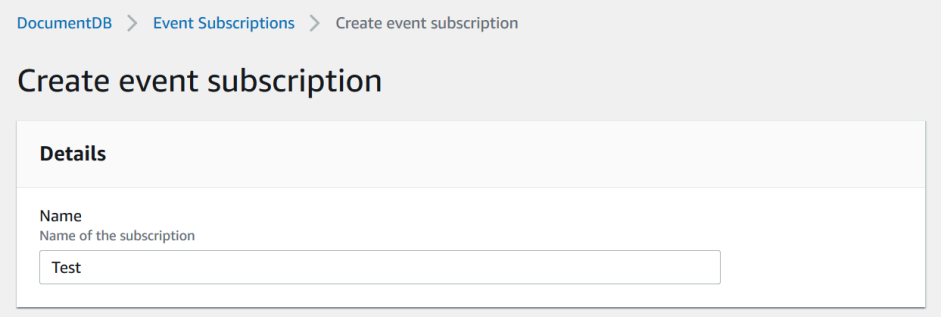


3. Di panel Langganan peristiwa, pilih Buat langganan peristiwa.

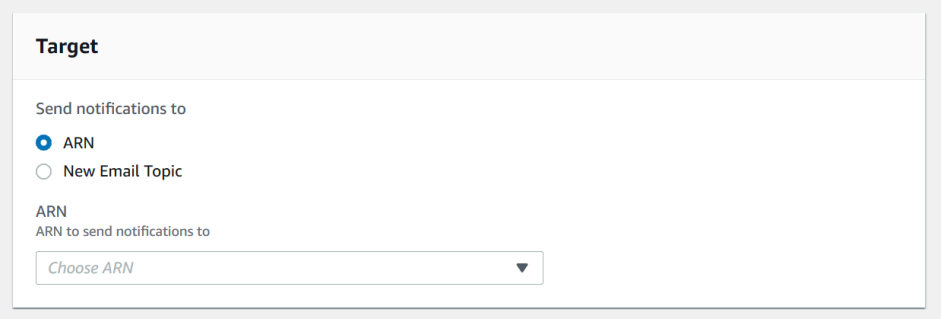


4. Di kotak dialog Buat langganan kejadian, lakukan hal berikut:

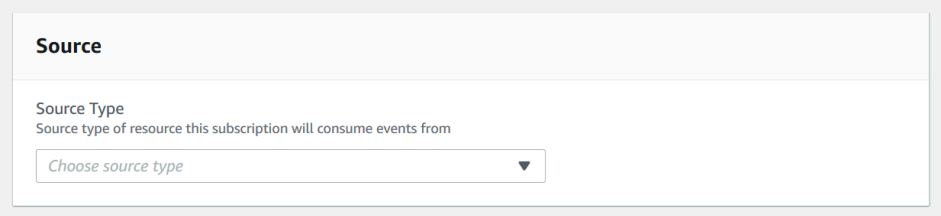
   - Untuk Nama, masukkan nama untuk langganan pemberitahuan kejadian.



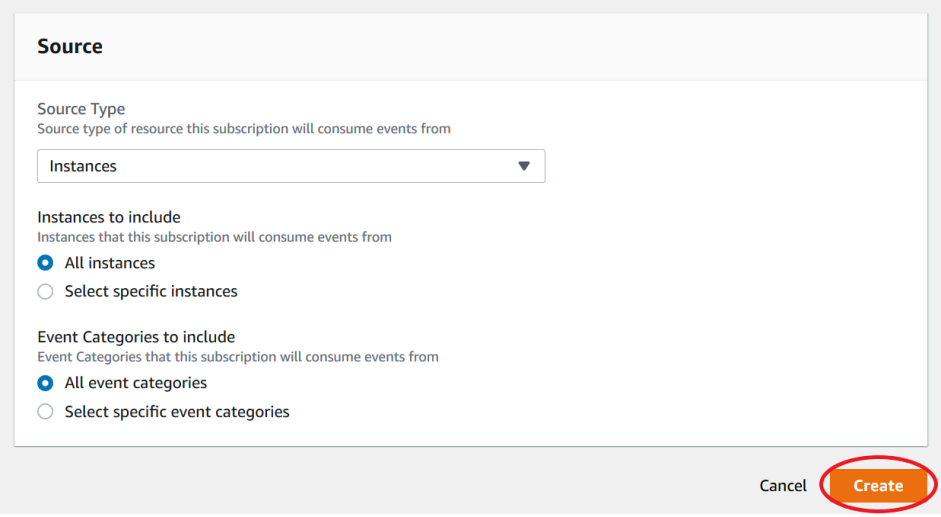
   - Untuk Target, pilih target yang Anda tuju untuk mengirimkan notifikasi. Anda dapat memilih ARN yang ada atau memilih Topik Email Baru untuk memasukkan nama topik dan daftar penerima.



- Untuk Sumber, pilih jenis sumber. Tergantung pada jenis sumber yang Anda pilih, tentukan kategori kejadian dan sumber yang Anda inginkan untuk menjadi pangkal notifikasi kejadian.



- Pilih Buat.



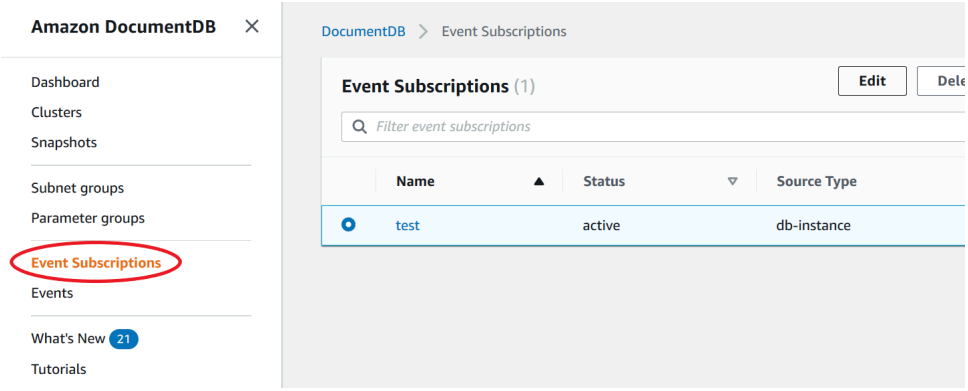
Mengelola langganan pemberitahuan acara Amazon DocumentDB

Jika Anda memilih Berlangganan acara di panel navigasi konsol Amazon DocumentDB, Anda dapat menampilkan kategori berlangganan dan daftar langganan Anda saat ini. Anda juga dapat mengubah atau menghapus langganan tertentu.

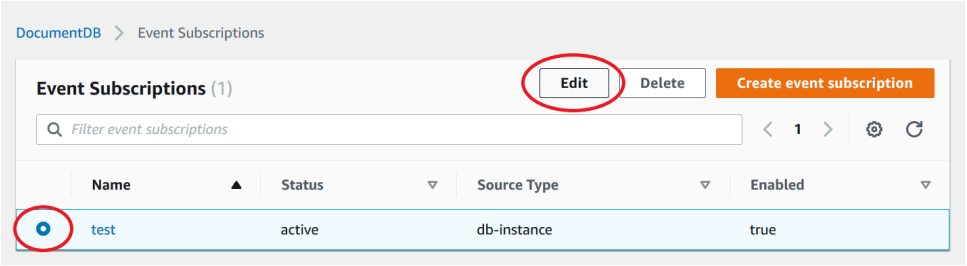
Untuk mengubah langganan notifikasi kejadian Amazon DocumentDB Anda saat ini

1. Masuk ke Konsol Manajemen AWS athttps://console.aws.amazon.com/docdb.

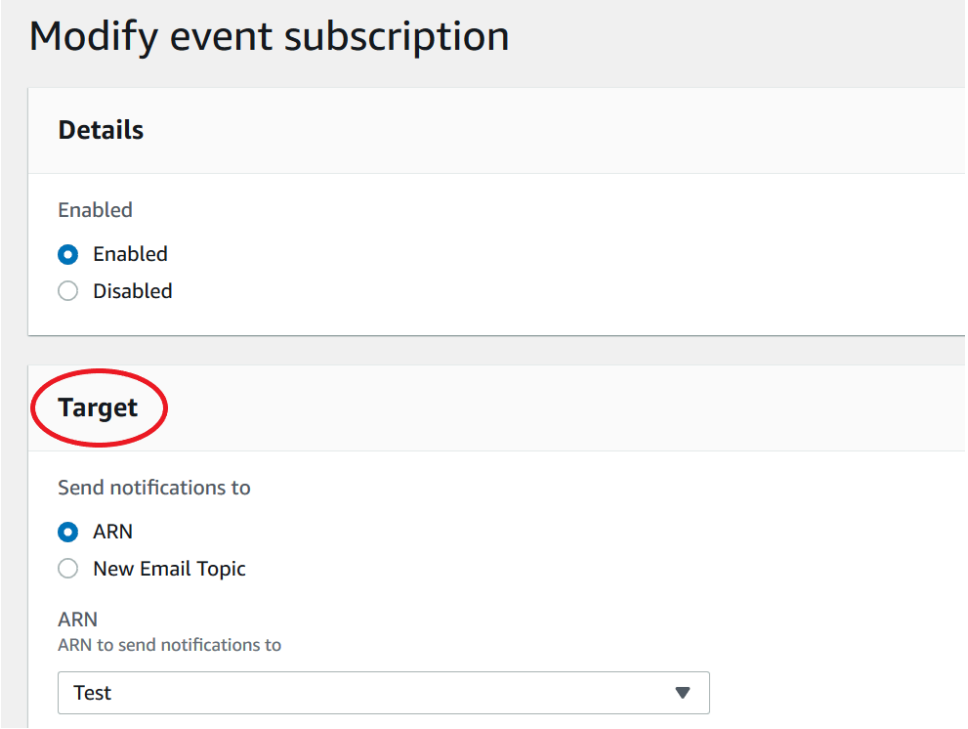
2. Di panel navigasi, pilih Berlangganan peristiwa. Panel Langganan peristiwa menampilkan semua langganan pemberitahuan peristiwa.



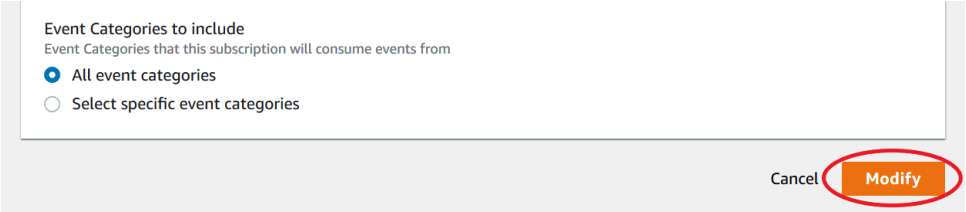
3. Di panel Langganan peristiwa, pilih langganan yang ingin diubah, lalu pilih Edit.



4. Buat perubahan pada langganan di bagian Target atau Sumber. Anda dapat menambahkan atau menghapus pengidentifikasi sumber dengan memilih atau membatalkan pilihan pada bagian Sumber.



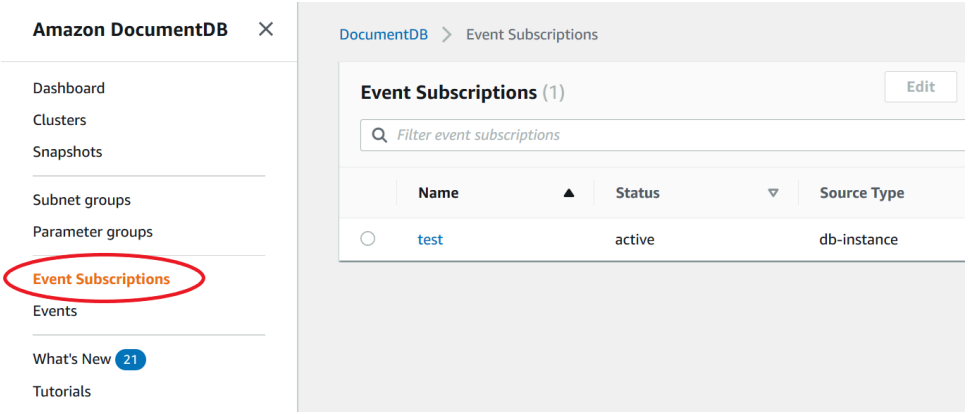
5. Pilih Ubah. Konsol Amazon DocumentDB menunjukkan bahwa langganan sedang dimodifikasi.



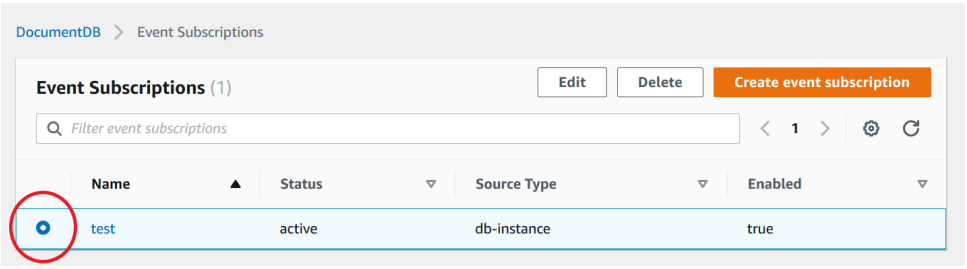
Menghapus langganan notifikasi kejadian Amazon DocumentDB

Anda dapat menghapus langganan jika sudah tidak membutuhkannya lagi. Semua pelanggan topik tidak akan lagi menerima pemberitahuan peristiwa yang ditentukan oleh langganan.

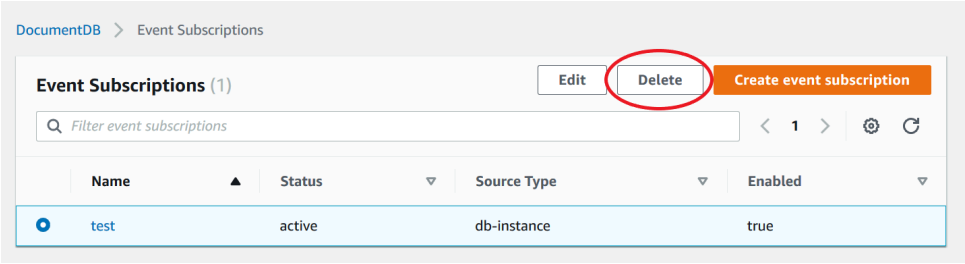
1. Masuk ke Konsol Manajemen AWS athttps://console.aws.amazon.com/docdb.
2. Di panel navigasi, pilih Langganan peristiwa.



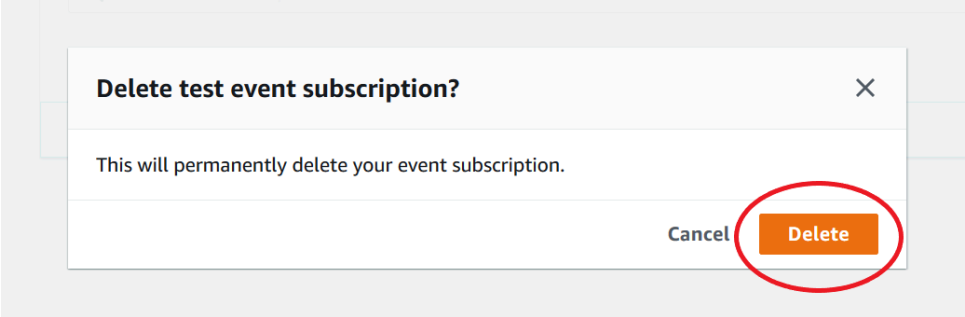
3. Di panel Langganan Kejadian, pilih langganan yang ingin Anda hapus.



4. Pilih Hapus.



5. Jendela pop-up akan muncul menanyakan apakah Anda ingin menghapus notifikasi ini secara permanen. Pilih Hapus.



Kategori dan pesan acara Amazon DocumentDB

Amazon DocumentDB menghasilkan sejumlah besar kejadian dalam kategori yang dapat Anda langgani menggunakan konsol. Setiap kategori berlaku untuk tipe sumber, yang dapat berupa instance, cluster, snapshot, atau grup parameter.

> **Note**
>
> Amazon DocumentDB menggunakan definisi dan ID kejadian Amazon RDS yang ada.

Kejadian Amazon DocumentDB yang berasal dari instans

Kategori	Deskripsi
ketersediaan	Instans dimulai ulang.
ketersediaan	Pematian instans DB.
perubahan konfigurasi	Menerapkan modifikasi pada kelas instans.

Kategori	Deskripsi
perubahan konfigurasi	Selesai menerapkan modifikasi pada kelas instans.
perubahan konfigurasi	Setel ulang kredensial utama.
pembuatan	Instans dibuat.
penghapusan	Instans dihapus
kegagalan	Instans gagal karena konfigurasi yang tidak kompatibel atau masalah penyimpanan yang mendasari. Mulai pemulihan titik-dalam-waktu untuk instans.
pemberitahuan	Instans berhenti.
notifikasi	Instans dimulai.
pemberitahuan	Instans sedang dimulai karena melebihi waktu maksimum penghentian yang diizinkan.
pemulihan	Pemulihan instans telah dimulai. Waktu pemulihan beragam dengan jumlah data yang akan dipulihkan.
pemulihan	Pemulihan instans selesai.
patching keamanan	Pembaruan sistem operasi tersedia untuk instans Anda. Untuk informasi tentang menerapkan pembaruan, lihat Memelihara Amazon DocumentDB.

Kejadian Amazon DocumentDB yang berasal dari klaster

Kategori	Deskripsi
pembuatan	Klaster dibuat
penghapusan	Klaster dihapus.
failover	Mempromosikan ulang primer sebelumnya.
failover	Failover ke instans selesai.
failover	Failover ke instans DB dimulai: %s
failover	Failover AZ yang sama ke instans DB dimulai: %s
failover	Failover AZ melintang ke instans DB dimulai: %s
pemeliharaan	Klaster telah diberi patch.
pemeliharaan	Klaster basis data berada dalam status yang tidak dapat ditingkatkan: %s
pemberitahuan	Klaster berhenti.
pemberitahuan	Klaster dimulai.
pemberitahuan	Perhentian klaster DB gagal.
pemberitahuan	klaster sedang dimulai karena melebihi waktu maksimal yang diperbolehkan untuk dihentikan.
pemberitahuan	Klaster berganti nama dari %s menjadi %s.

Kejadian Amazon DocumentDB yang berasal dari snapshot klaster

Tabel berikut menunjukkan kategori kejadian dan daftar kejadian saat snapshot klaster Amazon DocumentDB merupakan jenis sumber.

Kategori	Deskripsi
pencadangan	Membuat snapshot klaster manual.
pencadangan	Snapshot klaster manual dibuat.
pencadangan	Membuat snapshot klaster otomatis.
pencadangan	Snapshot klaster otomatis dibuat.

Kejadian Amazon DocumentDB yang berasal dari grup parameter

Tabel berikut menunjukkan kategori kejadian dan daftar kejadian saat grup parameter merupakan jenis sumber.

Kategori	Deskripsi
perubahan konfigurasi	Parameter diperbarui %s ke %s dengan metode aplikasi %s

Memantau Amazon DocumentDB dengan CloudWatch

Amazon DocumentDB (dengan kompatibilitas MongoDB) terintegrasi dengan CloudWatch Amazon sehingga Anda dapat mengumpulkan dan menganalisis metrik operasional untuk cluster Anda. Anda dapat memantau metrik ini menggunakan CloudWatch konsol, konsol Amazon DocumentDB, () AWS Command Line Interface ,AWS CLI atau API. CloudWatch

CloudWatch juga memungkinkan Anda mengatur alarm sehingga Anda dapat diberi tahu jika nilai metrik melanggar ambang batas yang Anda tentukan. Anda bahkan dapat mengatur CloudWatch Acara Amazon untuk mengambil tindakan korektif jika terjadi pelanggaran. Untuk informasi selengkapnya tentang penggunaan CloudWatch dan alarm, lihat CloudWatch dokumentasi Amazon.

Topik

- Metrik Amazon DocumentDB
- Melihat CloudWatch data
- Dimensi Amazon DocumentDB

- Memantau metrik Opcounter
- Memantau koneksi database

Metrik Amazon DocumentDB

Untuk memantau kesehatan dan kinerja klaster dan instans Amazon DocumentDB Anda, Anda dapat melihat metrik berikut di konsol Amazon DocumentDB.

> **Note**
>
> Metrik dalam tabel berikut berlaku untuk cluster berbasis instance dan elastis.

Topik

- Metrik pemanfaatan sumber daya
- Metrik latensi
- NVMe-backed metrik contoh
- Metrik operasi
- Metrik throughput
- Metrik sistem
- Metrik contoh T3

Metrik pemanfaatan sumber daya

Metrik	Deskripsi	
`BackupRetentionPeriodStorageUsed`	Jumlah total penyimpanan cadangan dalam byte yang digunakan untuk mendukung fitur pemulihan point-in-time dalam jendela retensi Amazon DocumentDB. Termasuk dalam total yang dilaporkan oleh metrik `TotalBackupStorageBilled` .	

Metrik	Deskripsi	
	Dihitung secara terpisah untuk setiap klaster Amazon DocumentDB.	
`ChangeStreamLogSize`	Jumlah penyimpanan yang digunakan oleh klaster Anda untuk menyimpan log aliran perubahan dalam megabyte. Nilai ini adalah subset dari total penyimpanan untuk klaster (`VolumeBytesUsed`) dan memengaruhi biaya klaster. Untuk informasi harga penyimpanan, lihat Halaman produk Amazon DocumentDB. Ukuran log aliran perubahan adalah fungsi dari seberapa banyak perubahan yang terjadi di klaster Anda dan durasi retensi aliran perubahan yang lama. Untuk informasi lebih lanjut tentang aliran perubahan, lihat Menggunakan aliran perubahan dengan Amazon DocumentDB.	
`CPUUtilization`	Persentase CPU yang digunakan oleh instans.	
`DatabaseConnections`	Jumlah koneksi (aktif dan idle) terbuka pada instance yang diambil pada frekuensi 1 menit.	

Metrik	Deskripsi
DatabaseConnectionsMax	Jumlah maksimum koneksi database terbuka (aktif dan idle) pada instance dalam periode 1 menit.
DatabaseConnectionsLimit	Jumlah maksimum koneksi database bersamaan (aktif dan idle) diizinkan pada instance pada waktu tertentu.
DatabaseCursors	Jumlah kursor terbuka pada instance yang diambil pada frekuensi 1 menit.
DatabaseCursorsMax	Jumlah maksimum kursor terbuka pada instance dalam periode 1 menit.
DatabaseCursorsLimit	Jumlah maksimum kursor yang diizinkan pada suatu instance pada waktu tertentu.
DatabaseCursorsTimedOut	Jumlah kursor yang habis waktunya dalam periode 1 menit.
FreeableMemory	Jumlah memori akses acak yang tersedia, dalam byte.

Metrik	Deskripsi
`FreeLocalStorage`	Metrik ini melaporkan jumlah penyimpanan yang tersedia untuk setiap instans untuk tabel dan log sementara . Nilai ini tergantung pada kelas instans. Anda dapat meningkatkan jumlah ruang penyimpanan gratis untuk sebuah instans dengan memilih kelas instans yang lebih besar untuk instans Anda. (Ini tidak berlaku untuk DocumentDB Tanpa Server.)
`LowMemThrottleQueueDepth`	Kedalaman antrian untuk permintaan yang dibatasi karena memori yang tersedia rendah diambil pada frekuensi 1 menit.
`LowMemThrottleMaxQueueDepth`	Kedalaman antrian maksimum untuk permintaan yang dibatasi karena memori yang tersedia rendah dalam periode 1 menit.
`LowMemNumOperationsThrottled`	Jumlah permintaan yang dibatasi karena memori yang tersedia rendah dalam periode 1 menit.

Metrik	Deskripsi
SnapshotStorageUsed	Jumlah total penyimpanan cadangan dalam byte yang dikonsumsi oleh semua snapshot untuk cluster Amazon DocumentDB tertentu di luar jendela retensi cadangannya. Termasuk dalam total yang dilaporkan oleh metrik TotalBackupStorageBilled . Dihitung secara terpisah untuk setiap klaster Amazon DocumentDB.
SwapUsage	Jumlah ruang tukar yang digunakan pada instans.
TotalBackupStorageBilled	Jumlah total penyimpanan cadangan dalam byte yang Anda ditagih untuk klaster Amazon DocumentDB tertentu. Termasuk penyimpanan backup yang diukur oleh metrik BackupRetentionPeriodStorageUsed dan SnapshotStorageUsed . Dihitung secara terpisah untuk setiap klaster Amazon DocumentDB.
TransactionsOpen	Jumlah transaksi terbuka pada instance yang diambil pada frekuensi 1 menit.

Metrik	Deskripsi
TransactionsOpenMax	Jumlah maksimum transaksi terbuka pada instans dalam periode 1 menit.
TransactionsOpenLimit	Jumlah maksimum transaksi bersamaan yang diizinkan pada suatu instans pada waktu tertentu.
VolumeBytesUsed	Jumlah penyimpanan yang digunakan oleh klaster Anda dalam byte. Nilai ini memengaruhi biaya klaster. Untuk informasi harga, lihat Halaman produk Amazon DocumentDB.

Metrik latensi

Metrik	Deskripsi
DBClusterReplicaLagMaximum	Jumlah maksimum lag, dalam milidetik, antara instans primer dan setiap instans Amazon DocumentDB dalam klaster.
DBClusterReplicaLagMinimum	Jumlah minimum lag, dalam milidetik, antara instans primer dan setiap instans replika dalam klaster.
DBInstanceReplicaLag	Jumlah lag, dalam milidetik, saat mereplikasi pembaruan dari instans primer ke instans replika.

Metrik	Deskripsi
`ReadLatency`	Jumlah rata-rata waktu yang dibutuhkan per I/O operasi disk.
`WriteLatency`	Jumlah rata-rata waktu, dalam milidetik, diambil per I/O operasi disk.

NVMe-backed metrik contoh

Metrik	Deskripsi
`NVMeStorageCacheHitRatio`	Persentase permintaan yang dilayani oleh cache berjenjang.
`FreeNVMeStorage`	Jumlah penyimpanan NVMe Efemeral yang tersedia.
`ReadIOPSNVMeStorage`	Jumlah rata-rata I/O operasi membaca disk ke penyimpanan NVMe Ephemeral.
`ReadLatencyNVMeStorage`	Jumlah rata-rata waktu yang dibutuhkan per I/O operasi pembacaan disk untuk penyimpanan NVMe Ephemeral.
`ReadThroughputNVMeStorage`	Jumlah rata-rata byte yang dibaca dari disk per detik untuk penyimpanan NVMe Efemeral.

Metrik	Deskripsi
`WriteIOPSNVMeStorage`	Jumlah rata-rata I/O operasi penulisan disk ke penyimpanan NVMe Ephemeral.
`WriteLatencyNVMeStorage`	Jumlah rata-rata waktu yang dibutuhkan per I/O operasi penulisan disk untuk penyimpanan NVMe Ephemeral.
`WriteThroughputNVMeStorage`	Jumlah rata-rata byte yang ditulis ke disk per detik untuk penyimpanan NVMe Efemeral.

Metrik operasi

Metrik	Deskripsi
`DocumentsDeleted`	Jumlah dokumen yang dihapus dalam periode 1 menit.
`DocumentsInserted`	Jumlah dokumen yang dimasukkan dalam periode 1 menit.
`DocumentsReturned`	Jumlah dokumen yang dikembalikan dalam periode 1 menit.
`DocumentsUpdated`	Jumlah dokumen yang diperbarui dalam periode 1 menit.

Metrik	Deskripsi
OpcountersCommand	Jumlah perintah yang dikeluarkan dalam periode 1 menit.
OpcountersDelete	Jumlah operasi penghapus an yang dikeluarkan dalam periode 1 menit.
OpcountersGetmore	Jumlah getmores yang dikeluarkan dalam periode 1 menit.
OpcountersInsert	Jumlah operasi insert yang dikeluarkan dalam periode 1 menit.
OpcountersQuery	Jumlah pertanyaan yang dikeluarkan dalam periode 1 menit.
OpcountersUpdate	Jumlah operasi pembaruan yang dikeluarkan dalam periode 1 menit.
TransactionsStarted	Jumlah transaksi dimulai pada sebuah instance dalam periode 1 menit.
TransactionsCommit ted	Jumlah transaksi yang dilakukan pada suatu instans dalam periode 1 menit.
TransactionsAborted	Jumlah transaksi yang dibatalkan pada suatu instans dalam periode 1 menit.

Metrik	Deskripsi	
TTLDeletedDocuments	Jumlah dokumen yang dihapus oleh TTLMonitor dalam periode 1 menit.	

Metrik throughput

Metrik	Deskripsi	
NetworkReceiveThroughput	Jumlah throughput jaringan, dalam byte per detik, yang diterima dari klien oleh setiap instans dalam klaster. Throughput ini tidak mencakup lalu lintas jaringan antar instans dalam klaster dan volume klaster.	
NetworkThroughput	Jumlah throughput jaringan, dalam byte per detik, yang diterima dari dan ditransmisikan ke klien oleh setiap instans dalam klaster Amazon DocumentDB. Throughput ini tidak mencakup lalu lintas jaringan antar instans dalam klaster dan volume klaster.	
NetworkTransmitThroughput	Jumlah throughput jaringan, dalam byte per detik, yang dikirim ke klien oleh setiap instans dalam klaster. Throughput ini tidak mencakup lalu lintas jaringan antar	

Metrik	Deskripsi
	instans dalam klaster dan volume klaster.
`ReadIOPS`	Jumlah rata-rata I/O operasi membaca disk per detik. Amazon DocumentDB melaporkan membaca dan menulis IOPS secara terpisah, dan pada interval 1 menit.
`ReadThroughput`	Jumlah byte rata-rata yang dibaca dari disk per detik.
`StorageNetworkReceiveThroughput`	Jumlah throughput jaringan, dalam byte per detik, yang diterima dari volume penyimpanan cluster Amazon DocumentDB oleh setiap instance di cluster.
`StorageNetworkTransmitThroughput`	Jumlah throughput jaringan, dalam byte per detik, dikirim ke volume penyimpanan cluster Amazon DocumentDB oleh setiap instance di cluster.
`StorageNetworkThroughput`	Jumlah throughput jaringan, dalam byte per detik, diterima dan dikirim ke volume penyimpanan cluster Amazon DocumentDB oleh setiap instance di cluster Amazon DocumentDB.

Metrik	Deskripsi
VolumeReadIOPs	Jumlah rata-rata I/O operasi baca yang ditagih dari volume cluster, dilaporkan pada interval 5 menit. Operasi baca yang ditagih dihitung pada tingkat volume klaster, diagregasikan dari semua instans dalam klaster, dan kemudian dilaporkan pada interval 5 menit. Nilai dihitung dengan mengambil nilai dari metrik operasi baca selama periode 5 menit. Anda dapat menentukan jumlah operasi pembacaan yang ditagih per detik dengan mengambil nilai metrik operasi baca yang ditagih dan membaginya dengan 300 detik. Misalnya, jika VolumeReadIOPs menghasilkan 13.686, kemudian operasi baca yang ditagih per detik adalah 45 (13.686 / 300 = 45,62). Anda mengakumulasi operasi baca tertagih untuk kueri yang meminta halaman basis data yang tidak ada dalam cache buffer dan oleh karena itu harus dimuat dari penyimpanan. Anda mungkin melihat lonjakan dalam operasi baca yang ditagih karena hasil kueri

Metrik	Deskripsi	
	dibaca dari penyimpanan, lalu dimuat ke dalam cache buffer.	

Metrik	Deskripsi
`VolumeWriteIOPs`	Jumlah rata-rata I/O operasi penulisan yang ditagih dari volume cluster, dilaporkan pada interval 5 menit. Operasi tulis yang ditagih dihitung pada tingkat volume klaster, diagregasikan dari semua instans dalam klaster, dan kemudian dilaporkan pada interval 5 menit. Nilai dihitung dengan mengambil nilai dari metrik operasi tulis selama periode 5 menit. Anda dapat menentukan jumlah operasi tulis yang ditagih per detik dengan mengambil nilai metrik operasi tulis yang ditagih dan membaginya dengan 300 detik. Misalnya, jika `VolumeWriteIOPs` menghasilkan 13.686, kemudian operasi tulis yang ditagih per detik adalah 45 (13.686 / 300 = 45,62). Perhatikan bahwa metrik `VolumeReadIOPs` dan `VolumeWriteIOPs` dihitung oleh lapisan penyimpanan DocumentDB dan itu termasuk IO yang dilakukan oleh instans primer dan replika. Data dikumpulkan setiap 20-30 menit dan kemudian dilaporka

Metrik	Deskripsi
	n pada interval 5 menit, sehingga memancarkan titik data yang sama untuk metrik dalam periode waktu. Jika Anda mencari metrik untuk dikorelasikan dengan operasi penyisipan selama interval 1 menit, Anda dapat menggunakan metrik WriteIOPS tingkat instans. Metrik ini tersedia di tab pemantauan instans primer Amazon DocumentDB Anda.
`WriteIOPS`	Jumlah rata-rata I/O operasi penulisan disk per detik. Ketika digunakan pada tingkat cluster, `WriteIOPs` dievaluasi di semua instance di cluster. IOPS baca dan tulis dilaporkan secara terpisah, dengan interval 1 menit.
`WriteThroughput`	Jumlah byte rata-rata yang ditulis dari disk per detik.

Metrik sistem

Metrik	Deskripsi
`AvailableMVCCIds`	Penghitung yang menunjukkan jumlah operasi penulisan yang tersisa yang tersedia sebelum mencapai nol. Saat penghitung ini mencapai nol,

Metrik	Deskripsi
	klaster Anda akan memasuki mode hanya-baca hingga ID direklamasi dan didaur ulang. Penghitung berkurang dengan setiap operasi penulisan dan meningkat saat pengumpul an sampah mendaur ulang ID MVCC lama.
`BufferCacheHitRatio`	Persentase permintaan yang dilayani oleh cache buffer.
`DiskQueueDepth`	Jumlah I/O operasi yang menunggu untuk ditulis atau dibaca dari disk.
`EngineUptime`	Jumlah waktu, dalam detik, instans telah berjalan.
`IndexBufferCacheHi tRatio`	Persentase permintaan indeks yang dilayani oleh cache buffer. Anda mungkin melihat lonjakan lebih dari 100 persen untuk metrik tepat setelah Anda menjatuhkan indeks, koleksi, atau basis data. Ini akan secara otomatis dikoreksi setelah 60 detik. Batasan ini akan diperbaik i dalam pembaruan patch di masa mendatang.

Metrik	Deskripsi
LongestActiveGCRuntime	Durasi dalam hitungan detik dari proses pengumpulan sampah aktif terpanjang. Memperbarui setiap menit dan hanya melacak operasi aktif, tidak termasuk proses yang selesai dalam jendela satu menit.

Metrik contoh T3

Metrik	Deskripsi
CPUCreditUsage	Jumlah kredit CPU yang digunakan selama periode pengukuran.
CPUCreditBalance	Jumlah kredit CPU yang diperoleh instans. Saldo ini habis saat CPU melonjak dan kredit CPU digunakan lebih cepat daripada yang diperoleh .
CPUSurplusCreditBalance	Jumlah kredit CPU surplus yang dihabiskan untuk mempertahankan kinerja CPU ketika CPUCreditBalance nilainya nol.
CPUSurplusCreditsCharged	Jumlah kredit CPU surplus yang melebihi jumlah kredit CPU maksimum yang dapat diperoleh dalam periode 24 jam, dan dengan demikian

Metrik	Deskripsi	
	menarik biaya tambahan. Untuk informasi selengkapnya, lihat Memantau kredit CPU Anda.	

Melihat CloudWatch data

Anda dapat melihat CloudWatch data Amazon menggunakan CloudWatch konsol, konsol Amazon DocumentDB AWS Command Line Interface ,AWS CLI(), atau API. CloudWatch

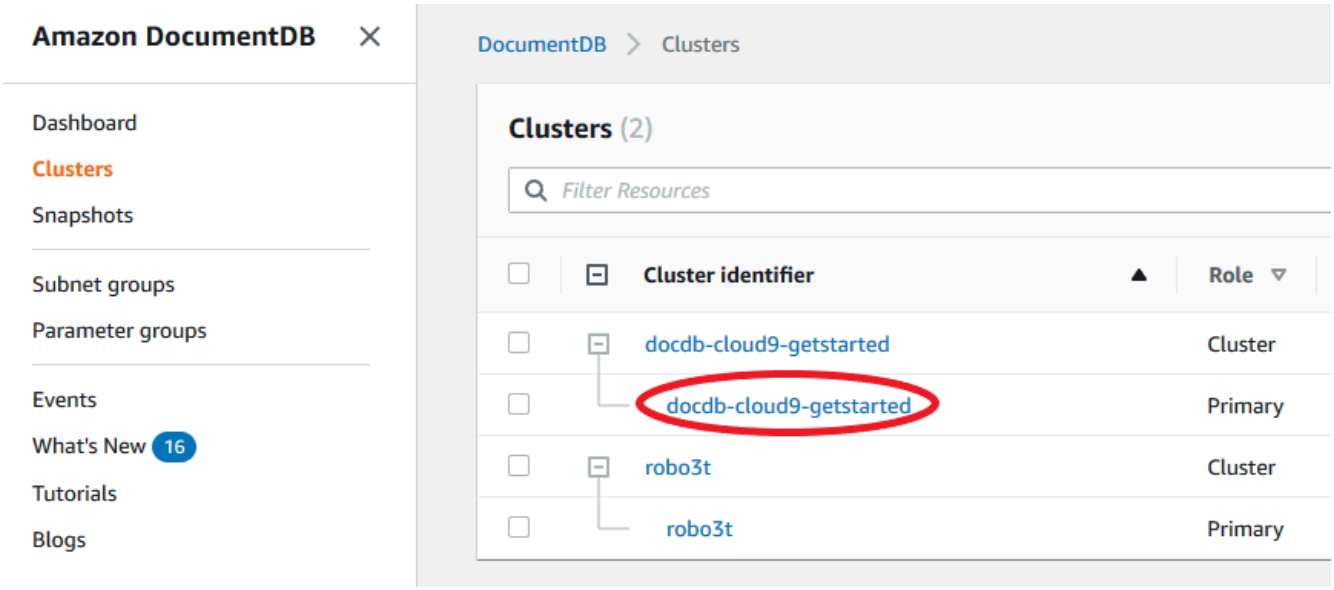
Using the Konsol Manajemen AWS

Untuk melihat CloudWatch metrik menggunakan Amazon DocumentDB Management Console, selesaikan langkah-langkah berikut.

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Klaster.

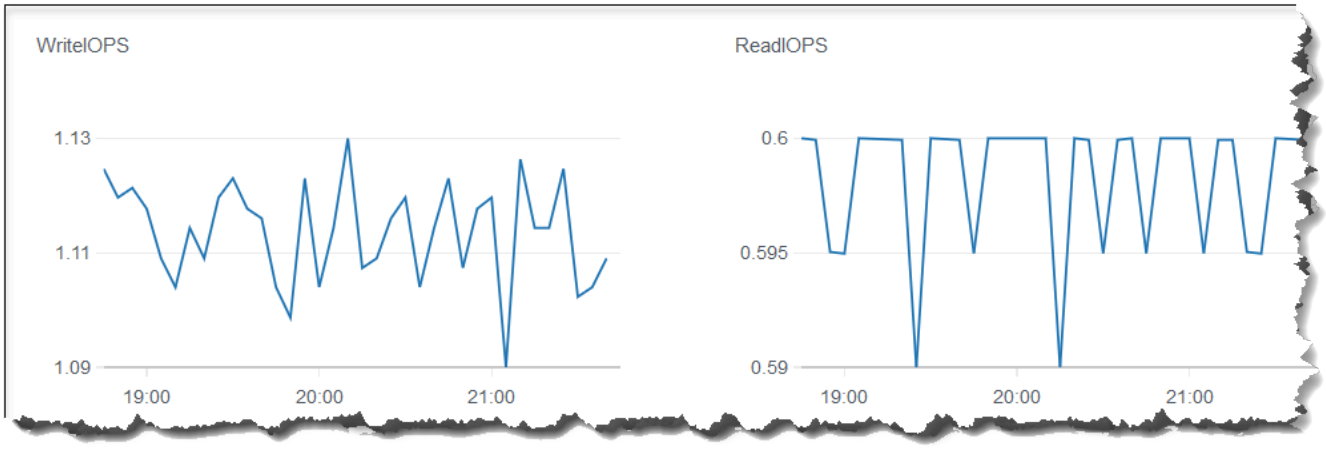
   > **Tip**
   >
   > Jika Anda tidak melihat panel navigasi pada sisi kiri layar Anda, pilih ikon menu (≡) di sudut kiri atas halaman.

3. Di kotak navigasi Klaster, Anda akan melihat kolom Pengidentifikasi klaster. Instance Anda tercantum di bawah cluster, mirip dengan tangkapan layar berikut.



4. Dari daftar instans, pilih nama instans yang Anda inginkan metriknya.
5. Di halaman ringkasan instans yang dihasilkan, pilih tab Pemantauan untuk melihat representasi grafis dari metrik instans Amazon DocumentDB Anda. Karena grafik harus dibuat untuk setiap metrik, mungkin perlu beberapa menit untuk mengisi CloudWatchgrafik.

   Gambar berikut menunjukkan representasi grafis dari dua CloudWatch metrik di konsol Amazon DocumentDB, dan. `WriteIOPS` `ReadIOPS`



Using the CloudWatch Management Console

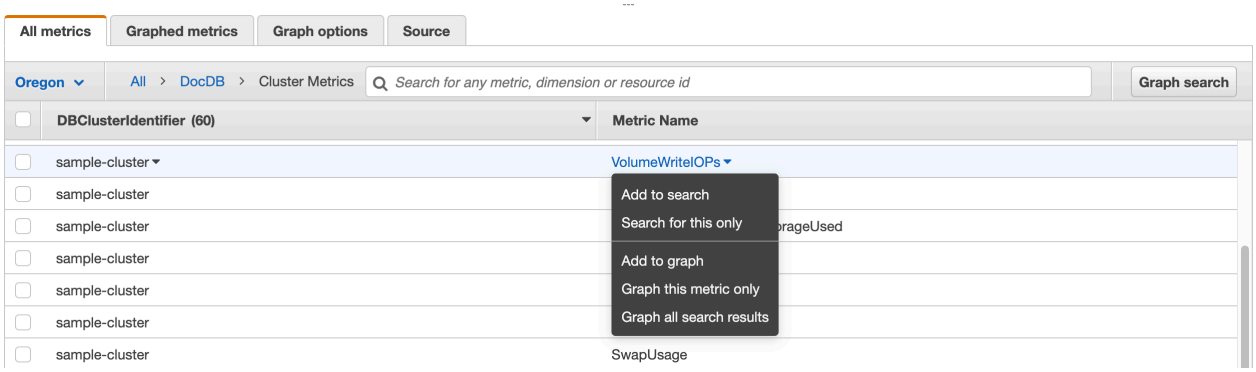
Untuk melihat CloudWatch metrik menggunakan CloudWatch Management Console, selesaikan langkah-langkah berikut.

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/cloudwatch
2. Pada panel navigasi, silakan pilih Metrik. Kemudian, dari daftar nama layanan, pilih DocDB.
3. Pilih dimensi metrik (misalnya, Metrik Klaster).
4. Tab Semua metrik menampilkan semua metrik untuk dimensi tersebut di DocDB.

    a. Untuk menyortir tabel, gunakan judul kolomnya.

    b. Untuk membuat grafik sebuah metrik, pilih kotak centang di sebelah metrik. Untuk memilih semua metrik, pilih kotak centang di baris judul tabel.

    c. Untuk memfilter menurut metrik, arahkan kursor ke nama metrik dan pilih panah tarik-turun di sebelah nama metrik. Kemudian, pilih Tambahkan ke pencarian, seperti yang ditunjukkan pada gambar berikut.



Using the AWS CLI

Untuk melihat CloudWatch data Amazon DocumentDB, gunakan operasi dengan CloudWatch `get-metric-statistics` parameter berikut.

Parameter

- **`--namespace`** — Diperlukan. Namespace layanan yang Anda inginkan CloudWatch metriknya. Untuk Amazon DocumentDB, ini harus `AWS/DocDB`.
- **`--metric-name`** — Diperlukan. Nama metrik yang Anda inginkan datanya.
- **`--start-time`** — Diperlukan. Stempel waktu yang menentukan titik data pertama untuk dikembalikan.

  Nilai yang ditentukan bersifat inklusif; hasil termasuk titik data dengan stempel waktu yang ditentukan. Stempel waktu harus dalam format ISO 8601 UTC (misalnya, 2016-10-03). T23:00:00Z

- **`--end-time`** — Diperlukan. Stempel waktu yang menentukan titik data terakhir untuk dikembalikan.

  Nilai yang ditentukan bersifat inklusif; hasil termasuk titik data dengan stempel waktu yang ditentukan. Stempel waktu harus dalam format ISO 8601 UTC (misalnya, 2016-10-03). T23:00:00Z
- **`--period`** — Diperlukan. Perincian, dalam detik, dari titik data yang dikembalikan. Untuk metrik dengan resolusi reguler, periode dapat sesingkat satu menit (60 detik) dan harus dalam kelipatan 60. Untuk metrik resolusi tinggi yang dikumpulkan dengan interval kurang dari satu menit, periode dapat selama 1, 5, 10, 30, 60, atau kelipatan 60.
- **`--dimensions`**— Opsional. Jika metrik berisi beberapa dimensi, Anda harus menyertakan nilai untuk setiap dimensi. CloudWatch memperlakukan setiap kombinasi dimensi yang unik sebagai metrik terpisah. Jika kombinasi dimensi tertentu tidak dipublikasikan, Anda tidak dapat mengambil statistiknya. Anda harus menentukan dimensi yang sama yang digunakan saat metrik dibuat.
- **`--statistics`** — Opsional. Statistik metrik, selain persentil. Untuk statistik persentil, gunakan `ExtendedStatistics`. Ketika memanggil `GetMetricStatistics`, Anda harus menentukan salah satu `Statistics` atau `ExtendedStatistics`, tapi tidak keduanya.

  Nilai yang diizinkan:

  - `SampleCount`
  - `Average`
  - `Sum`
  - `Minimum`
  - `Maximum`
- **`--extended-statistics`** — Opsional. Statistik `percentile`. Tentukan nilai antara p0.0 dan p100. Ketika memanggil `GetMetricStatistics`, Anda harus menentukan salah satu `Statistics` atau `ExtendedStatistics`, tapi tidak keduanya.
- **`--unit`** — Opsional. Unit untuk metrik tertentu. Metrik dapat dilaporkan dalam beberapa unit. Tidak menyediakan sebuah unit mengakibatkan semua unit dikembalikan. Jika Anda hanya menentukan unit yang tidak dilaporkan oleh metrik, hasil panggilan adalah null.

  Kemungkinan nilai:

  - `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

Contoh berikut menemukan `CPUUtilization` maksimum untuk periode 2 jam dengan mengambil sampel setiap 60 detik.

Untuk Linux, macOS, atau 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
```

Untuk 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
```

Output dari operasi ini terlihat seperti berikut.

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

Dimensi Amazon DocumentDB

Metrik untuk Amazon DocumentDB dikualifikasikan berdasarkan nilai untuk akun atau operasi. Anda dapat menggunakan CloudWatch konsol untuk mengambil data Amazon DocumentDB yang difilter oleh salah satu dimensi dalam tabel berikut.

Dimensi	Deskripsi
`DBClusterIdentifier`	Memfilter data yang Anda minta untuk klaster Amazon DocumentDB tertentu.
`DBClusterIdentifier, Role`	Memfilter data yang Anda minta untuk klaster Amazon DocumentDB tertentu, menggabun gkan metrik berdasarkan peran instance (). WRITER/READER Misalnya, Anda dapat mengagregasikan metrik-metrik untuk semua instans PEMBACA milik sebuah klaster.
`DBInstanceIdentifier`	Memfilter data yang Anda minta untuk instans basis data tertentu.

Memantau metrik Opcounter

Metrik opcounter memiliki nilai bukan nol (biasanya ~50) untuk klaster siaga. Hal ini karena Amazon DocumentDB melakukan pemeriksaan kesehatan berkala, operasi internal, dan tugas pengumpulan metrik.

Memantau koneksi database

Saat Anda melihat jumlah koneksi dengan menggunakan perintah mesin database sepertidb.runCommand({ serverStatus: 1 }), Anda mungkin melihat hingga 10 koneksi

lebih banyak daripada yang Anda `DatabaseConnections` lihat CloudWatch. Hal ini terjadi karena Amazon DocumentDB melakukan pemeriksaan kesehatan berkala dan tugas pengumpulan metrik yang tidak diperhitungkan dalam `DatabaseConnections`. `DatabaseConnections` mewakili koneksi yang diprakarsai pelanggan saja.

Mencatat panggilan API Amazon DocumentDB dengan AWS CloudTrail

Amazon DocumentDB (dengan kompatibilitas MongoDB) terintegrasi AWS CloudTrail dengan, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau layanan di AWS Amazon DocumentDB (dengan kompatibilitas MongoDB). CloudTrail menangkap semua panggilan AWS CLI API untuk Amazon DocumentDB sebagai peristiwa, termasuk panggilan dari konsol Amazon DocumentDB dan dari panggilan kode ke Amazon DocumentDB SDK. Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail acara secara terus menerus ke bucket Amazon S3, termasuk acara untuk Amazon DocumentDB. Jika Anda tidak mengonfigurasi jejak, Anda masih dapat melihat peristiwa terbaru di CloudTrail konsol dalam Riwayat acara. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat ke Amazon DocumentDB (dengan kompatibilitas MongoDB), alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail lainnya.

> **Important**
>
> Untuk fitur pengelolaan tertentu, Amazon DocumentDB menggunakan teknologi operasional yang dibagi dengan Amazon Relational Database Service (Amazon RDS). Konsol Amazon DocumentDB AWS CLI,, dan panggilan API dicatat sebagai panggilan yang dilakukan ke Amazon RDS API.

Untuk mempelajari selengkapnya AWS CloudTrail, lihat Panduan AWS CloudTrail Pengguna.

Informasi Amazon DocumentDB di CloudTrail

CloudTrail diaktifkan pada Akun AWS saat Anda membuat akun. Ketika aktivitas terjadi di Amazon DocumentDB (dengan kompatibilitas MongoDB), aktivitas tersebut direkam dalam CloudTrail suatu peristiwa bersama dengan peristiwa layanan lainnya dalam riwayat Acara. AWS Anda dapat melihat, mencari, dan mengunduh acara terbaru di situs Anda Akun AWS. Untuk informasi selengkapnya, lihat Melihat Acara dengan Riwayat CloudTrail Acara.

Untuk catatan peristiwa yang sedang berlangsung di Anda Akun AWS, termasuk acara untuk Amazon DocumentDB (dengan kompatibilitas MongoDB), buat jejak. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Secara default, saat Anda membuat jejak di konsol, jejak tersebut berlaku untuk semua Wilayah AWS. Jejak mencatat peristiwa dari semua Wilayah di partisi AWS dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi AWS layanan lain untuk menganalisis lebih lanjut dan menindaklanjuti data peristiwa yang dikumpulkan dalam CloudTrail log. Untuk informasi selengkapnya, lihat topik berikut di Panduan Pengguna AWS CloudTrail :

- Gambaran Umum untuk Membuat Jejak
- CloudTrail Layanan dan Integrasi yang Didukung
- Mengkonfigurasi Notifikasi Amazon SNS untuk CloudTrail
- Menerima File CloudTrail Log dari Beberapa Wilayah
- Menerima File CloudTrail Log dari Beberapa Akun

Setiap entri kejadian atau log menyertakan informasi tentang siapa yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan hal berikut ini:

- Baik permintaan tersebut dibuat dengan kredensial pengguna atau root.
- Apakah permintaan tersebut dibuat dengan kredensial keamanan sementara untuk satu peran atau pengguna gabungan.
- Apakah permintaan itu dibuat oleh AWS layanan lain.

Untuk informasi lain, lihat Elemen userIdentity CloudTrail.

Membuat profil operasi Amazon DocumentDB

Anda dapat menggunakan profiler di Amazon DocumentDB (dengan kompatibilitas MongoDB) untuk mencatat waktu eksekusi dan detail operasi yang dilakukan di klaster Anda. Profiler berguna untuk memantau operasi paling lambat di klaster Anda untuk membantu Anda meningkatkan performa kueri individual dan performa klaster secara keseluruhan.

Secara default, fitur profiler dinonaktifkan. Saat diaktifkan, profiler mencatat operasi yang memakan waktu lebih lama dari nilai ambang batas yang ditentukan pelanggan (misalnya, 100 ms) ke Amazon Logs. CloudWatch Detail yang dicatat termasuk perintah yang diprofilkan, waktu, ringkasan rencana,

dan metadata klien. Setelah operasi dicatat ke CloudWatch Log, Anda dapat menggunakan Wawasan CloudWatch Log untuk menganalisis, memantau, dan mengarsipkan data profil Amazon DocumentDB Anda. Kueri umum disediakan di bagian Pertanyaan umum.

Saat diaktifkan, profiler menggunakan sumber daya tambahan di klaster Anda. Kami merekomendasikan Anda memulai dengan nilai ambang batas yang tinggi (misalnya, 500 milidetik) dan secara bertahap menurunkan nilai tersebut untuk mengidentifikasi operasi yang lambat. Memulai dengan nilai ambang 50 milidetik dapat menyebabkan masalah performa pada klaster Anda untuk aplikasi dengan throughput tinggi. Profiler diaktifkan di tingkat klaster dan berfungsi pada semua instans dan basis data dalam klaster. Amazon DocumentDB mencatat operasi ke CloudWatch Amazon Logs dengan upaya terbaik.

Meskipun Amazon DocumentDB tidak mengenakan biaya tambahan untuk mengaktifkan profiler, Anda dikenakan tarif standar untuk penggunaan Log. CloudWatch Untuk informasi tentang harga CloudWatch Log, lihat CloudWatch harga Amazon.

Topik

- Operasi yang didukung
- Batasan
- Mengaktifkan profiler Amazon DocumentDB
- Menonaktifkan profiler Amazon DocumentDB
- Menonaktifkan ekspor log profiler
- Mengakses log profiler Amazon DocumentDB Anda
- Pertanyaan umum

Operasi yang didukung

Profiler Amazon DocumentDB mendukung operasi berikut:

- `aggregate`
- `count`
- `delete`
- `distinct`
- `find` (OP_QUERY dan perintah)
- `findAndModify`

- `insert`
- `update`

Batasan

Profiler kueri lambat hanya dapat memancarkan log profiler jika seluruh set hasil kueri dapat ditampung dalam satu batch, dan jika set hasil di bawah 16 MB (ukuran BSON maksimum). Set hasil yang lebih besar dari 16MB secara otomatis dibagi menjadi beberapa batch.

Sebagian besar driver atau shell dapat menetapkan ukuran batch default yang kecil. Anda dapat menentukan ukuran batch sebagai bagian dari kueri Anda. Untuk tujuan menangkap log kueri lambat, kami merekomendasikan ukuran batch yang melebihi ukuran set hasil yang Anda harapkan. Jika Anda tidak yakin dengan ukuran set hasil, atau jika bervariasi, Anda juga dapat mengatur ukuran batch ke jumlah besar (mis., 100rb).

Namun, menggunakan ukuran batch yang lebih besar berarti lebih banyak hasil yang harus diambil dari basis data sebelum respons dikirim ke klien. Untuk beberapa kueri, hal itu dapat membuat penundaan lebih lama sebelum Anda mendapatkan hasil. Jika Anda tidak berencana untuk mengkonsumsi seluruh set hasil, ada kemungkinan bahwa Anda akan menghabiskan lebih banyak I/Os untuk memproses kueri dan membuang hasilnya.

Mengaktifkan profiler Amazon DocumentDB

Mengaktifkan profiler pada sebuah klaster adalah proses tiga langkah. Pastikan bahwa semua langkah selesai, atau log profil tidak akan dikirim ke CloudWatch Log. Profiler diatur pada tingkat klaster dan dilakukan pada semua basis data dan intans klaster.

Untuk mengaktifkan profiler di klaster

1. Karena Anda tidak dapat mengubah grup parameter klaster default, pastikan Anda memiliki grup parameter klaster kustom yang tersedia. Untuk informasi selengkapnya, lihat Membuat grup parameter cluster Amazon DocumentDB.
2. Dengan menggunakan grup parameter klaster kustom yang tersedia, ubah parameter berikut: `profiler`, `profiler_threshold_ms`, dan `profiler_sampling_rate`. Untuk informasi selengkapnya, lihat Memodifikasi grup parameter cluster Amazon DocumentDB.
3. Buat atau modifikasi cluster Anda untuk menggunakan grup parameter cluster kustom dan untuk mengaktifkan mengekspor `profiler` CloudWatch log ke Log.

Bagian berikut menunjukkan bagaimana menerapkan langkah-langkah ini menggunakan Konsol Manajemen AWS dan AWS Command Line Interface (AWS CLI).

Using the Konsol Manajemen AWS

1. Sebelum Anda mulai, buat klaster Amazon DocumentDB dan grup parameter klaster kustom jika Anda belum memilikinya. Untuk informasi selengkapnya, lihat Membuat grup parameter cluster Amazon DocumentDB dan Membuat cluster Amazon DocumentDB.
2. Dengan menggunakan grup parameter klaster kustom yang tersedia, ubah parameter berikut. Untuk informasi selengkapnya, lihat Memodifikasi grup parameter cluster Amazon DocumentDB.

   - `profiler` — Mengaktifkan atau menonaktifkan pembuatan profil kueri. Nilai yang diizinkan adalah `enabled` dan `disabled`. Nilai default-nya adalah `disabled`. Untuk mengaktifkan pembuatan profil, atur nilainya ke `enabled`.
   - `profiler_threshold_ms`— Ketika `profiler` diatur ke`enabled`, semua perintah yang memakan waktu lebih lama dari `profiler_threshold_ms` yang dicatat CloudWatch. Nilai yang diizinkan adalah `[50-INT_MAX]`. Nilai default-nya adalah `100`.
   - `profiler_sampling_rate` — Bagian dari operasi lambat yang harus diprofilkan atau dicatat. Nilai yang diizinkan adalah `[0.0-1.0]`. Nilai default-nya adalah `1.0`.
3. Ubah klaster Anda untuk menggunakan grup parameter klaster kustom dan setel ekspor log profiler untuk dipublikasikan ke Amazon. CloudWatch

   a. Di panel navigasi, pilih Klaster untuk menambahkan grup parameter kustom Anda ke sebuah klaster.

   b. Pilih tombol di sebelah kiri nama klaster yang ingin Anda kaitkan dengan grup parameter Anda. Pilih Tindakan, lalu Modifikasi untuk memodifikasi klaster Anda.

   c. Di bawah Opsi klaster, pilih grup parameter kustom dari langkah di atas untuk menambahkannya ke klaster Anda.

   d. Di bawah Ekspor log, pilih Log profiler untuk dipublikasikan ke Amazon. CloudWatch

   e. Pilih Lanjutkan dan lihat ringkasan dari modifikasi Anda.

   f. Setelah memverifikasi perubahan Anda, Anda dapat menerapkannya segera atau selama jendela pemeliharaan berikutnya di bawah Penjadwalan modifikasi .

   g. Pilih Ubah klaster untuk memperbarui klaster Anda dengan grup parameter baru Anda.

Using the AWS CLI

Prosedur berikut mengaktifkan profiler pada semua operasi yang didukung untuk klaster `sample-cluster`.

1. Sebelum Anda mulai, pastikan bahwa Anda memiliki grup parameter klaster kustom yang tersedia dengan menjalankan perintah berikut, dan meninjau output untuk grup parameter klaster yang tidak memiliki `default` dalam nama dan memiliki `docdb3.6` sebagai keluarga grup parameter. Jika Anda tidak memiliki grup parameter klaster bukan-default, lihat Membuat grup parameter cluster Amazon DocumentDB.

```
aws docdb describe-db-cluster-parameter-groups \
    --query 'DBClusterParameterGroups[*].
[DBClusterParameterGroupName,DBParameterGroupFamily]'
```

   Pada output berikut, hanya `sample-parameter-group` yang memenuhi kedua kriteria.

```
[
       [
           "default.docdb3.6",
           "docdb3.6"
       ],
       [
           "sample-parameter-group",
           "docdb3.6"
       ]
]
```

2. Dengan menggunakan grup parameter klaster kustom Anda, ubah parameter berikut:

   - `profiler` — Mengaktifkan atau menonaktifkan pembuatan profil kueri. Nilai yang diizinkan adalah `enabled` dan `disabled`. Nilai default-nya adalah `disabled`. Untuk mengaktifkan pembuatan profil, atur nilainya ke `enabled`.
   - `profiler_threshold_ms`— Ketika `profiler` diatur ke`enabled`, semua perintah membutuhkan waktu lebih lama dari `profiler_threshold_ms` yang dicatat CloudWatch. Nilai yang diizinkan adalah `[50-INT_MAX]`. Nilai default-nya adalah `100`.
   - `profiler_sampling_rate` — Bagian dari operasi lambat yang harus diprofilkan atau dicatat. Nilai yang diizinkan adalah `[0.0-1.0]`. Nilai default-nya adalah `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. Ubah klaster Amazon DocumentDB Anda sehingga menggunakan grup parameter klaster kustom `sample-parameter-group` dari langkah sebelumnya dan atur parameter `--enable-cloudwatch-logs-exports` ke `profiler`.

   Kode berikut memodifikasi cluster `sample-cluster` untuk menggunakan `sample-parameter-group` dari langkah sebelumnya, dan `profiler` menambah ekspor CloudWatch Log yang diaktifkan.

   ```
   aws docdb modify-db-cluster \
          --db-cluster-identifier sample-cluster \
          --db-cluster-parameter-group-name sample-parameter-group \
          --cloudwatch-logs-export-configuration '{"EnableLogTypes":["profiler"]}'
   ```

   Output dari operasi ini akan terlihat seperti berikut.

   ```
   {
       "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
    }
}
```

Menonaktifkan profiler Amazon DocumentDB

Untuk menonaktifkan profiler, Anda menonaktifkan `profiler` parameter dan ekspor `profiler` log ke CloudWatch Log.

Menonaktifkan Profiler

Anda dapat menonaktifkan `profiler` parameter menggunakan Konsol Manajemen AWS atau AWS CLI, sebagai berikut.

Using the Konsol Manajemen AWS

Prosedur berikut menggunakan Konsol Manajemen AWS untuk menonaktifkan Amazon `profiler` DocumentDB.

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Grup parameter. Kemudian pilih nama grup parameter klaster yang ingin Anda nonaktifkan profilernya.
3. Di halaman Parameter klaster yang dihasilkan, pilih tombol di sebelah kiri parameter `profiler` dan pilih Edit.
4. Di kotak dialog Ubah profiler, pilih `disabled` di dalam daftar.
5. Pilih Ubah parameter klaster.

Using the AWS CLI

Untuk menonaktifkan `profiler` pada klaster menggunakan AWS CLI, modifikasi klaster sebagai berikut.

```
aws docdb modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name sample-parameter-group \
    --parameters
 ParameterName=profiler,ParameterValue=disabled,ApplyMethod=immediate
```

Menonaktifkan ekspor log profiler

Anda dapat menonaktifkan mengekspor `profiler` CloudWatch log ke Log dengan menggunakan salah satu Konsol Manajemen AWS atau AWS CLI, sebagai berikut.

Using the Konsol Manajemen AWS

Prosedur berikut menggunakan Konsol Manajemen AWS untuk menonaktifkan Amazon DocumentDB mengekspor log ke. CloudWatch

1. Buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Pada panel navigasi, silakan pilih Klaster. Pilih tombol di sebelah kiri nama klaster yang log ekspornya ingin Anda nonaktifkan.
3. Pada menu Tindakan, pilih Modifikasi.
4. Gulir ke bawah ke bagian Ekspor log dan batalkan pilihan Log profiler.
5. Pilih Lanjutkan.
6. Tinjau perubahan Anda, lalu pilih kapan Anda ingin perubahan ini diterapkan ke klaster Anda:
   - Terapkan selama jendela pemeliharaan terjadwal berikutnya
   - Terapkan segera
7. Pilih Ubah klaster.

Using the AWS CLI

Kode berikut memodifikasi cluster `sample-cluster` dan menonaktifkan log CloudWatch profiler.

Example

Untuk Linux, macOS, atau Unix:

```
aws docdb modify-db-cluster \
   --db-cluster-identifier sample-cluster \
   --cloudwatch-logs-export-configuration '{"DisableLogTypes":["profiler"]}'
```

Untuk Windows:

```
aws docdb modify-db-cluster ^
   --db-cluster-identifier sample-cluster ^
   --cloudwatch-logs-export-configuration '{"DisableLogTypes":["profiler"]}'
```

Output dari operasi ini terlihat seperti berikut.

```
{
```

```
    "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
    }
}
```

Mengakses log profiler Amazon DocumentDB Anda

Ikuti langkah-langkah ini untuk mengakses log profil Anda di Amazon CloudWatch.

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pastikan Anda berada di Wilayah yang sama dengan klaster Amazon DocumentDB Anda.
3. Pilih Log di panel navigasi.
4. Untuk menemukan log profiler untuk klaster Anda, dalam daftar, pilih `/aws/docdb/`*`yourClusterName`*`/profiler`.

   Log profil untuk setiap instans Anda tersedia di bawah setiap nama instans masing-masing.

Pertanyaan umum

Berikut ini adalah beberapa kueri umum yang dapat Anda gunakan untuk menganalisis perintah yang diprofilkan. Untuk informasi selengkapnya tentang Wawasan CloudWatch Log, lihat Menganalisis Data Log dengan Wawasan CloudWatch Log dan Kueri Contoh.

Dapatkan 10 operasi paling lambat pada koleksi tertentu

```
filter ns="test.foo" | sort millis desc | limit 10
```

Dapatkan semua operasi pembaruan pada koleksi yang memakan waktu lebih dari 60 ms

```
filter millis > 60 and op = "update"
```

Dapatkan 10 operasi paling lambat dalam sebulan terakhir

```
sort millis desc | limit 10
```

Dapatkan semua pertanyaan dengan ringkasan paket COLLSCAN

```
filter planSummary="COLLSCAN"
```

Memantau dengan Wawasan Kinerja

Performance Insights menambahkan fitur pemantauan Amazon DocumentDB yang ada untuk menggambarkan kinerja klaster Anda dan membantu Anda menganalisis masalah apa pun yang mempengaruhinya. Dengan dasbor Performance Insights, Anda dapat memvisualisasikan pemuatan database dan memfilter beban berdasarkan menunggu, pernyataan kueri, host, atau aplikasi.

> **Note**
>
> Performance Insights hanya tersedia untuk klaster berbasis instans Amazon DocumentDB 3.6, 4.0, 5.0, dan 8.0.

Bagaimana ini berguna?

- Visualisasikan kinerja database - Visualisasikan beban untuk menentukan kapan dan di mana beban berada di database
- Tentukan apa yang menyebabkan beban pada database — Tentukan kueri, host, dan aplikasi mana yang berkontribusi terhadap pemuatan instans Anda
- Tentukan kapan ada pemuatan pada database Anda — Perbesar dasbor Performance Insights untuk fokus pada peristiwa tertentu atau memperkecil untuk melihat tren di rentang waktu yang lebih besar
- Peringatan tentang pemuatan basis data - Akses metrik pemuatan basis data baru secara otomatis dari CloudWatch tempat Anda dapat memantau metrik pemuatan DB bersama metrik Amazon DocumentDB lainnya dan setel peringatan padanya

Apa batasan Amazon DocumentDB Performance Insights?

- Performance Insights di wilayah AWS GovCloud (US-East) dan AWS GovCloud (US-West) tidak tersedia
- Performance Insights untuk Amazon DocumentDB mempertahankan data kinerja hingga 7 hari
- Kueri yang lebih panjang dari 1.024 byte tidak digabungkan dalam Performance Insights

Topik

- Konsep Performance Insights
- Mengaktifkan dan menonaktifkan Wawasan Kinerja
- Mengonfigurasi kebijakan akses untuk Wawasan Performa
- Menganalisis metrik dengan dasbor Wawasan Performa
- Mengambil metrik dengan API Wawasan Performa
- CloudWatch Metrik Amazon untuk Performance Insights
- Performance Insights untuk metrik penghitung

Konsep Performance Insights

Topik

- Sesi aktif rata-rata
- Dimensi
- Maks vCPU

Sesi aktif rata-rata

Beban basis data (beban DB) mengukur tingkat aktivitas dalam database Anda. Metrik utama dalam Wawasan Performa adalah `DB Load`, yang dikumpulkan setiap detik. Unit untuk `DBLoad` metrik adalah Average Active Sessions (AAS) untuk instance Amazon DocumentDB.

Sesi aktif adalah koneksi yang telah mengirimkan pekerjaan ke instance Amazon DocumentDB dan sedang menunggu tanggapan. Misalnya, jika Anda mengirimkan kueri ke instance Amazon DocumentDB, sesi database aktif saat instance memproses kueri.

Untuk mendapatkan sesi aktif rata-rata, Wawasan Kinerja mengambil sampel jumlah sesi yang secara bersamaan menjalankan kueri. AAS adalah jumlah total sesi dibagi dengan jumlah total sampel. Tabel berikut menunjukkan lima sampel berturut-turut dari query berjalan.

Sampel	Jumlah sesi yang menjalankan kueri	AAS	Penghitungan
1	2	2	2 sesi / 1 sampel
2	0	1	2 sesi / 2 sampel
3	4	2	6 sesi / 3 sampel
4	0	1.5	6 sesi / 4 sampel
5	4	2	10 sesi / 5 sampel

Pada contoh sebelumnya, Beban DB untuk interval waktu dari 1-5 adalah 2 AAS. Peningkatan beban DB berarti, rata-rata, lebih banyak sesi berjalan di database.

Dimensi

Metrik `DB Load` berbeda dari metrik seri waktu lainnya karena Anda dapat membaginya menjadi subkomponen yang disebut dimensi. Anda dapat menganggap dimensi sebagai kategori untuk karakteristik `DB Load` metrik yang berbeda. Saat Anda mendiagnosis masalah kinerja, dimensi yang paling berguna adalah status tunggu dan kueri teratas.

tunggu negara

Status tunggu menyebabkan pernyataan kueri menunggu peristiwa tertentu terjadi sebelum dapat terus berjalan. Misalnya, pernyataan kueri mungkin menunggu hingga sumber daya yang terkunci dibuka. Dengan menggabungkan `DB Load` dengan status tunggu, Anda bisa mendapatkan gambaran lengkap tentang status sesi. Berikut adalah berbagai status tunggu Amazon DocumentDB:

Status tunggu Amazon DocumentDB	Tunggu Deskripsi Negara
Kait	Status tunggu Latch terjadi saat sesi menunggu halaman kumpulan buffer. Sering paging masuk dan keluar dari buffer pool dapat terjadi lebih sering ketika sering ada kueri besar yang diproses oleh sistem, pemindaian koleksi,

Status tunggu Amazon DocumentDB	Tunggu Deskripsi Negara
	atau ketika kumpulan buffer terlalu kecil untuk menangani set kerja.
CPU	Status menunggu CPU terjadi ketika sesi sedang menunggu pada CPU.
CollectionLock	Status CollectionLock tunggu terjadi ketika sesi menunggu untuk mendapatkan kunci pada koleksi. Peristiwa ini terjadi ketika ada operasi DDL pada koleksi.
DocumentLock	Status DocumentLock tunggu terjadi ketika sesi menunggu untuk memperoleh kunci pada dokumen. Jumlah penulisan bersamaan yang tinggi ke dokumen yang sama akan berkontri busi pada lebih banyak status DocumentLock menunggu pada dokumen itu.
SystemLock	Status SystemLock tunggu terjadi ketika sesi sedang menunggu di sistem. Hal ini dapat terjadi ketika sering ada query berjalan lama, transaksi berjalan lama, atau konkurensi tinggi pada sistem.
IO	Status tunggu IO terjadi ketika sesi menunggu IO selesai.
BufferLock	Status BufferLock tunggu terjadi ketika sesi menunggu untuk memperoleh kunci pada halaman bersama di buffer. BufferLockstatus tunggu dapat diperpanjang jika proses lain menahan kursor terbuka pada halaman yang diminta.

Status tunggu Amazon DocumentDB	Tunggu Deskripsi Negara
LowMemThrottle	Status LowMemThrottle tunggu terjadi ketika sesi menunggu karena tekanan memori yang berat pada instance Amazon DocumentDB. Jika status ini berlanjut untuk waktu yang lama, pertimbangkan untuk meningkatkan instance untuk menyediakan memori tambahan. Untuk informasi lebih lanjut, lihat Gubernur Sumber Daya.
BackgroundActivity	Status BackgroundActivity tunggu terjadi ketika sesi sedang menunggu proses sistem internal.
Lainnya	Status tunggu lainnya adalah keadaan tunggu internal. Jika status ini bertahan untuk waktu yang lama, pertimbangkan untuk menghentikan kueri ini. Untuk informasi selengkapnya, lihat Bagaimana Cara Menemukan dan Menghentikan Kueri Berjalan Lama atau Diblokir?

Kueri teratas

Sementara status tunggu menunjukkan kemacetan, kueri teratas menunjukkan kueri mana yang paling berkontribusi pada pemuatan DB. Misalnya, saat ini mungkin ada banyak kueri yang berjalan di database, tetapi kueri tunggal mungkin menggunakan 99% dari muatan DB. Dalam hal ini, muatan tinggi mungkin menunjukkan masalah dalam kueri.

Maks vCPU

Di dasbor, bagan Basis data muatan mengumpulkan, menggabungkan, dan menampilkan informasi sesi. Untuk mengetahui apakah sesi aktif melebihi CPU maksimum, lihat hubungannya dengan baris vCPU Maks. Nilai Max vCPU ditentukan oleh jumlah inti vCPU (virtual CPU) untuk instans Amazon DocumentDB Anda.

Jika muatan DB sering melampaui baris vCPU Maks dan status tunggu utamanya adalah CPU, CPU akan kelebihan muatan. Dalam hal ini, Anda mungkin ingin membatasi koneksi ke instance, menyetel kueri apa pun dengan beban CPU yang tinggi, atau mempertimbangkan kelas instance yang lebih

besar. Instans yang tinggi dan konsisten dari setiap status tunggu menunjukkan bahwa mungkin terjadi kemacetan atau masalah pertentangan sumber daya yang perlu diselesaikan. Hal ini bisa terjadi meski muatan DB tidak melampaui baris vCPU Maks.

Mengaktifkan dan menonaktifkan Wawasan Kinerja

Untuk menggunakan Wawasan Kinerja, aktifkan di instans DB Anda. Anda dapat menonaktifkannya nanti jika perlu. Mengaktifkan dan menonaktifkan Wawasan Kinerja tidak menyebabkan waktu henti, boot ulang, atau kegagalan.

Agen Wawasan Performa menggunakan CPU dan memori terbatas di host DB. Ketika muatan DB tinggi, agen membatasi dampak performa dengan mengurangi frekuensi pengumpulan data.

Mengaktifkan Performance Insights saat membuat klaster

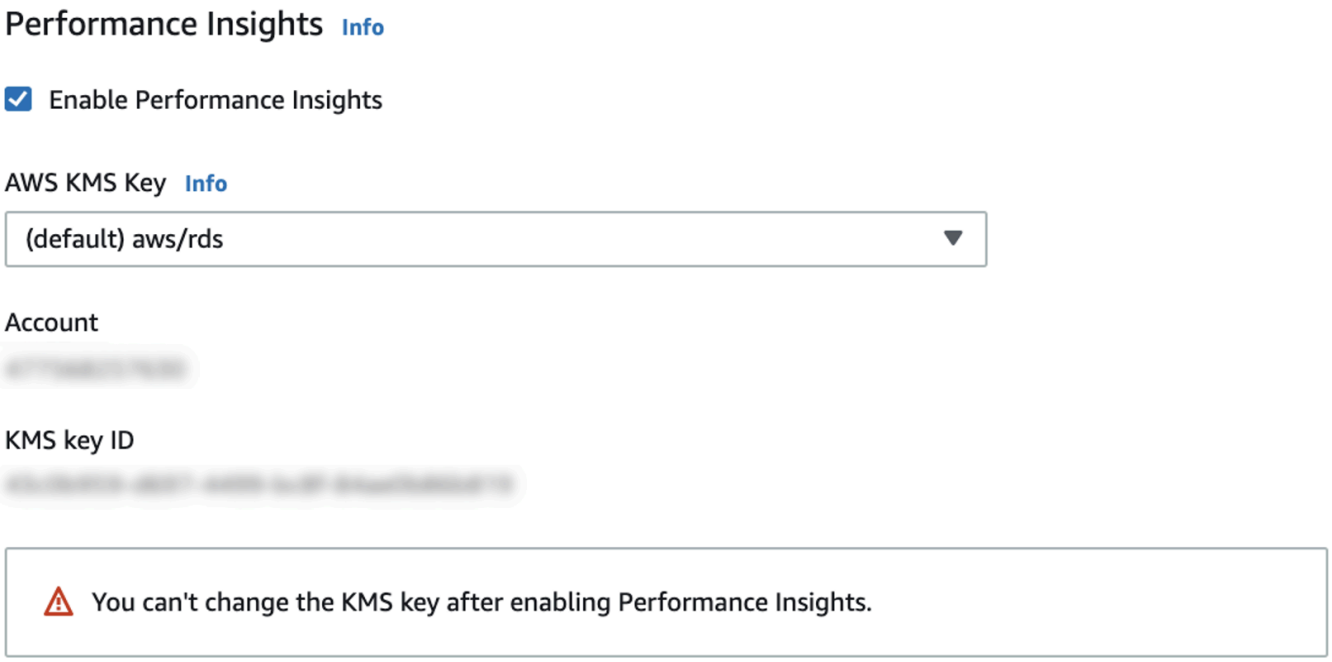
Di konsol, Anda dapat mengaktifkan atau menonaktifkan Wawasan Kinerja saat membuat atau mengubah instans DB baru.

Menggunakan Konsol Manajemen AWS

Di konsol, Anda dapat mengaktifkan Performance Insights saat membuat klaster Amazon DocumentDB. Saat Anda membuat klaster Amazon DocumentDB baru, aktifkan Performance Insights dengan memilih Aktifkan Performance Insights di bagian Performance Insights.

Instruksi konsol

1. Untuk membuat cluster, ikuti petunjuk untuk Membuat cluster Amazon DocumentDB.
2. Pilih Aktifkan Performance Insights di bagian Performance Insights.



> **Note**
>
> Periode retensi data Performance Insights akan berlangsung selama tujuh hari.

AWS KMS kunci — Tentukan kunci AWS KMS Anda. Performance Insights mengenkripsi semua data yang berpotensi sensitif menggunakan kunci Anda. AWS KMS Data dienkripsi saat dipindahkan dan saat tidak aktif. Untuk informasi selengkapnya, lihat Mengonfigurasi AWS AWS KMS kebijakan untuk Performance Insights.

Mengaktifkan dan menonaktifkan saat memodifikasi instance

Anda dapat mengubah instans DB untuk mengaktifkan atau menonaktifkan Performance Insights menggunakan konsol atau. AWS CLI

Using the Konsol Manajemen AWS

Instruksi konsol

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Pilih Klaster.

3. Pilih instans DB, lalu pilih Modifikasi.
4. Di bagian Performance Insights, pilih Aktifkan Performance Insights atau Nonaktifkan Performance Insights.

> **Note**
>
> Jika memilih Aktifkan Performance Insights, Anda dapat menentukan kunci. AWS AWS KMS Performance Insights mengenkripsi semua data yang berpotensi sensitif menggunakan kunci Anda. AWS KMS Data dienkripsi saat dipindahkan dan saat tidak aktif. Untuk informasi selengkapnya, lihat Mengenkripsi Data Amazon DocumentDB saat Istirahat.

5. Pilih Lanjutkan.
6. Untuk Penjadwalan Modifikasi, pilih Langsung terapkan. Jika Anda memilih Terapkan selama jendela pemeliharaan terjadwal berikutnya, instans Anda akan mengabaikan setelan ini dan segera mengaktifkan Performance Insights.
7. Pilih Ubah instans.

Using the AWS CLI

Saat menggunakan `modify-db-instance` AWS AWS CLI perintah `create-db-instance` atau, Anda dapat mengaktifkan Performance Insights dengan menentukan`--enable-performance-insights`, atau menonaktifkannya dengan menentukan. `--no-enable-performance-insights`

Prosedur berikut menjelaskan cara mengaktifkan atau menonaktifkan Wawasan Kinerja untuk instans DB menggunakan AWS AWS CLI.

AWS AWS CLI instruksi

Panggil `modify-db-instance` AWS AWS CLI perintah dan berikan nilai-nilai berikut:

- `--db-instance-identifer`— Nama instans DB
- `--enable-performance-insights` untuk mengaktifkan atau `--no-enable-performance-insights` untuk menonaktifkan

Example

Contoh berikut memungkinkan Performance Insights untuk: `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
```

Mengonfigurasi kebijakan akses untuk Wawasan Performa

Untuk mengakses Wawasan Kinerja, Anda harus memiliki izin yang sesuai dari AWS Identity and Access Management (IAM). Anda memiliki opsi berikut untuk memberikan akses:

- Lampirkan kebijakan `AmazonRDSPerformanceInsightsReadOnly` terkelola ke set izin atau peran.
- Membuat kebijakan IAM kustom dan melampirkannya ke kumpulan izin atau peran.

Selain itu, jika Anda menetapkan kunci terkelola pelanggan saat mengaktifkan Performance Insights, pastikan pengguna di akun memiliki izin `kms:Decrypt` dan `kms:GenerateDataKey` izin pada kunci KMS.

> **Note**
>
> Untuk enkripsi saat istirahat dengan AWS KMS kunci dan manajemen grup keamanan, Amazon DocumentDB memanfaatkan teknologi operasional yang dibagikan dengan Amazon RDS.

Melampirkan AmazonRDSPerformanceInsightsReadOnly kebijakan ke kepala IAM

`AmazonRDSPerformanceInsightsReadOnly`adalah kebijakan AWS terkelola yang memberikan akses ke semua operasi hanya-baca dari Amazon DocumentDB Performance Insights API. Saat ini, semua operasi di API ini hanya-baca. Jika Anda melampirkan `AmazonRDSPerformanceInsightsReadOnly` ke kumpulan izin atau peran, penerima dapat menggunakan Wawasan Performa beserta fitur konsol lainnya.

Membuat kebijakan IAM kustom untuk Wawasan Performa

Untuk pengguna yang tidak memiliki `AmazonRDSPerformanceInsightsReadOnly` kebijakan, Anda dapat memberikan akses ke Performance Insights dengan membuat atau memodifikasi kebijakan IAM yang dikelola pengguna. Jika Anda melampirkan kebijakan ke set izin atau peran, penerima dapat menggunakan Performance Insights.

Untuk membuat kebijakan kustom

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi, pilih Kebijakan.
3. Pilih Buat kebijakan.
4. Di halaman Buat Kebijakan, pilih tab JSON.
5. Salin dan tempel teks berikut, ganti *us-east-1* dengan nama AWS Wilayah Anda dan *111122223333* dengan nomor akun pelanggan Anda.

   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. Pilih Tinjau kebijakan.
7. Berikan nama untuk kebijakan tersebut dan secara opsional deskripsi, lalu pilih Buat kebijakan.

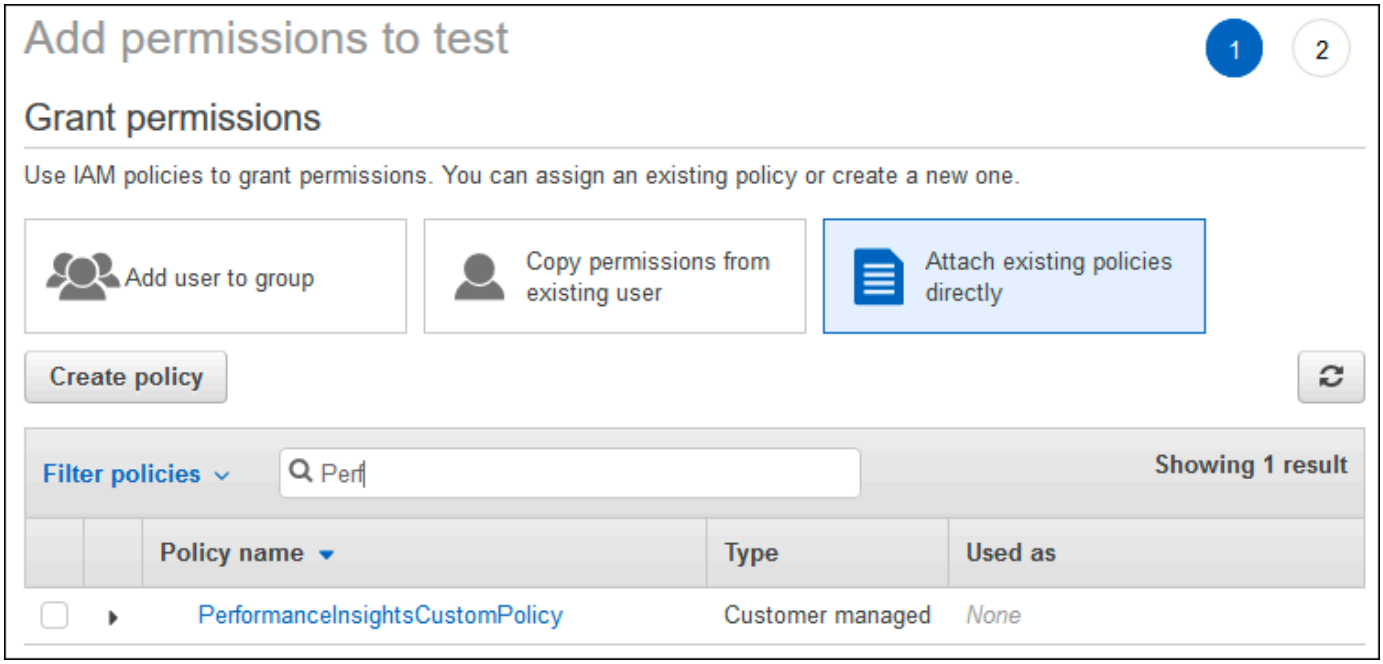
Sekarang, Anda dapat menyisipkan kebijakan ke kumpulan izin atau peran. Prosedur berikut mengasumsikan bahwa Anda sudah memiliki pengguna yang tersedia untuk tujuan ini.

Untuk melampirkan kebijakan ini ke pengguna

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi, pilih Pengguna.
3. Pilih pengguna yang ada dari daftar.

> **Important**
>
> Untuk menggunakan Performance Insights, pastikan Anda memiliki akses ke Amazon DocumentDB selain kebijakan kustom. Misalnya, kebijakan yang AmazonDocDBReadOnlyAccesstelah ditentukan menyediakan akses hanya-baca ke Amazon informasi DocDB.For selengkapnya, lihat Mengelola akses menggunakan kebijakan.

4. Di halaman Ringkasan, pilih Tambahkan izin.
5. Pilih Lampirkan kebijakan yang sudah ada secara langsung. Untuk Pencarian, ketik beberapa karakter pertama dari nama kebijakan Anda, seperti yang ditampilkan di bawah ini.



6. Pilih kebijakan Anda, lalu pilih Berikutnya: Tinjauan.
7. Pilih Tambahkan izin.

Mengkonfigurasi AWS KMS kebijakan untuk Performance Insights

Performance Insights menggunakan perangkat AWS KMS key untuk mengenkripsi data sensitif. Saat mengaktifkan Wawasan Kinerja melalui API atau konsol, Anda akan memiliki opsi berikut:

- Pilih default Kunci yang dikelola AWS.

  Amazon DocumentDB menggunakan Kunci yang dikelola AWS instans DB baru Anda. Amazon DocumentDB membuat untuk Kunci yang dikelola AWS akun Anda. AWS AWS Akun Anda memiliki Amazon DocumentDB yang berbeda Kunci yang dikelola AWS untuk setiap Wilayah. AWS
- Memilih kunci yang dikelola pelanggan.

  Jika Anda menentukan kunci yang dikelola pelanggan, pengguna di akun Anda yang memanggil API Wawasan Performa memerlukan izin `kms:Decrypt` dan `kms:GenerateDataKey` pada kunci KMS. Anda dapat mengonfigurasi izin ini melalui kebijakan IAM. Namun, sebaiknya Anda mengelola izin ini melalui kebijakan kunci KMS Anda. Untuk informasi selengkapnya, lihat Menggunakan kebijakan kunci dalam KMS AWS.

Example

Contoh kebijakan kunci berikut menunjukkan cara menambahkan pernyataan ke kebijakan kunci KMS Anda. Pernyataan ini mengizinkan akses ke Wawasan Performa. Tergantung pada bagaimana Anda menggunakan AWS KMS, Anda mungkin ingin mengubah beberapa batasan. Sebelum menambahkan pernyataan ke kebijakan, hapus semua komentar.

Menganalisis metrik dengan dasbor Wawasan Performa

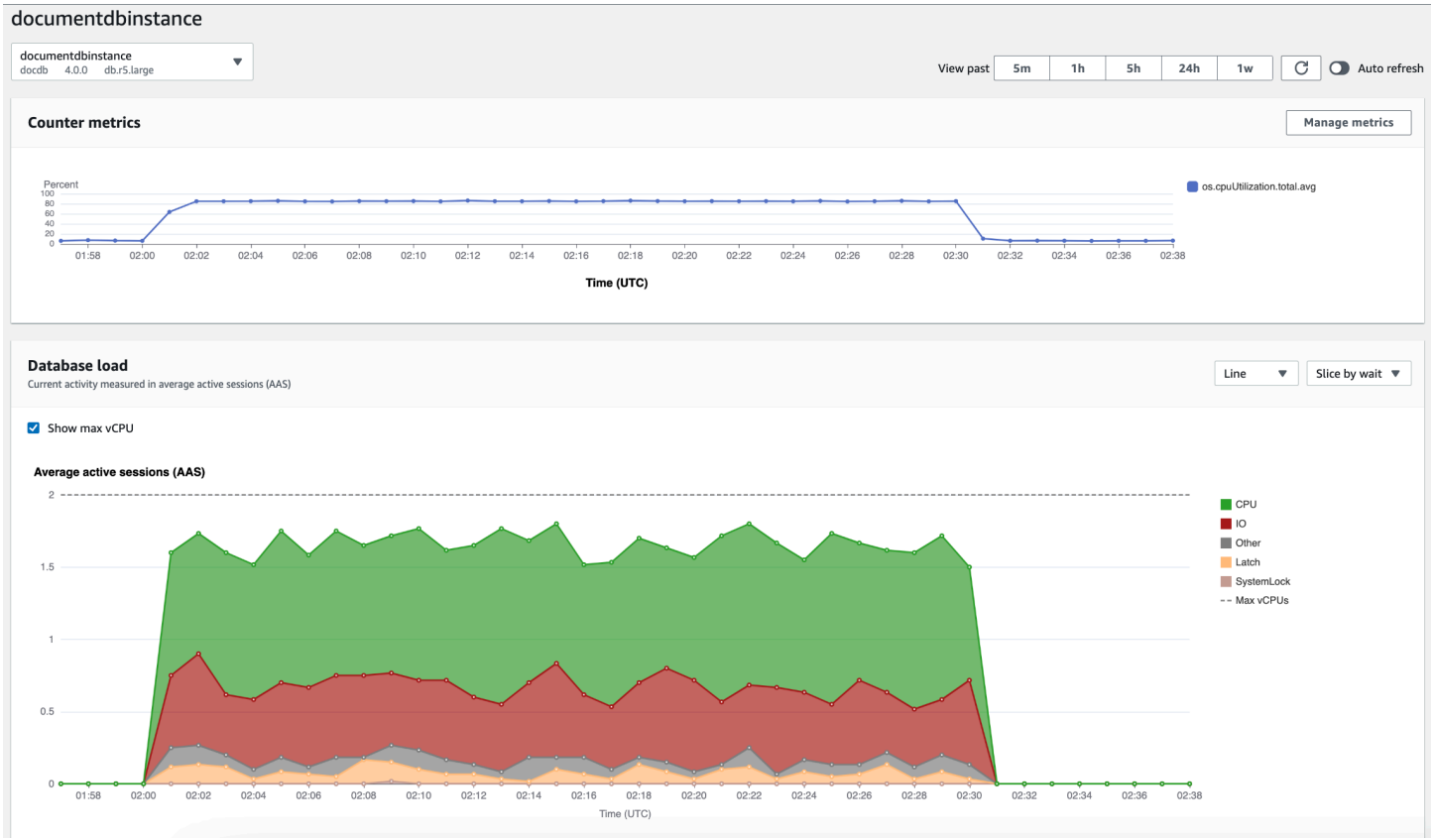
Dasbor Wawasan Performa berisi informasi performa basis data untuk membantu Anda menganalisis dan memecahkan masalah performa. Pada halaman dasbor utama, Anda dapat melihat informasi tentang beban database (beban DB). Anda dapat “mengiris” beban DB berdasarkan dimensi seperti status tunggu atau kueri.

Topik

- Ikhtisar dasbor Wawasan Performa
- Membuka dasbor Wawasan Kinerja
- Menganalisis beban basis data dengan status tunggu
- Ikhtisar tab Kueri teratas
- Memperbesar bagan beban database

Ikhtisar dasbor Wawasan Performa

Dasbor adalah cara termudah untuk berinteraksi dengan Wawasan Performa. Contoh berikut menunjukkan dasbor untuk instance Amazon DocumentDB. Secara default, dasbor Wawasan Kinerja menunjukkan data selama satu jam terakhir.



Dasbor dibagi menjadi beberapa bagian berikut:

1. Metrik penghitung — Menampilkan data untuk metrik penghitung kinerja tertentu.
2. Beban basis data - Menunjukkan bagaimana beban DB dibandingkan dengan kapasitas instans DB seperti yang diwakili oleh baris vCPU Max.
3. Dimensi teratas - Menunjukkan dimensi teratas yang berkontribusi terhadap beban DB. Dimensi ini meliputi`waits`,`queries`,`hosts`,`databases`, dan`applications`.

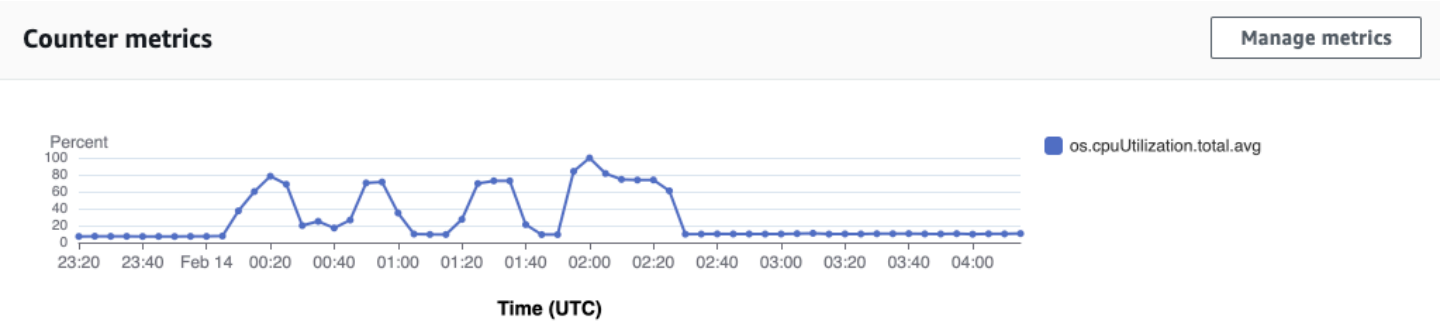
Topik

- Bagan metrik penghitung
- Bagan muatan basis data
- Tabel Dimensi teratas

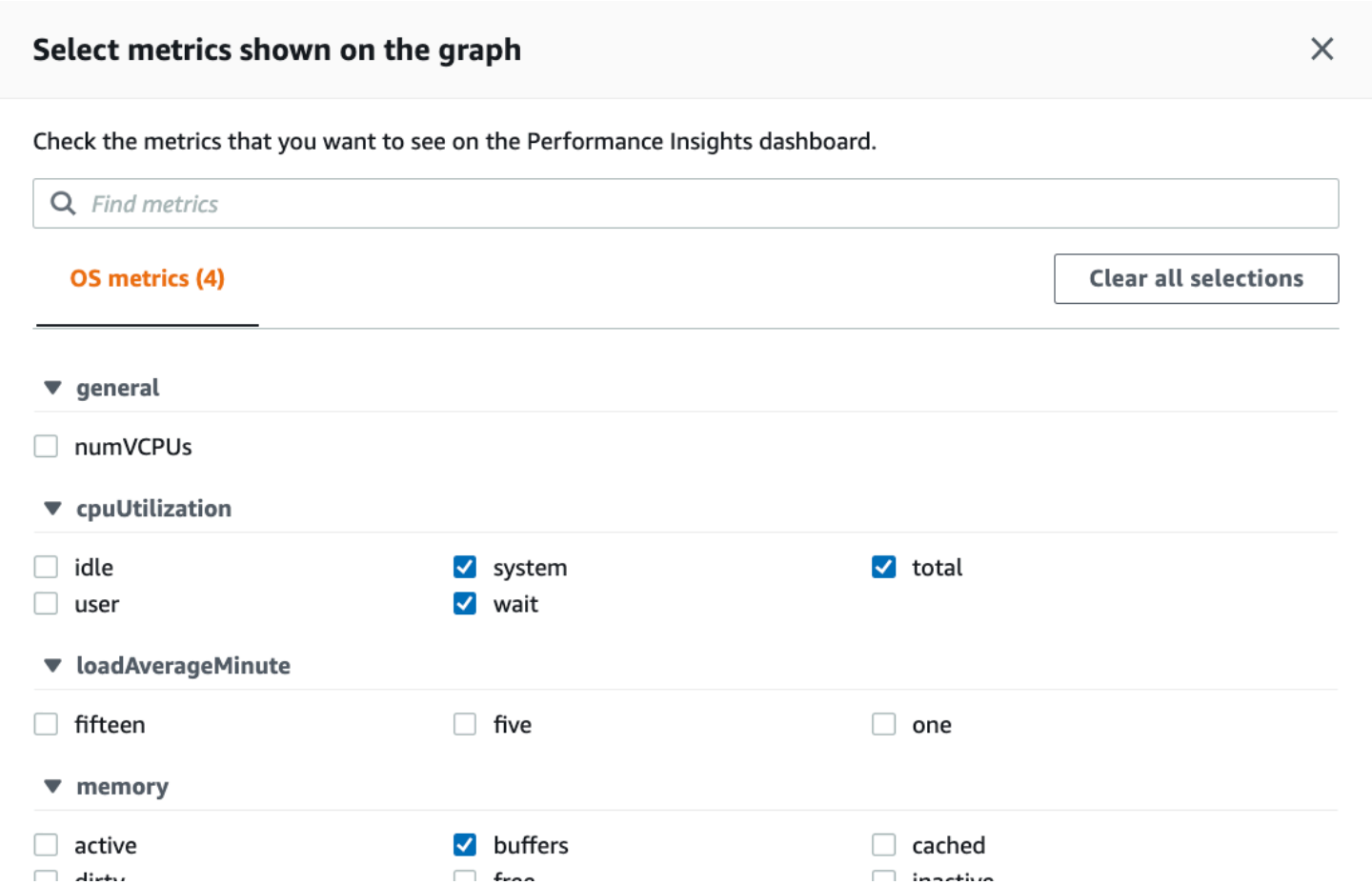
Bagan metrik penghitung

Dengan metrik penghitung, Anda dapat menyesuaikan dasbor Wawasan Performa untuk menyertakan hingga 10 grafik tambahan. Grafik ini menunjukkan pilihan lusinan metrik sistem operasi. Anda dapat menghubungkan informasi ini dengan muatan DB untuk membantu mengidentifikasi dan menganalisis masalah performa.

Bagan Metrik Penghitung menampilkan data untuk penghitung performa.



Untuk mengubah penghitung kinerja, pilih Kelola metrik. Anda dapat memilih beberapa metrik OS seperti yang ditunjukkan pada gambar berikut. Untuk melihat detail setiap metrik, arahkan kursor ke nama metrik.



Bagan muatan basis data

Bagan beban Database menunjukkan bagaimana aktivitas database dibandingkan dengan kapasitas instance seperti yang diwakili oleh baris Max vCPU. Secara default, bagan garis bertumpuk mewakili muatan DB sebagai sesi aktif rata-rata per unit waktu. Muatan DB diiris (dikelompokkan) berdasarkan status tunggu.



Muatan DB diiris berdasarkan dimensi

Anda dapat memilih untuk menampilkan muatan sebagai sesi aktif yang dikelompokkan berdasarkan dimensi yang didukung. Gambar berikut menunjukkan dimensi untuk instance Amazon DocumentDB.

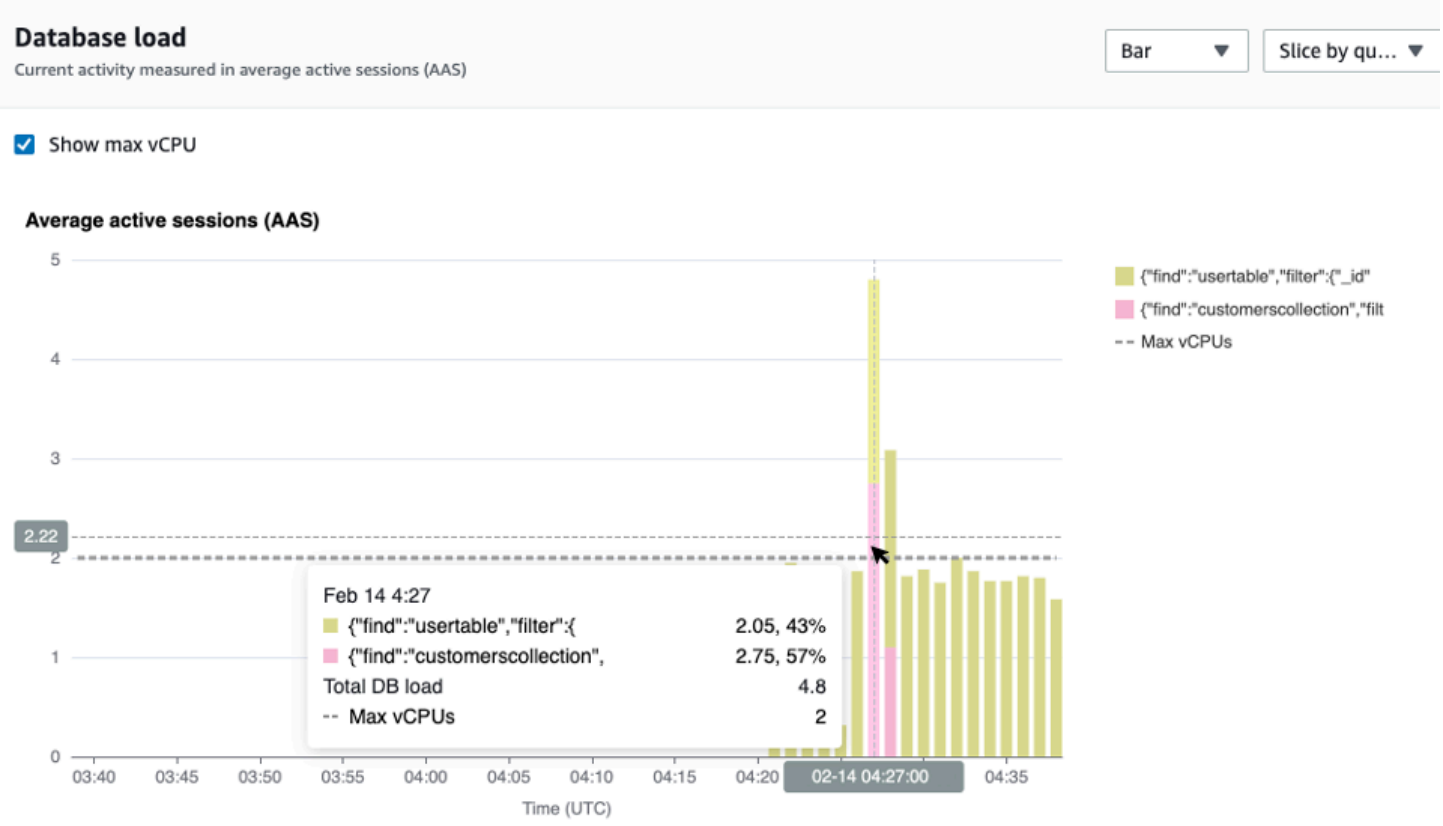


Detail muatan DB untuk item dimensi

Untuk melihat detail tentang item muatan DB dalam dimensi, arahkan kursor ke nama item. Gambar berikut menunjukkan rincian untuk pernyataan query.



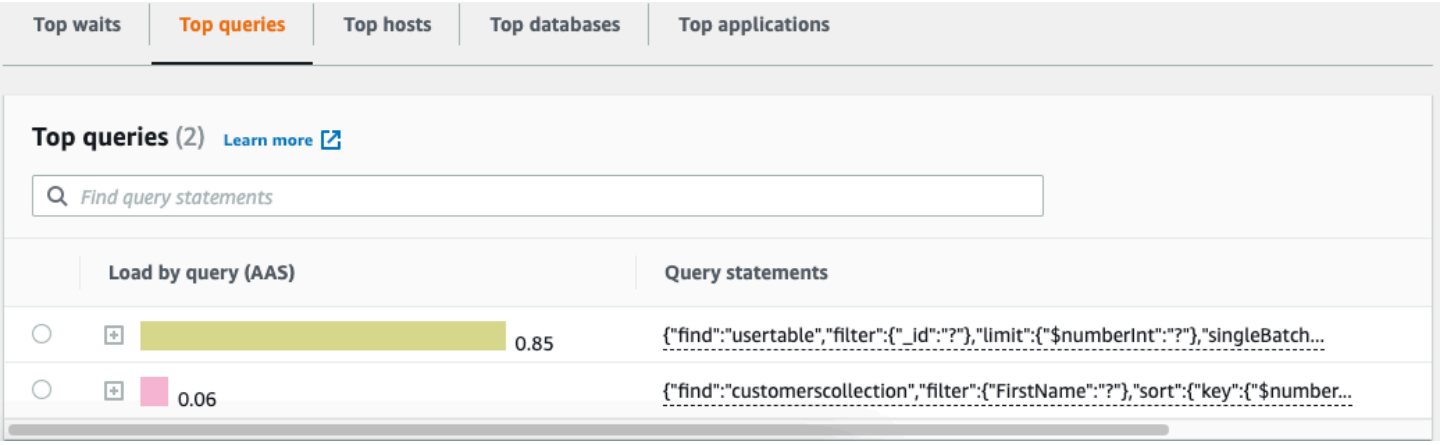
Untuk melihat detail setiap item selama periode waktu yang dipilih dalam legenda, arahkan kursor ke item tersebut.



Tabel Dimensi teratas

Tabel dimensi atas mengiris beban DB dengan dimensi yang berbeda. Dimensi adalah kategori atau “irisan menurut” untuk karakteristik muatan DB yang berbeda. Jika dimensinya adalah kueri, Kueri teratas menunjukkan pernyataan kueri yang paling berkontribusi pada beban DB.

Pilih salah satu tab dimensi berikut.



Tabel berikut memberikan deskripsi singkat dari setiap tab.

Deskripsi
Penggu na teratas mana backend basis data sedang menunggu
Pernyataa n teratas kueri yang sedang berjalan
Host teratas dan port klien yang terhubung
Basis data teratas yang terhubung ke klien
Aplikasi teratas yang

Tab	Deskripsi
	terhubung ke basis data

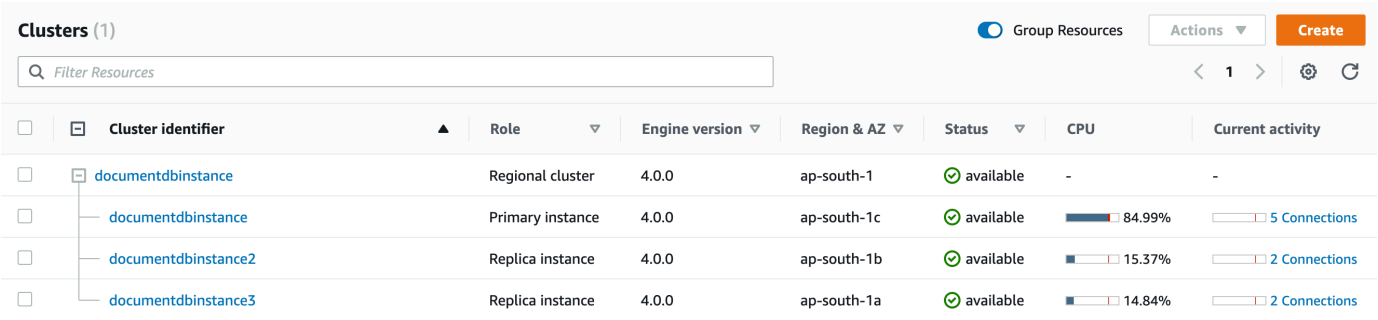
Untuk mempelajari cara menganalisis kueri menggunakan tab Kueri teratas, lihat. Ikhtisar tab Kueri teratas

Membuka dasbor Wawasan Kinerja

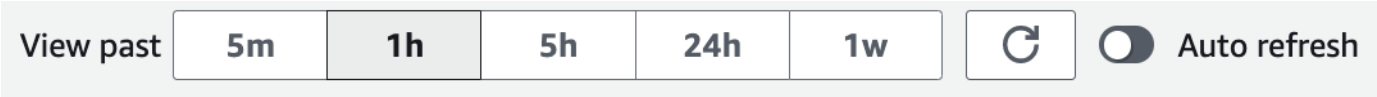
Untuk melihat dasbor Performance Insights di AWS Management Console, gunakan langkah-langkah berikut:

1. Buka konsol Performance Insights di. https://console.aws.amazon.com/docdb/
2. Pilih instans DB. Dasbor Performance Insights ditampilkan untuk instans Amazon DocumentDB tersebut.

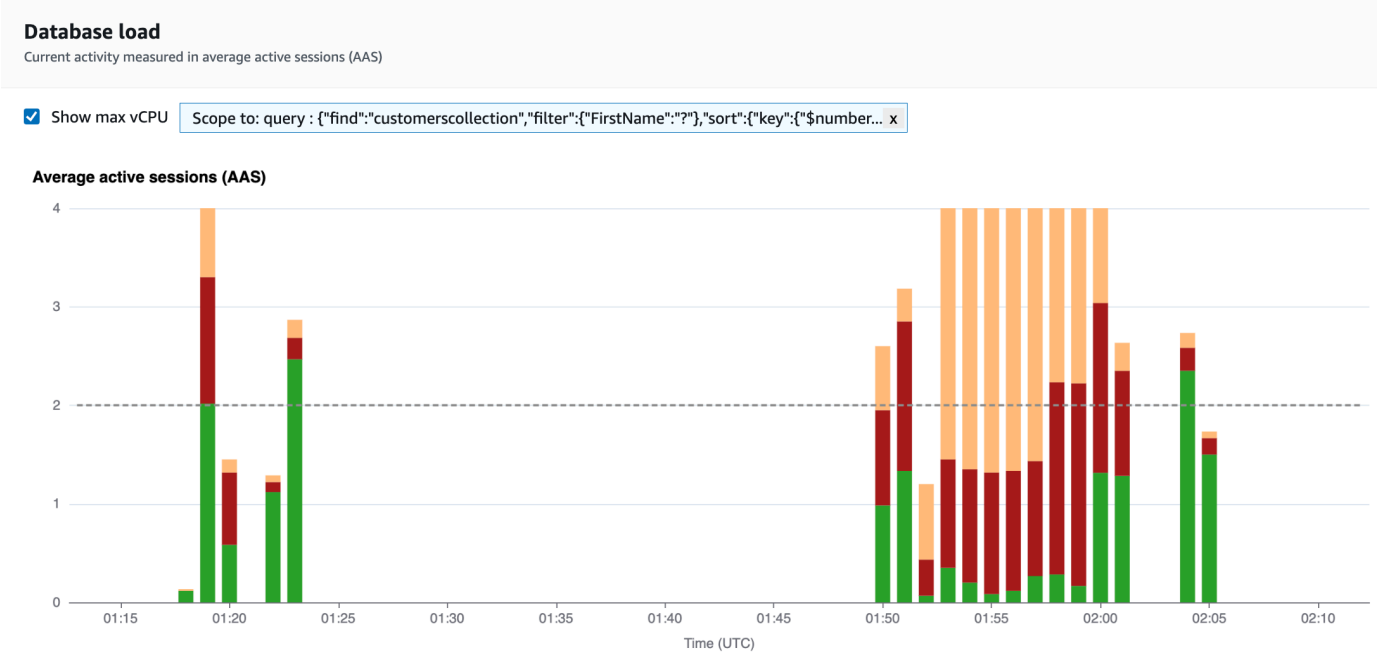
   Untuk instans Amazon DocumentDB dengan Performance Insights diaktifkan, Anda juga dapat mencapai dasbor dengan memilih item Sessions dalam daftar instance. Di bagian Aktivitas saat ini, item Sesi menunjukkan muatan basis data dalam sesi aktif rata-rata selama lima menit terakhir. Bilah ini secara grafis menunjukkan muatan. Ketika bar kosong, instance adalah idle. Saat muatan meningkat, bilah akan terisi dengan warna biru. Ketika beban melewati jumlah CPU virtual (vCPU) pada kelas instance, bilah berubah menjadi merah, menunjukkan potensi hambatan.



3. (Opsional) Pilih interval waktu yang berbeda dengan memilih tombol di kanan atas. Misalnya, untuk mengubah interval menjadi 1 jam, pilih 1 jam.



Pada tangkapan layar berikut, interval beban DB adalah 1 jam.



4. Untuk menyegarkan data Anda secara otomatis, aktifkan Penyegaran otomatis.



Dasbor Wawasan Kinerja secara otomatis menyegarkan kembali data baru. Kecepatan penyegaran tergantung pada jumlah data yang ditampilkan:

- 5 menit menyegarkan setiap 5 detik.
- 1 jam menyegarkan setiap menit.
- 5 jam menyegarkan setiap menit.
- 24 jam menyegarkan setiap 5 menit.
- 1 minggu menyegarkan setiap jam.

Menganalisis beban basis data dengan status tunggu

Jika bagan beban Database (beban DB) menunjukkan kemacetan, Anda dapat mengetahui dari mana beban itu berasal. Untuk melakukannya, lihat tabel item muatan teratas di bawah bagan

Muatan basis data. Pilih item tertentu, seperti kueri atau aplikasi, untuk menelusuri item itu dan melihat detailnya.

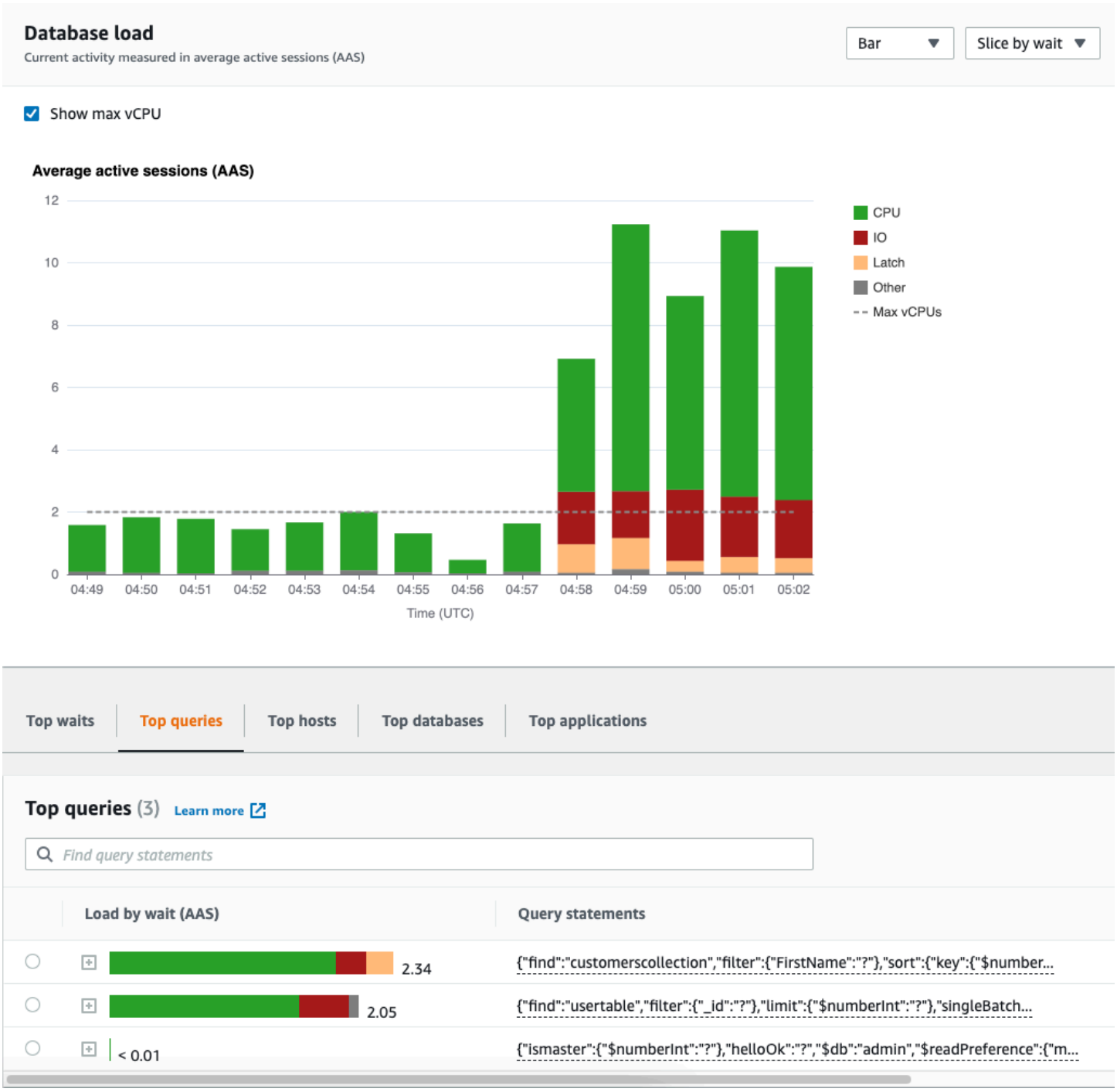
Beban DB yang dikelompokkan berdasarkan menunggu dan kueri teratas biasanya memberikan wawasan paling banyak tentang masalah kinerja. Muatan DB yang dikelompokkan berdasarkan peristiwa tunggu menunjukkan apakah ada sumber daya atau kemacetan konkurensi dalam basis data. Dalam hal ini, tab Kueri teratas pada tabel item pemuatan atas menunjukkan kueri mana yang mendorong pemuatan itu.

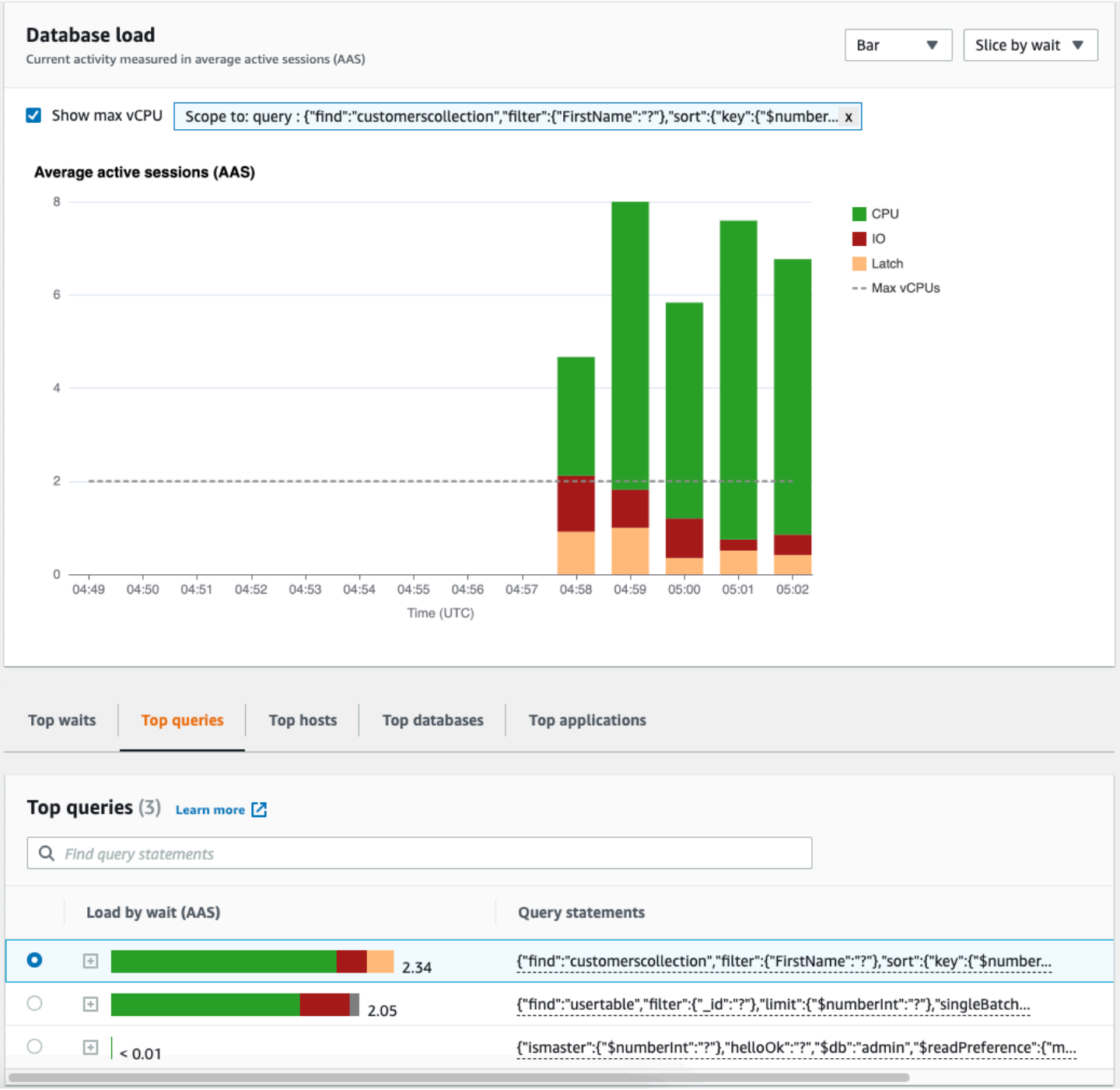
Alur kerja tipikal Anda untuk mendiagnosis masalah performa adalah sebagai berikut:

1. Tinjau bagan Muatan basis data dan lihat apakah ada insiden muatan basis data yang melebihi baris CPU Maks.
2. Jika ada, lihat bagan Muatan basis data dan identifikasi satu atau beberapa status tunggu yang paling bertanggung jawab.
3. Identifikasi kueri intisari yang menyebabkan pemuatan dengan melihat kueri mana yang tab Kueri teratas di tabel item pemuatan atas yang paling berkontribusi pada status tunggu tersebut. Anda dapat mengidentifikasi ini dengan kolom Load by Wait (AAS).
4. Pilih salah satu kueri intisari ini di tab Kueri teratas untuk memperluasnya dan melihat kueri anak yang terdiri darinya.

Anda juga dapat melihat host atau aplikasi mana yang berkontribusi paling banyak dengan memilih Host teratas atau Aplikasi teratas, masing-masing. Nama aplikasi ditentukan dalam string koneksi ke instance Amazon DocumentDB. `Unknown`menunjukkan bahwa bidang aplikasi tidak ditentukan.

Misalnya, di dasbor berikut, CPU menunggu sebagian besar beban DB. Memilih kueri teratas di bawah Kueri teratas akan mencakup bagan pemuatan Database untuk fokus pada beban terbanyak yang disumbangkan oleh kueri pilih.


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...



Ikhtisar tab Kueri teratas

Secara default, tab Kueri teratas menunjukkan kueri yang paling berkontribusi pada pemuatan DB. Anda dapat menganalisis teks kueri untuk membantu menyetel kueri Anda.

Topik

- Intisari kueri

- Muatan berdasarkan status tunggu (AAS)
- Melihat informasi kueri terperinci
- Mengakses teks kueri pernyataan
- Melihat dan mengunduh teks permintaan pernyataan

Intisari kueri

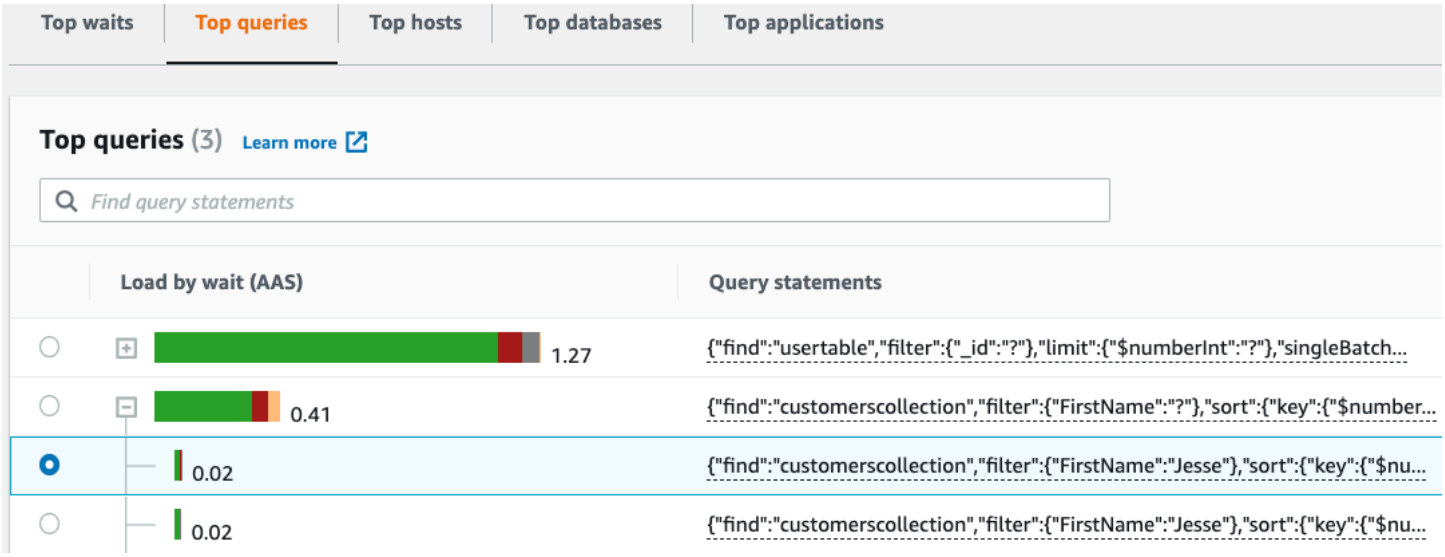
Intisari kueri adalah gabungan dari beberapa kueri aktual yang secara struktural serupa tetapi mungkin memiliki nilai literal yang berbeda. Digest menggantikan nilai berkode keras dengan tanda tanya. Misalnya, intisari kueri mungkin terlihat seperti ini:

```
{"find":"customerscollection","filter":{"FirstName":"?"},"sort":{"key":
{"$numberInt":"?"}},"limit":{"$numberInt":"?"}}
```

Digest ini dapat mencakup kueri turunan berikut:

```
{"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"}}
```

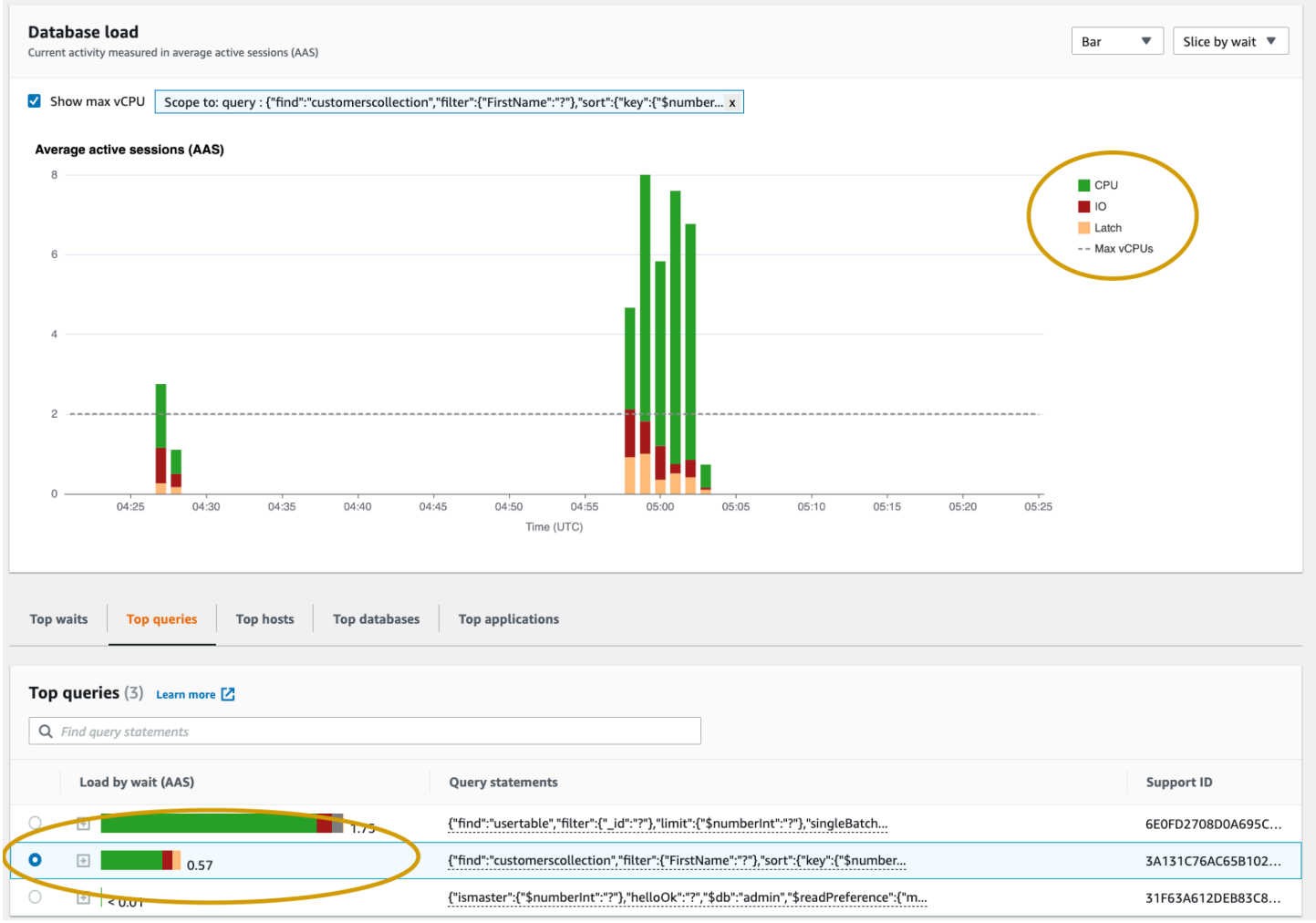
Untuk melihat pernyataan kueri literal dalam intisari, pilih kueri, lalu pilih simbol plus (+). Di tangkapan layar berikut, kueri yang dipilih adalah sebuah intisari.



> **Note**
>
> Intisari kueri mengelompokkan pernyataan kueri serupa, tetapi tidak menyunting informasi sensitif.

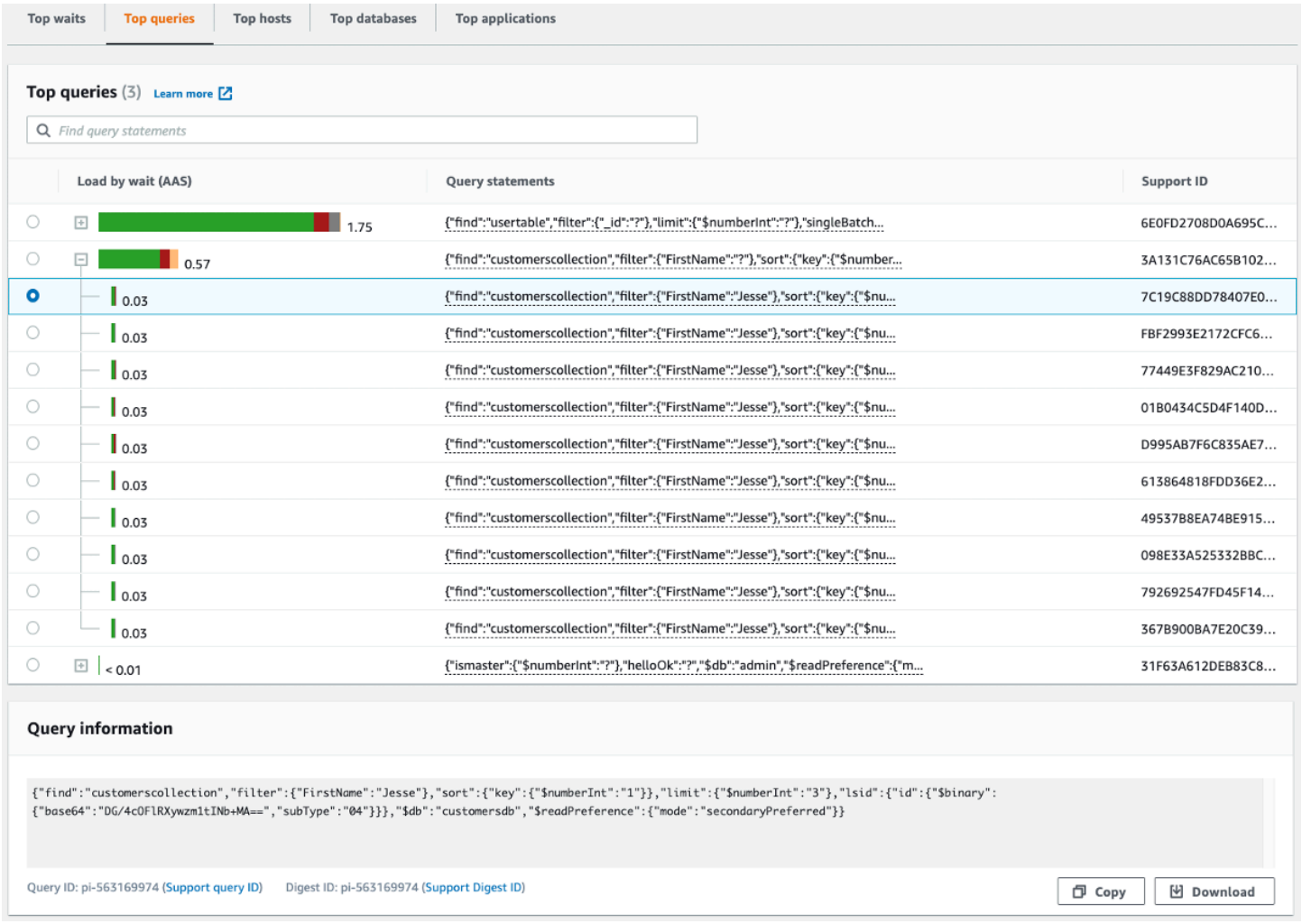
Muatan berdasarkan status tunggu (AAS)

Di Kueri teratas, kolom Load by waits (AAS) menggambarkan persentase beban database yang terkait dengan setiap item pemuatan teratas. Kolom ini mencerminkan beban untuk item tersebut dengan pengelompokan apa pun yang saat ini dipilih dalam bagan beban DB. Misalnya, Anda mungkin mengelompokkan bagan Muatan DB berdasarkan status tunggu. Dalam kasus ini, bilah Muatan DB berdasarkan Status Tunggu diberi ukuran, disegmentasi, dan diberi kode warna untuk menunjukkan seberapa banyak status tunggu tertentu yang dikontribusikan oleh kueri. Bilah ini juga menunjukkan status tunggu yang memengaruhi kueri yang dipilih.



Melihat informasi kueri terperinci

Dalam tabel kueri Top, Anda dapat membuka pernyataan intisari untuk melihat informasinya. Informasi ini muncul di panel bawah.



Jenis pengidentifikasi (ID) berikut dikaitkan dengan pernyataan kueri:

1. Support query ID - Nilai hash dari ID kueri. Nilai ini hanya untuk mereferensikan ID kueri saat Anda bekerja dengan AWS Support. AWS Support tidak memiliki akses ke ID kueri dan teks kueri Anda yang sebenarnya.
2. Support digest ID — Nilai hash dari ID intisari. Nilai ini hanya untuk mereferensikan ID intisari saat Anda bekerja dengan Support AWS . AWS Support tidak memiliki akses ke ID intisari dan teks kueri Anda yang sebenarnya.

Mengakses teks kueri pernyataan

Secara default, setiap baris dalam tabel Kueri Top menunjukkan 500 byte teks kueri untuk setiap pernyataan kueri. Jika pernyataan intisari melebihi 500 byte, Anda dapat melihat teks lainnya dengan membuka pernyataan di dasbor Performance Insights. Dalam hal ini, panjang maksimum untuk kueri yang ditampilkan adalah 1 KB. Jika Anda melihat pernyataan kueri lengkap, Anda juga dapat memilih Unduh.

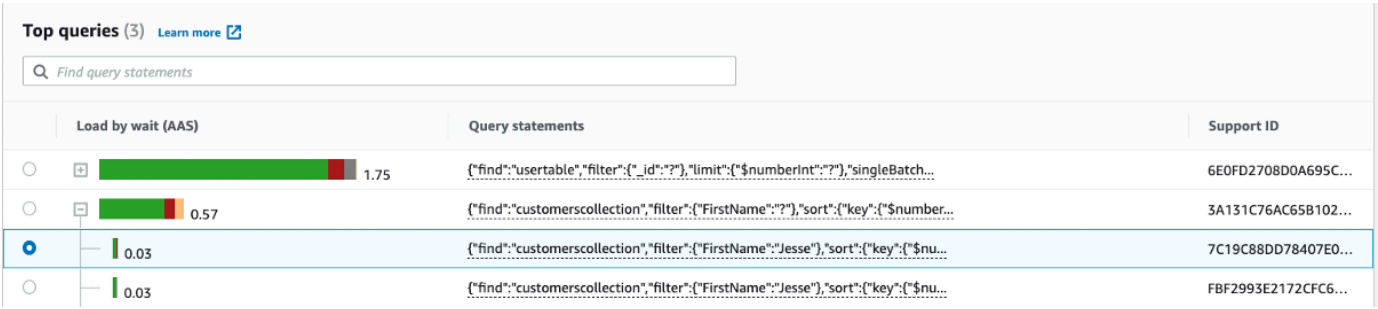
Melihat dan mengunduh teks permintaan pernyataan

Di dasbor Performance Insights, Anda dapat melihat atau mengunduh teks kueri.

Untuk melihat teks kueri lainnya di dasbor Performance Insights

1. Buka konsol Amazon DocumentDB di: https://console.aws.amazon.com/docdb/
2. Di panel navigasi, pilih Wawasan Performa.
3. Pilih instans DB. Dasbor Wawasan Kinerja ditampilkan untuk instans DB tersebut.

   Pernyataan kueri dengan teks yang lebih besar dari 500 byte akan terlihat seperti gambar berikut:



4. Periksa bagian informasi kueri untuk melihat lebih banyak teks kueri.



Dasbor Performance Insights dapat menampilkan hingga 1 KB untuk setiap pernyataan kueri lengkap.

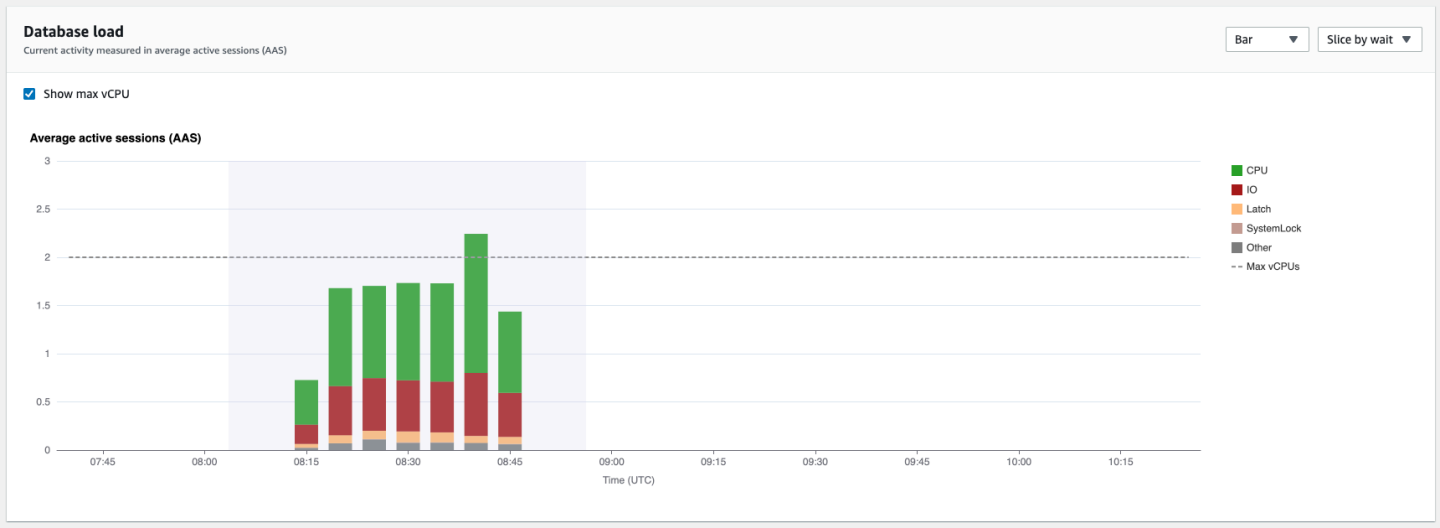
> **Note**
>
> Untuk menyalin atau mengunduh pernyataan kueri, nonaktifkan pemblokir pop-up apa pun.

Memperbesar bagan beban database

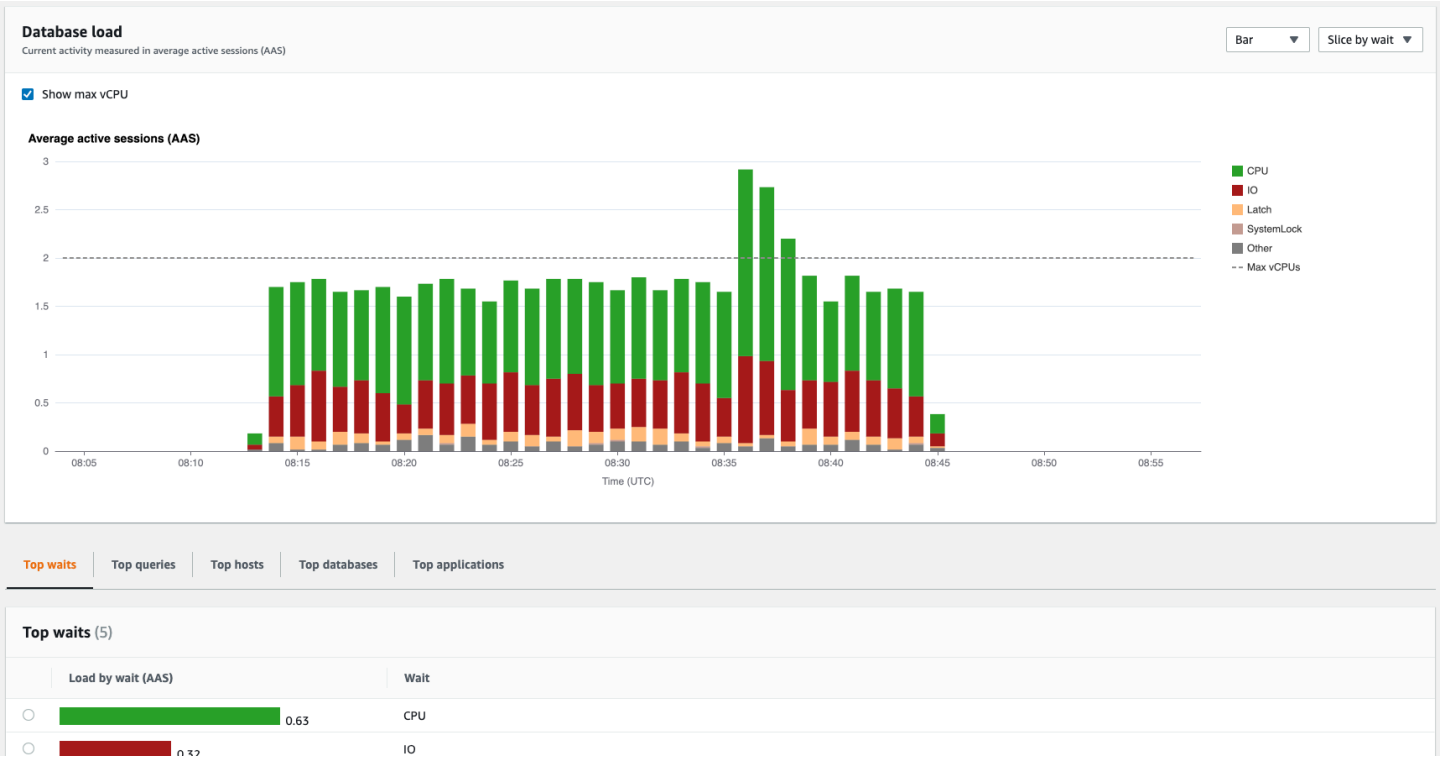
Anda dapat menggunakan fitur lain di antarmuka pengguna Wawasan Kinerja untuk membantu menganalisis data kinerja.

Click-and-Drag Memperbesar

Dalam antarmuka Wawasan Kinerja, Anda dapat memilih sebagian kecil bagan muatan dan memperbesar detailnya.



Untuk memperbesar sebagian bagan muatan, pilih waktu mulai dan seret ke akhir periode waktu yang Anda inginkan. Saat melakukannya, area yang dipilih akan disorot. Saat Anda melepaskan mouse, bagan beban memperbesar area yang dipilih, dan tabel Item teratas dihitung ulang.



Mengambil metrik dengan API Wawasan Performa

Saat Wawasan Kinerja diaktifkan, API menyediakan visibilitas ke dalam kinerja instans. Amazon CloudWatch Logs menyediakan sumber otoritatif untuk metrik pemantauan terjual untuk layanan. AWS

Wawasan Performa menawarkan tampilan spesifik domain dari muatan basis data yang diukur sebagai sesi aktif rata-rata (AAS). Metrik ini muncul sebagai set data deret waktu dua dimensi bagi konsumen API. Dimensi waktu data menyediakan data muatan DB untuk setiap titik waktu dalam rentang waktu yang dikueri. Setiap titik waktu menguraikan keseluruhan muatan dalam hubungannya dengan dimensi yang diminta, seperti `Query`, `Wait-state`, `Application`, atau `Host`, yang diukur pada titik waktu tersebut.

Amazon DocumentDB Performance Insights memantau instans Amazon DocumentDB DB Anda sehingga Anda dapat menganalisis dan memecahkan masalah kinerja database. Salah satu cara untuk menampilkan data Wawasan Performa dapat ditemukan di Konsol Manajemen AWS. Wawasan Performa juga menyediakan API publik sehingga Anda dapat mengueri data Anda sendiri. Anda dapat menggunakan API untuk melakukan tindakan berikut:

- Membongkar data ke dalam basis data
- Menambahkan data Wawasan Performa ke dasbor pemantauan yang ada

- Merancang alat pemantauan

Untuk menggunakan API Performance Insights, aktifkan Performance Insights di salah satu instans Amazon DocumentDB Anda. Untuk informasi tentang cara mengaktifkan Wawasan Performa, lihat Mengaktifkan dan menonaktifkan Wawasan Kinerja. Untuk informasi selengkapnya tentang API Performance Insights, lihat Referensi API Performance Insights.

API Wawasan Performa menyediakan operasi berikut.

Tindakan Wawasan Performa	AWS CLI perintah	Deskripsi
DescribeDimensionKeys	aws pi describe-dimension-keys	Mengambil kunci dimensi N teratas untuk metrik dalam jangka waktu tertentu.
GetDimensionKeyDetails	aws pi get-dimension-key-details	Mengambil atribut dari grup dimensi tertentu untuk instans DB atau sumber data. Misalnya, jika Anda menentukan ID kueri, dan jika detail dimensi tersedia, `GetDimensionKeyDetails` mengambil teks lengkap dari dimensi yang `db.query.statement` terkait dengan ID ini. Operasi ini berguna karena `GetResourceMetrics` dan `DescribeDimensionKeys` tidak mendukung pengambilan teks pernyataan kueri besar.
GetResourceMetadata	aws pi get-resource-metadata	Mengambil metadata untuk fitur yang berbeda. Misalnya, metadata mungkin menunjukkan bahwa fitur diaktifkan atau

Tindakan Wawasan Performa	AWS CLI perintah	Deskripsi
		dinonaktifkan pada instans DB tertentu.
GetResourceMetrics	aws pi get-resource-metrics	Mengambil metrik Wawasan Performa untuk sekumpulan sumber data, selama periode waktu tertentu. Anda dapat menyediakan grup dimensi dan dimensi tertentu, serta memberikan kriteria penggabungan dan pemfilteran untuk setiap grup.
ListAvailableResourceDimensions	aws pi list-available-resource-dimensions	Mengambil dimensi yang dapat dikueri untuk setiap jenis metrik yang ditentukan pada instans tertentu.
ListAvailableResourceMetrics	aws pi list-available-resource-metrics	Mengambil semua metrik jenis metrik tertentu yang tersedia yang dapat dikueri untuk instans DB tertentu.

Topik

- AWS CLI untuk Performance Insights
- Mengambil metrik deret waktu
- AWS CLI contoh untuk Performance Insights

AWS CLI untuk Performance Insights

Anda dapat melihat data Wawasan Performa menggunakan AWS CLI. Anda dapat menampilkan bantuan untuk perintah AWS CLI untuk Wawasan Performa dengan memasukkan berikut ini di baris perintah.

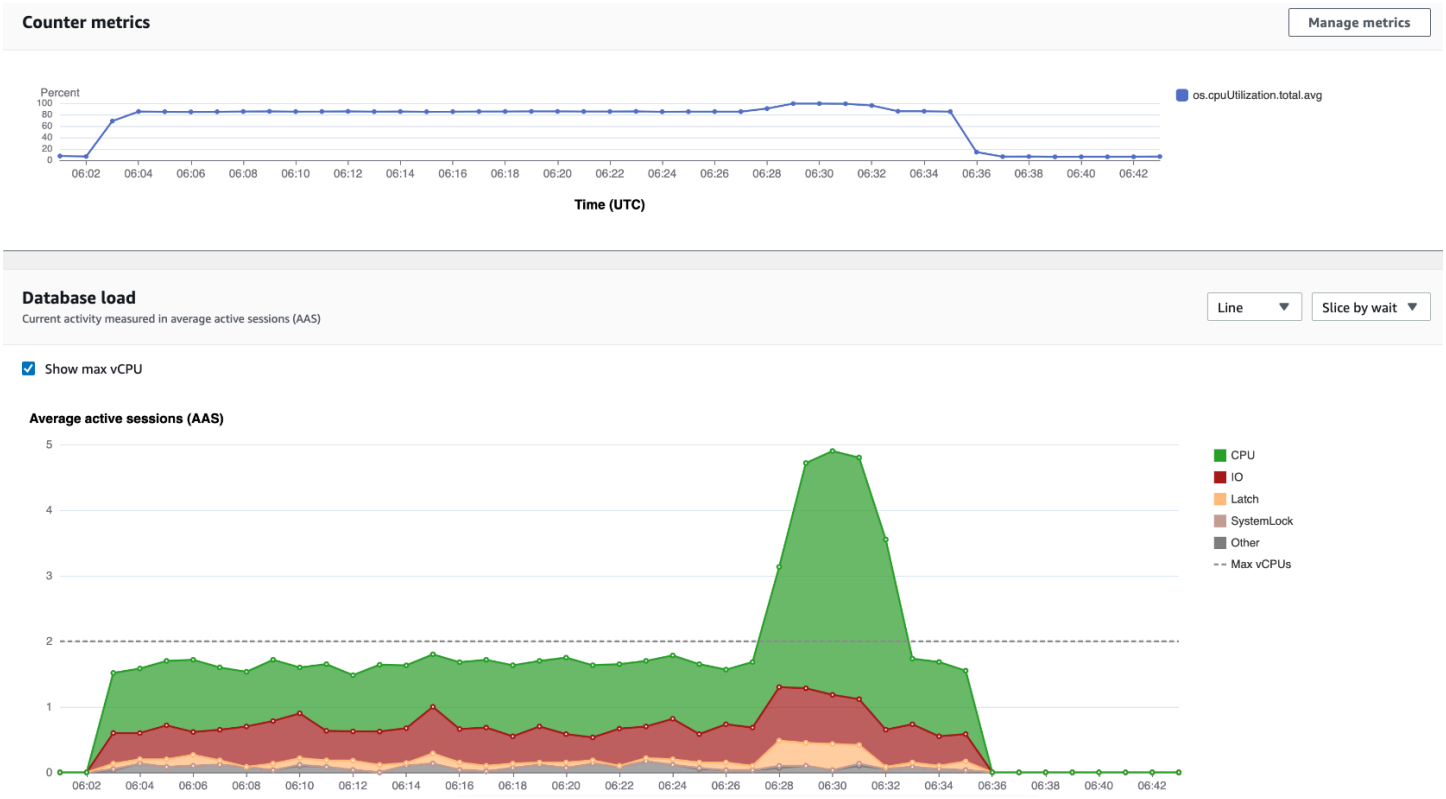
```
aws pi help
```

Jika Anda belum menginstal, lihat AWS CLI Menginstal Antarmuka Baris AWS Perintah di Panduan AWS CLI Pengguna untuk informasi tentang menginstalnya.

Mengambil metrik deret waktu

Operasi `GetResourceMetrics` mengambil satu atau beberapa metrik deret waktu dari data Wawasan Performa. `GetResourceMetrics` memerlukan metrik dan periode waktu, dan menampilkan respons dengan daftar poin data.

Misalnya, Konsol Manajemen AWS penggunaan `GetResourceMetrics` untuk mengisi bagan Metrik Penghitung dan bagan Pemuatan Database, seperti yang terlihat pada gambar berikut.



Semua metrik yang ditampilkan oleh `GetResourceMetrics` adalah metrik deret waktu standar, dengan pengecualian `db.load`. Metrik ini ditampilkan dalam bagan Muatan Basis Data. Metrik `db.load` berbeda dengan metrik deret waktu lainnya karena Anda dapat membaginya menjadi beberapa sub-komponen yang disebut dimensi. Di gambar sebelumnya, `db.load` dibagi dan dikelompokkan berdasarkan status tunggu yang membentuk `db.load`.

> **Note**
>
> `GetResourceMetrics` juga dapat menampilkan metrik `db.sampleload`, tetapi metrik `db.load` sesuai di sebagian besar kasus.

Untuk informasi tentang metrik penghitung yang ditampilkan oleh `GetResourceMetrics`, lihat Performance Insights untuk metrik penghitung.

Penghitungan berikut didukung untuk metrik:

- Rata-rata – Nilai rata-rata untuk metrik selama periode waktu tertentu. Tambahkan `.avg` ke nama metrik.
- Minimum – Nilai minimum metrik selama periode waktu tertentu. Tambahkan `.min` ke nama metrik.
- Maksimum – Nilai maksimum metrik selama periode waktu tertentu. Tambahkan `.max` ke nama metrik.
- Jumlah – Jumlah nilai metrik selama periode waktu tertentu. Tambahkan `.sum` ke nama metrik.
- Jumlah sampel – Frekuensi pengumpulan metrik selama periode waktu tertentu. Tambahkan `.sample_count` ke nama metrik.

Sebagai contoh, misalkan sebuah metrik dikumpulkan selama 300 detik (5 menit), dan metrik tersebut dikumpulkan satu kali setiap menit. Nilai untuk setiap menit adalah 1, 2, 3, 4, dan 5. Dalam kasus ini, penghitungan berikut ditampilkan:

- Rata-rata – 3
- Minimum – 1
- Maksimum – 5
- Jumlah – 15
- Jumlah sampel – 5

Untuk informasi tentang menggunakan `get-resource-metrics` AWS CLI perintah, lihat `get-resource-metrics`.

Untuk opsi `--metric-queries`, tentukan satu atau beberapa kueri yang diinginkan untuk mendapatkan hasil. Setiap kueri terdiri dari `Metric` wajib dan `GroupBy` opsional serta parameter `Filter`. Berikut ini adalah contoh spesifikasi opsi `--metric-queries`.

```
{
   "Metric": "string",
   "GroupBy": {
     "Group": "string",
     "Dimensions": ["string", ...],
     "Limit": integer
   },
   "Filter": {"string": "string"
     ...}
```

AWS CLI contoh untuk Performance Insights

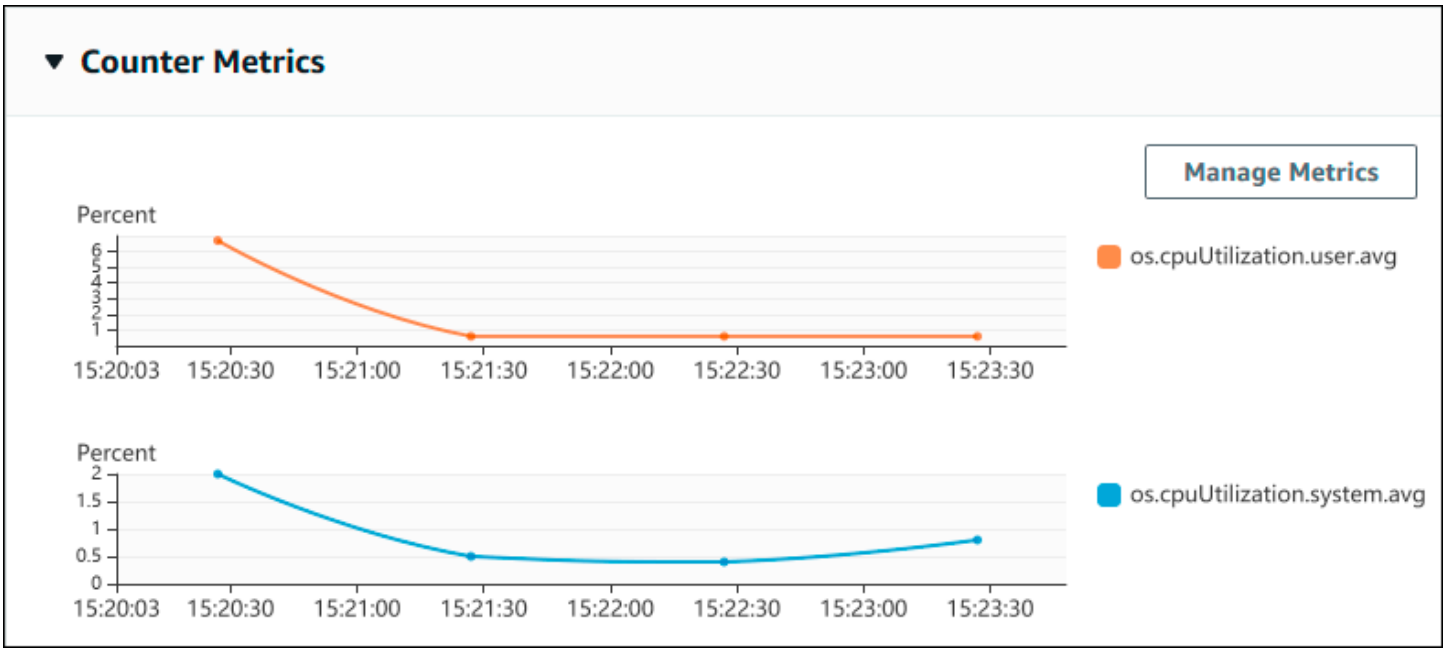
Contoh berikut menunjukkan cara menggunakan Performance AWS CLI Insights.

Topik

- Mengambil metrik penghitung
- Mengambil rata-rata beban DB untuk status tunggu teratas
- Mengambil rata-rata beban DB untuk Query teratas
- Mengambil rata-rata beban DB yang disaring oleh Query

Mengambil metrik penghitung

Tangkapan layar berikut menunjukkan dua bagan metrik penghitung dalam Konsol Manajemen AWS.



Contoh berikut menunjukkan cara mengumpulkan data yang sama yang digunakan oleh Konsol Manajemen AWS untuk menghasilkan dua bagan metrik penghitung.

Untuk Linux, macOS, atau 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"}]'
```

Untuk 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"}]'
```

Anda juga dapat membuat perintah menjadi lebih mudah dibaca dengan menentukan file untuk opsi `--metrics-query`. Contoh berikut menggunakan file yang disebut query.json untuk opsi tersebut. File memiliki konten berikut.

```
[
    {
        "Metric": "os.cpuUtilization.user.avg"
    },
    {
        "Metric": "os.cpuUtilization.idle.avg"
    }
]
```

Jalankan perintah berikut untuk menggunakan file.

Untuk Linux, macOS, atau 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
```

Untuk 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
```

Contoh sebelumnya menentukan nilai-nilai berikut untuk opsi:

- `--service-type`- `DOCDB` untuk Amazon DocumentDB
- `--identifier` – ID sumber daya untuk instans DB
- `--start-time` dan `--end-time` – Nilai `DateTime` ISO 8601 untuk periode kueri, dengan berbagai format yang didukung

Ini akan dikueri selama rentang waktu satu jam:

- `--period-in-seconds` – `60` untuk kueri per menit
- `--metric-queries` – Rangkaian dua kueri, masing-masing hanya untuk satu metrik.

  Nama metrik menggunakan titik untuk mengklasifikasikan metrik dalam kategori yang berguna, dengan elemen terakhir sebagai fungsi. Dalam contoh, fungsinya adalah `avg` untuk setiap kueri. Seperti halnya Amazon CloudWatch, fungsi yang didukung adalah `min`,`max`,`total`, dan`avg`.

Responsnya terlihat seperti berikut.

```
{
    "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
```

Respons ini memiliki `Identifier`, `AlignedStartTime`, dan `AlignedEndTime`. B nilai `--period-in-seconds` adalah `60`, waktu mulai dan akhir telah disesuaikan dengan menit. Jika `--period-in-seconds` adalah `3600`, waktu mulai dan akhir akan disesuaikan dengan jam.

`MetricList` dalam respons memiliki sejumlah entri, masing-masing dengan entri `Key` dan `DataPoints`. Masing-masing `DataPoint` memiliki `Timestamp` dan `Value`. Masing-masing daftar `Datapoints` memiliki 60 poin data karena kueri tersebut adalah untuk data per menit selama

satu jam, dengan `Timestamp1/Minute1`, `Timestamp2/Minute2`, dan seterusnya, hingga `Timestamp60/Minute60`.

Karena kueri tersebut adalah untuk dua metrik penghitung yang berbeda, ada dua elemen dalam respons `MetricList`.

Mengambil rata-rata beban DB untuk status tunggu teratas

Contoh berikut adalah query yang sama yang Konsol Manajemen AWS digunakan untuk menghasilkan grafik garis area bertumpuk. Contoh ini mengambil `db.load.avg` untuk jam terakhir dengan beban dibagi menurut tujuh status tunggu teratas. Perintah ini sama dengan perintah dalam Mengambil metrik penghitung. Namun, file query.json berisi konten berikut.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.wait_state", "Limit": 7 }
    }
]
```

Jalankan perintah berikut.

Untuk Linux, macOS, atau 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
```

Untuk 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
```

Contoh menentukan metrik `db.load.avg` dan a `GroupBy` dari tujuh status tunggu teratas. Untuk detail tentang nilai yang valid untuk contoh ini, lihat DimensionGroup dalam Referensi API Wawasan Kinerja.

Responsnya terlihat seperti berikut.

```
{
    "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
```

Dalam tanggapan ini, ada tiga entri dalam. `MetricList` Ada satu entri untuk total`db.load.avg`, dan tiga entri masing-masing untuk `db.load.avg` dibagi menurut salah satu dari tiga negara tunggu teratas. Karena ada dimensi pengelompokan (tidak seperti contoh pertama), harus ada satu kunci untuk setiap pengelompokan metrik. Tidak boleh hanya ada satu kunci untuk setiap metrik, seperti dalam kasus penggunaan metrik penghitung dasar.

Mengambil rata-rata beban DB untuk Query teratas

Contoh berikut dikelompokkan `db.wait_state` oleh 10 pernyataan kueri teratas. Ada dua kelompok berbeda untuk pernyataan kueri:

- `db.query`— Pernyataan kueri lengkap, seperti `{"find":"customers","filter": {"FirstName":"Jesse"},"sort":{"key":{"$numberInt":"1"}}}`
- `db.query_tokenized`— Pernyataan kueri tokenized, seperti `{"find":"customers","filter":{"FirstName":"?"},"sort":{"key": {"$numberInt":"?"}},"limit":{"$numberInt":"?"}}`

Saat menganalisis kinerja database, akan berguna untuk mempertimbangkan pernyataan kueri yang hanya berbeda dengan parameternya sebagai satu item logika. Jadi, Anda dapat menggunakan `db.query_tokenized` saat melakukan kueri. Namun, terutama ketika Anda tertarik`explain()`, terkadang lebih berguna untuk memeriksa pernyataan kueri lengkap dengan parameter. Ada hubungan orang tua-anak antara kueri tokenized dan full, dengan beberapa kueri lengkap (turunan) dikelompokkan di bawah kueri tokenized yang sama (induk).

Perintah dalam contoh ini terlihat seperti perintah dalam Mengambil rata-rata beban DB untuk status tunggu teratas. Namun, file query.json berisi konten berikut.

```
[
```

```
    {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.query_tokenized", "Limit": 10 }
    }
]
```

Contoh berikut menggunakan `db.query_tokenized`.

Untuk Linux, macOS, atau 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
```

Untuk 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
```

Contoh ini menanyakan lebih dari 1 jam, dengan periode satu menit dalam detik.

Contoh menentukan metrik `db.load.avg` dan a `GroupBy` dari tujuh status tunggu teratas. Untuk detail tentang nilai yang valid untuk contoh ini, lihat DimensionGroup dalam Referensi API Wawasan Kinerja.

Responsnya terlihat seperti berikut.

```
{
    "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
```

Respons ini memiliki 11 entri dalam `MetricList` (total 1, 10 kueri tokenized teratas), dengan setiap entri memiliki 24 per jam. `DataPoints`

Untuk kueri tokenized, ada tiga entri di setiap daftar dimensi:

- `db.query_tokenized.statement`— Pernyataan kueri tokenized.
- `db.query_tokenized.db_id` — ID sintetis yang dihasilkan Performance Insights untuk Anda. Contoh ini menampilkan ID sintetis `pi-1064184600`.
- `db.query_tokenized.id` – ID kueri di dalam Wawasan Performa.

  Dalam Konsol Manajemen AWS, ID ini disebut Support ID. Ini dinamakan ini karena ID adalah data yang AWS Support dapat memeriksa untuk membantu Anda memecahkan masalah dengan database Anda. AWS mengambil keamanan dan privasi data Anda dengan sangat serius, dan hampir semua data disimpan terenkripsi dengan Anda. AWS KMS key Oleh karena itu, tidak ada orang di dalam yang AWS dapat melihat data ini. Di contoh sebelumnya, baik `tokenized.statement` maupun `tokenized.db_id` disimpan dengan enkripsi. Jika Anda

memiliki masalah dengan database Anda, AWS Support dapat membantu Anda dengan mereferensikan ID Dukungan.

Ketika melakukan kueri, mungkin lebih mudah untuk menentukan `Group` dalam `GroupBy`. Namun, untuk kontrol lebih mendetail atas data yang ditampilkan, tentukan daftar dimensi. Misalnya, jika yang dibutuhkan hanya `db.query_tokenized.statement`, atribut `Dimensions` dapat ditambahkan ke file query.json.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": {
            "Group": "db.query_tokenized",
            "Dimensions":["db.query_tokenized.statement"],
            "Limit": 10
        }
    }
]
```

Mengambil rata-rata beban DB yang disaring oleh Query

Kueri API terkait dalam contoh ini sama seperti perintah di Mengambil rata-rata beban DB untuk Query teratas. Namun, file query.json berisi konten berikut.

```
[
 {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.wait_state", "Limit": 5  },
        "Filter": { "db.query_tokenized.id": "AKIAIOSFODNN7EXAMPLE" }
    }
]
```

Dalam tanggapan ini, semua nilai difilter sesuai dengan kontribusi kueri token AKIAIOSFODNN7EXAMPLE yang ditentukan dalam file query.json. Kunci juga mungkin mengikuti urutan yang berbeda dari kueri tanpa filter, karena ini adalah lima status tunggu teratas yang memengaruhi kueri yang difilter.

CloudWatch Metrik Amazon untuk Performance Insights

Performance Insights secara otomatis menerbitkan metrik ke Amazon. CloudWatch Data yang sama dapat ditanyakan dari Performance Insights, tetapi memiliki metrik CloudWatch memudahkan untuk menambahkan alarm. CloudWatch Ini juga memudahkan untuk menambahkan metrik ke Dasbor CloudWatch yang ada.

Metrik	Deskripsi
DBLoad	Jumlah sesi aktif untuk Amazon DocumentDB. Biasanya, Anda menginginkan data untuk jumlah rata-rata sesi aktif. Dalam Wawasan Performa, data ini dikueri sebagai `db.load.avg` .
DBLoadCPU	Jumlah sesi aktif di mana tipe status tunggu adalah CPU. Di Performance Insights, data ini ditanyakan sebagai`db.load.avg` , difilter berdasarkan jenis status tunggu. `CPU`
DBLoadNonCPU	Jumlah sesi aktif di mana tipe status tunggu bukan CPU.

> **Note**
>
> Metrik ini dipublikasikan CloudWatch hanya jika ada beban pada instans DB.

Anda dapat memeriksa metrik ini menggunakan CloudWatch konsol, the AWS CLI, atau CloudWatch API.

Misalnya, Anda bisa mendapatkan statistik untuk metrik `DBLoad` dengan menjalankan perintah 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
```

Contoh ini menghasilkan output yang terlihat seperti berikut.

```
{
    "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"
}
```

Anda dapat menggunakan fungsi matematika `DB_PERF_INSIGHTS` metrik di CloudWatch konsol untuk menanyakan metrik penghitung Performance Insights Amazon DocumentDB. `DB_PERF_INSIGHTS`Fungsi ini juga mencakup `DBLoad` metrik pada interval sub-menit. Anda dapat mengatur CloudWatch alarm pada metrik ini. Untuk detail selengkapnya tentang cara membuat alarm, lihat Membuat alarm di metrik penghitung Performance Insights dari database. AWS

Untuk informasi selengkapnya CloudWatch, lihat Apa itu Amazon CloudWatch? di Panduan CloudWatch Pengguna Amazon.

Performance Insights untuk metrik penghitung

Metrik penghitung adalah metrik sistem operasi di dasbor Performance Insights. Untuk membantu mengidentifikasi dan menganalisis masalah performa, Anda dapat mengaitkan metrik penghitung dengan muatan DB.

Penghitung sistem operasi Wawasan Performa

Penghitung sistem operasi berikut tersedia dengan Amazon DocumentDB Performance Insights.

Penghitung	Jenis	Metrik
aktif	memori	os.memory.active
buffer	memori	os.memory.buffers
dalam cache	memori	os.memory.cached
kotor	memori	os.memory.dirty
kosong	memori	os.memory.free
tidak aktif	memori	os.memory.inactive
dipetakan	memori	os.memory.mapped
pageTables	memori	os.memory.pageTables
lempengan	memori	os.memory.slab

Penghitung	Jenis	Metrik
total	memori	os.memory.total
tulis balik	memori	os.memory.writeback
diam	cpuUtilization	os.cpu Utilization.idle
sistem	cpuUtilization	os.cpu Utilization.system
total	cpuUtilization	os.cpu Utilization.total
user	cpuUtilization	os.cpu Utilization.user
tunggu	cpuUtilization	os.cpu Utilization.wait
satu	beban AverageMinute	os.beban AverageMinute.one
lima belas	beban AverageMinute	os.beban AverageMinute.fift een
lima	beban AverageMinute	os.beban AverageMinute.five
dalam cache	ganti	os.swap.cached
kosong	swap	os.swap.free
masuk	swap	os.swap.in
keluar	swap	os.swap.out
total	swap	os.swap.total
rx	jaringan	os.network.rx
tx	jaringan	os.network.tx
numVCPUs	umum	os.general.numVCPUs

Zero-ETL Integrasi dengan Amazon OpenSearch Service

Topik

- OpenSearch Layanan Amazon sebagai tujuan
- Batasan

OpenSearch Layanan Amazon sebagai tujuan

OpenSearch Integrasi layanan dengan Amazon DocumentDB memungkinkan Anda melakukan streaming beban penuh dan mengubah peristiwa data ke domain. OpenSearch Infrastruktur konsumsi di-host sebagai saluran pipa OpenSearch konsumsi dan menyediakan mekanisme latensi rendah skala tinggi untuk terus mengalirkan data dari koleksi Amazon DocumentDB.

Selama pemuatan penuh, integrasi nol-ETL pertama-tama mengekstrak data beban penuh historis untuk OpenSearch menggunakan pipa konsumsi. Setelah data beban penuh tertelan, saluran pipa OpenSearch konsumsi akan mulai membaca data dari aliran perubahan Amazon DocumentDB dan akhirnya mengejar ketinggalan untuk mempertahankan konsistensi data mendekati waktu nyata antara Amazon DocumentDB dan. OpenSearch OpenSearch menyimpan dokumen dalam indeks. Data yang masuk dari koleksi Amazon DocumentDB dapat dikirim ke salah satu indeks atau dapat dipartisi ke dalam indeks yang berbeda. Saluran pipa konsumsi akan menyinkronkan semua acara pembuatan, pembaruan, dan penghapusan dalam koleksi Amazon DocumentDB sebagai pembuatan, pembaruan, dan penghapusan OpenSearch dokumen yang sesuai untuk menjaga kedua sistem data tetap sinkron. Saluran pipa konsumsi dapat dikonfigurasi untuk membaca data dari satu koleksi dan menulis ke satu indeks atau membaca data dari satu koleksi dan merutekan secara kondisional ke beberapa indeks.

Saluran pipa konsumsi dapat dikonfigurasi untuk mengalirkan data dari Amazon DocumentDB ke Amazon Service menggunakan: OpenSearch

- Beban penuh saja
- Streaming perubahan peristiwa aliran dari Amazon DocumentDB tanpa beban penuh
- Beban penuh diikuti oleh aliran perubahan dari Amazon DocumentDB

Untuk mengatur saluran konsumsi Anda, lakukan langkah-langkah berikut:

Langkah 1: Buat domain OpenSearch Layanan Amazon atau koleksi OpenSearch tanpa server

Diperlukan koleksi OpenSearch Layanan Amazon dengan izin yang sesuai untuk membaca data. Lihat Memulai OpenSearch Layanan Amazon atau Memulai Amazon OpenSearch Tanpa Server di Panduan Pengembang OpenSearch Layanan Amazon untuk membuat koleksi. Lihat Amazon OpenSearch Ingestion di Panduan Pengembang OpenSearch Layanan Amazon untuk membuat peran AIM dengan izin yang benar untuk mengakses data tulis ke koleksi atau domain.

Langkah 2: Aktifkan aliran perubahan di cluster Amazon DocumentDB

Pastikan aliran perubahan diaktifkan pada koleksi yang diperlukan di klaster Amazon DocumentDB. Lihat Menggunakan aliran perubahan dengan Amazon DocumentDB untuk informasi lebih lanjut.

Langkah 3: Siapkan peran pipeline dengan izin untuk menulis ke bucket Amazon S3 dan domain atau koleksi tujuan

Setelah koleksi Amazon DocumentDB dibuat dan mengubah aliran diaktifkan, siapkan peran pipeline yang ingin digunakan dalam konfigurasi pipeline, dan tambahkan izin berikut dalam peran:

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/*"
            ]
        }
    ]
```

```
}
```

Agar OpenSearch pipeline dapat menulis data ke OpenSearch domain, domain harus memiliki kebijakan akses tingkat domain yang memungkinkan peran pipeline sts_role_arn untuk mengaksesnya. Contoh kebijakan akses domain berikut memungkinkan peran pipeline bernama`pipeline-role`, yang Anda buat pada langkah sebelumnya, untuk menulis data ke domain bernama`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}/*"
    }
  ]
}
```

Langkah 4: Tambahkan izin yang diperlukan pada peran pipeline untuk membuat 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" }
            }
        }
    ]
}
```

Langkah 5: Buat pipa

Konfigurasikan saluran OpenSearch konsumsi yang menentukan Amazon DocumentDB sebagai sumbernya. Konfigurasi pipeline sampel ini mengasumsikan penggunaan mekanisme pengambilan aliran perubahan. Lihat Menggunakan pipeline OpenSearch Ingestion dengan Amazon DocumentDB di Panduan Pengembang Layanan OpenSearch Amazon untuk informasi selengkapnya.

Batasan

Batasan berikut berlaku untuk integrasi Amazon OpenSearch DocumentDB:

- Hanya satu koleksi Amazon DocumentDB sebagai sumber per pipeline yang didukung.
- Cross-region konsumsi data tidak didukung. Cluster OpenSearch dan domain Amazon DocumentDB Anda harus berada di wilayah yang sama. AWS
- Cross-account konsumsi data tidak didukung. Cluster Amazon DocumentDB OpenSearch dan pipeline konsumsi Anda harus berada di akun yang sama. AWS
- Cluster elastis Amazon DocumentDB tidak didukung. Hanya cluster berbasis instans Amazon DocumentDB yang didukung.
- Pastikan klaster Amazon DocumentDB mengaktifkan autentikasi menggunakan rahasia. AWS AWS rahasia adalah satu-satunya mekanisme otentikasi yang didukung.
- Konfigurasi pipeline yang ada tidak dapat diperbarui untuk menyerap data dari database yang berbeda dengan koleksi and/or yang berbeda. Untuk memperbarui nama and/or kumpulan database dari pipeline, Anda harus membuat pipeline baru.

Menggunakan Amazon DocumentDB tanpa server

Amazon DocumentDB tanpa server adalah konfigurasi penskalaan otomatis sesuai permintaan yang mengotomatiskan proses penyesuaian kapasitas database Amazon DocumentDB Anda secara dinamis berdasarkan permintaan aplikasi. Anda hanya dikenakan biaya untuk sumber daya yang dikonsumsi cluster Anda. Dengan demikian, DocumentDB tanpa server dapat membantu Anda tetap dalam anggaran dan menghindari membayar sumber daya komputer yang tidak Anda gunakan.

Jenis otomatisasi ini sangat berharga untuk lingkungan dengan beban kerja yang sangat bervariasi dan tidak dapat diprediksi, seperti database multitenant, database terdistribusi, dan sistem pengembangan dan pengujian.

Topik

- Kasus penggunaan tanpa server DocumentDB
- Keuntungan dari Amazon DocumentDB tanpa server
- Cara kerja tanpa server Amazon DocumentDB
- Persyaratan dan batasan untuk DocumentDB tanpa server
- Membuat cluster yang menggunakan Amazon DocumentDB tanpa server
- Migrasi ke Amazon DocumentDB tanpa server
- Mengelola Amazon DocumentDB tanpa server
- Batas instans tanpa server Amazon DocumentDB
- Konfigurasi penskalaan tanpa server Amazon DocumentDB
- Pemantauan Amazon DocumentDB tanpa server

Kasus penggunaan tanpa server DocumentDB

Cluster yang disediakan Amazon DocumentDB dan DocumentDB tanpa server mendukung banyak jenis beban kerja basis data, mulai dari lingkungan pengembangan dan pengujian hingga aplikasi yang paling menuntut dan kritis bisnis yang memerlukan skala dan ketersediaan tinggi. Namun DocumentDB serverless menambahkan dimensi lain pada beban kerja pelanggan, yaitu kemampuan untuk mendukung situs web dan aplikasi yang memiliki beban kerja yang tidak terduga.

DocumentDB tanpa server sangat berguna untuk kasus penggunaan berikut:

- Beban kerja variabel — Anda menjalankan beban kerja yang memiliki peningkatan aktivitas yang tiba-tiba dan tidak dapat diprediksi. Contohnya adalah situs info lalu lintas yang mengalami lonjakan aktivitas saat hujan mulai turun. Contoh yang lain adalah situs e-commerce yang mengalami peningkatan lalu lintas ketika Anda menawarkan obral atau promosi khusus. Dengan DocumentDB tanpa server, database Anda secara otomatis menskalakan kapasitas untuk memenuhi kebutuhan beban puncak aplikasi dan menurunkan skala saat lonjakan aktivitas selesai. Dengan DocumentDB tanpa server, Anda tidak perlu lagi menyediakan kapasitas puncak atau rata-rata. Anda dapat menentukan batas kapasitas atas untuk menangani situasi terburuk, dan kapasitas tersebut tidak akan digunakan kecuali jika diperlukan.
  - Perincian penskalaan di DocumentDB tanpa server membantu Anda mencocokkan kapasitas dengan kebutuhan database Anda. Untuk klaster yang disediakan, penskalaan membutuhkan penambahan instance yang sama sekali baru. DocumentDB tanpa server dapat menambahkan setengah DCU ketika hanya sedikit lebih banyak kapasitas yang dibutuhkan. Ini dapat menambahkan 0,5, 1, 1,5, 2, atau setengah DCU tambahan berdasarkan kapasitas tambahan yang diperlukan untuk menangani peningkatan beban kerja. Dan itu dapat menghapus 0,5, 1, 1,5, 2, atau setengah DCU tambahan ketika beban kerja menurun dan kapasitas itu tidak lagi diperlukan.
- Multi-tenant aplikasi — Dengan DocumentDB tanpa server, Anda tidak perlu mengelola kapasitas database secara individual untuk setiap aplikasi di armada Anda. DocumentDB tanpa server mengelola kapasitas database individual untuk Anda.
  - Anda dapat membuat klaster untuk setiap penghuni. Dengan begitu, Anda dapat menggunakan fitur seperti kloning dan pemulihan snapshot untuk meningkatkan ketersediaan tinggi dan pemulihan bencana yang sesuai untuk setiap penyewa.
  - Setiap penghuni mungkin memiliki periode sibuk dan idle tertentu tergantung waktu dalam sehari, periode dalam setahun, acara promosi, dan sebagainya. Setiap klaster dapat memiliki rentang kapasitas yang luas. Dengan begitu, cluster dengan aktivitas rendah dikenakan biaya instans minimal. Setiap klaster dapat dengan cepat menaikkan skala untuk menangani periode aktivitas tinggi.
- Aplikasi baru — Anda menerapkan aplikasi baru dan Anda tidak yakin tentang ukuran instans yang Anda butuhkan. Dengan menggunakan DocumentDB tanpa server, Anda dapat menyiapkan klaster dengan satu atau banyak instance dan memiliki skala otomatis database untuk persyaratan kapasitas aplikasi Anda.
- Mixed-use aplikasi — Misalkan Anda memiliki aplikasi pemrosesan transaksi online (OLTP), tetapi Anda secara berkala mengalami lonjakan lalu lintas kueri. Dengan menentukan tingkatan promosi untuk instance tanpa server DocumentDB dalam klaster, Anda dapat mengonfigurasi klaster Anda

sehingga instance pembaca dapat menskalakan secara independen dari instance penulis untuk menangani beban tambahan. Saat lonjakan penggunaan mereda, instance pembaca menurunkan skala agar sesuai dengan kapasitas instance penulis.

- Perencanaan kapasitas — Misalkan Anda biasanya menyesuaikan kapasitas database Anda, atau memverifikasi kapasitas database optimal untuk beban kerja Anda, dengan memodifikasi kelas instance dari semua instance dalam sebuah cluster. Dengan DocumentDB tanpa server, Anda dapat menghindari overhead administratif ini. Anda dapat menentukan kapasitas minimum dan maksimum yang sesuai dengan menjalankan beban kerja dan memeriksa seberapa besar skala instans sebenarnya.
  - Anda dapat memodifikasi instance yang ada dari yang disediakan ke DocumentDB tanpa server atau dari DocumentDB tanpa server ke yang disediakan. Anda tidak perlu membuat cluster baru atau instance baru dalam kasus seperti itu.
- Pengembangan dan pengujian - Selain menjalankan aplikasi yang paling menuntut, Anda juga dapat menggunakan DocumentDB tanpa server untuk lingkungan pengembangan dan pengujian. Dengan DocumentDB tanpa server, Anda dapat membuat instance dengan kapasitas minimum rendah alih-alih menggunakan kelas instans db.t* yang dapat diburstable. Anda dapat mengatur kapasitas maksimum yang cukup tinggi sehingga instans tersebut masih dapat menjalankan beban kerja yang besar tanpa kehabisan memori. Ketika database tidak digunakan, semua instance diturunkan untuk menghindari biaya yang tidak perlu.

Menggunakan Amazon DocumentDB tanpa server untuk beban kerja yang sudah tersedia

Misalkan Anda sudah memiliki aplikasi DocumentDB yang berjalan pada cluster yang disediakan. Anda dapat memeriksa bagaimana aplikasi akan bekerja dengan DocumentDB tanpa server dengan menambahkan satu atau lebih instance tanpa server DocumentDB ke cluster yang ada sebagai instance pembaca. Anda dapat memeriksa seberapa sering instance pembaca naik dan turun. Anda dapat menggunakan mekanisme failover DocumentDB untuk mempromosikan instance tanpa server DocumentDB menjadi penulis dan memeriksa bagaimana menangani beban kerja. read/write Dengan demikian, Anda dapat beralih dengan waktu henti minimal dan tanpa mengubah titik akhir yang digunakan aplikasi klien Anda. Untuk detail tentang prosedur mengonversi cluster yang ada ke DocumentDB tanpa server, lihat. Migrasi ke Amazon DocumentDB tanpa server

Keuntungan dari Amazon DocumentDB tanpa server

DocumentDB tanpa server ditujukan untuk beban kerja variabel atau “runcing”. Dengan beban kerja yang tidak dapat diprediksi seperti itu, Anda mungkin mengalami kesulitan dalam merencanakan kapan harus mengubah kapasitas basis data Anda. Anda mungkin juga mengalami kesulitan dalam membuat perubahan kapasitas dengan cukup cepat menggunakan mekanisme yang sudah dikenal seperti menambahkan instance atau mengubah kelas instance. DocumentDB tanpa server memberikan keuntungan berikut untuk membantu kasus penggunaan seperti:

- Manajemen kapasitas yang lebih sederhana daripada yang disediakan - DocumentDB tanpa server mengurangi upaya untuk merencanakan ukuran instans dan mengubah ukuran instance saat beban kerja berubah. Ini juga mengurangi upaya untuk mempertahankan kapasitas yang konsisten untuk semua instance dalam sebuah cluster.

- Penskalaan yang lebih cepat dan lebih mudah selama periode aktivitas tinggi — DocumentDB tanpa server menskalakan kapasitas komputasi dan memori sesuai kebutuhan, tanpa gangguan pada transaksi klien atau beban kerja Anda secara keseluruhan. Kemampuan untuk menggunakan instance pembaca dengan DocumentDB tanpa server membantu Anda memanfaatkan penskalaan horizontal selain penskalaan vertikal. Kemampuan untuk menggunakan cluster global DocumentDB berarti Anda dapat menyebarkan beban kerja baca tanpa server DocumentDB Anda di beberapa. Wilayah AWS Kemampuan ini lebih praktis daripada mekanisme penskalaan untuk klaster terprovisi.

- Cost-effective selama periode aktivitas rendah - DocumentDB tanpa server membantu Anda menghindari penyediaan instans yang berlebihan. DocumentDB tanpa server menambahkan sumber daya secara bertahap saat instance meningkat. Anda hanya membayar untuk sumber daya database yang Anda konsumsi. Penggunaan sumber daya tanpa server DocumentDB diukur berdasarkan per detik. Dengan begitu, ketika sebuah instance menurunkan skala, pengurangan penggunaan sumber daya segera didaftarkan.

- Paritas fitur dengan disediakan - Anda dapat menggunakan semua fitur DocumentDB dengan DocumentDB tanpa server. Misalnya, dengan DocumentDB tanpa server, Anda dapat menggunakan instance pembaca, otentikasi database (IAM) AWS Identity and Access Management , dan Performance Insights.

Secara khusus, dengan DocumentDB tanpa server Anda dapat memanfaatkan fitur-fitur berikut dari cluster yang disediakan:

- Instance pembaca - DocumentDB tanpa server dapat memanfaatkan instance pembaca untuk menskalakan secara horizontal. Ketika sebuah cluster berisi satu atau lebih instance pembaca, cluster dapat segera gagal jika terjadi masalah dengan instance penulis.
- Multi-AZ cluster - Anda dapat mendistribusikan instance tanpa server DocumentDB dari sebuah cluster di beberapa Availability Zones (AZ). Menyiapkan Multi-AZ cluster membantu memastikan kelangsungan bisnis bahkan dalam kasus masalah yang jarang terjadi yang memengaruhi seluruh AZ.
- Kluster global - Anda dapat menggunakan DocumentDB tanpa server dalam kombinasi dengan cluster global DocumentDB untuk membuat salinan read-only tambahan dari cluster Anda di tempat lain untuk tujuan pemulihan bencana. Wilayah AWS

Cara kerja tanpa server Amazon DocumentDB

Topik

- Ikhtisar
- Konfigurasi untuk cluster Amazon DocumentDB
- Kapasitas penskalaan tanpa server Amazon DocumentDB
- Penskalaan tanpa server Amazon DocumentDB
- Keadaan menganggur (0,5 DCU)

Ikhtisar

Amazon DocumentDB tanpa server cocok untuk beban kerja yang paling menuntut dan sangat bervariasi. Misalnya, penggunaan basis data Anda mungkin berat untuk waktu yang singkat, diikuti dengan aktivitas ringan dalam waktu lama atau tidak ada aktivitas sama sekali. Beberapa contohnya adalah situs web retail, game, atau olahraga dengan peristiwa promosi berkala, dan basis data yang menghasilkan laporan jika diperlukan. Contoh lainnya adalah lingkungan pengembangan dan pengujian, serta aplikasi baru yang penggunaannya dapat meningkat dengan cepat. Untuk kasus seperti ini dan banyak kasus lainnya, mengonfigurasi kapasitas dengan benar sebelumnya tidak selalu dapat dilakukan dengan model terprovisi. Hal ini juga dapat mengakibatkan biaya yang lebih tinggi jika Anda menetapkan penyediaan yang berlebih dan memiliki kapasitas yang tidak Anda gunakan.

Sebaliknya, cluster yang disediakan DocumentDB cocok untuk beban kerja yang stabil. Dengan cluster yang disediakan, Anda memilih kelas instance yang memiliki jumlah memori, daya CPU, I/O bandwidth, dan sebagainya yang telah ditentukan sebelumnya. Jika beban kerja Anda berubah, Anda secara manual memodifikasi kelas instans penulis dan pembaca Anda. Model terprovisi berfungsi dengan baik saat Anda dapat menyesuaikan kapasitas sebelum pola konsumsi yang diharapkan dan Anda sanggup mengalami pemadaman singkat saat mengubah kelas instans penulis dan pembaca di klaster Anda.

DocumentDB tanpa server dirancang dari bawah ke atas untuk mendukung cluster tanpa server yang dapat diskalakan secara instan. DocumentDB tanpa server direkayasa untuk memberikan tingkat keamanan dan isolasi yang sama seperti penulis dan pembaca yang disediakan. Aspek-aspek ini sangat penting dalam lingkungan cloud nirserver multi-penghuni. Mekanisme penskalaan dinamis memiliki overhead yang sangat sedikit sehingga dapat merespons dengan cepat perubahan beban kerja basis data. Mekanisme ini juga cukup kuat untuk memenuhi permintaan memproses permintaan yang meningkat secara drastis.

Dengan menggunakan DocumentDB tanpa server, Anda dapat membuat cluster DocumentDB tanpa dikunci ke dalam kapasitas database tertentu untuk setiap penulis dan pembaca. Anda menentukan rentang kapasitas minimum dan maksimum. DocumentDB menskalakan setiap penulis atau pembaca tanpa server DocumentDB di cluster dalam rentang kapasitas tersebut. Dengan menggunakan Multi-AZ cluster di mana setiap penulis atau pembaca dapat menskalakan secara dinamis, Anda dapat memanfaatkan penskalaan dinamis dan ketersediaan tinggi.

DocumentDB tanpa server menskalakan sumber daya database secara otomatis berdasarkan spesifikasi kapasitas minimum dan maksimum Anda. Penskalaan dilakukan dengan cepat karena sebagian besar operasi peristiwa penskalaan mempertahankan penulis atau pembaca di host yang sama. Dalam kasus yang jarang terjadi ketika penulis atau pembaca tanpa server DocumentDB dipindahkan dari satu host ke host lainnya, DocumentDB tanpa server mengelola koneksi secara otomatis. Anda tidak perlu mengubah kode aplikasi klien basis data Anda atau string koneksi basis data Anda.

Dengan DocumentDB tanpa server, seperti halnya cluster yang disediakan, kapasitas penyimpanan dan kapasitas komputasi terpisah. Ketika kita mengacu pada kapasitas dan penskalaan tanpa server DocumentDB, selalu kapasitas komputasi yang meningkat atau menurun. Dengan demikian, klaster Anda dapat berisi banyak terabyte data bahkan ketika kapasitas CPU dan memori diturunkan skalanya ke tingkat rendah.

Alih-alih menyediakan dan mengelola server basis data, Anda menentukan kapasitas basis data. Kapasitas sebenarnya dari setiap penulis atau pembaca tanpa server DocumentDB bervariasi dari

waktu ke waktu, tergantung pada beban kerja Anda. Untuk detail tentang mekanisme tersebut, lihat Penskalaan tanpa server Amazon DocumentDB.

Konfigurasi untuk cluster Amazon DocumentDB

Untuk setiap cluster Amazon DocumentDB Anda, Anda dapat memilih kombinasi kapasitas tanpa server DocumentDB, kapasitas yang disediakan, atau keduanya.

Anda dapat mengatur cluster yang berisi DocumentDB tanpa server dan kapasitas yang disediakan, yang disebut cluster konfigurasi campuran. Misalnya, anggaplah Anda membutuhkan read/write kapasitas lebih dari yang tersedia untuk penulis tanpa server DocumentDB. Dalam hal ini, Anda dapat mengatur klaster dengan penulis terprovisi yang sangat besar. Kemudian Anda masih dapat menggunakan DocumentDB tanpa server untuk pembaca. Atau anggaplah beban kerja tulis untuk klaster Anda bervariasi, tetapi beban kerja baca stabil. Dalam hal ini, Anda dapat mengatur cluster Anda dengan penulis tanpa server DocumentDB dan satu atau lebih pembaca yang disediakan.

Anda juga dapat mengatur cluster di mana semua kapasitas dikelola oleh DocumentDB tanpa server. Untuk melakukan ini, Anda dapat membuat cluster baru dan menggunakan DocumentDB tanpa server dari awal. Atau Anda dapat mengganti semua kapasitas yang disediakan di cluster yang ada dengan DocumentDB tanpa server. Untuk prosedur untuk membuat cluster baru dengan DocumentDB serverless atau untuk mengalihkan cluster yang ada ke DocumentDB serverless, lihat dan. Membuat cluster yang menggunakan Amazon DocumentDB tanpa server Migrasi ke Amazon DocumentDB tanpa server

Jika Anda tidak menggunakan DocumentDB tanpa server sama sekali dalam sebuah cluster, semua penulis dan pembaca di cluster disediakan. Ini adalah jenis cluster yang paling umum yang sebagian besar pengguna akrab dengan. Kapasitas terprovisi bersifat konstan. Biayanya relatif mudah untuk diperkirakan. Namun, Anda harus memprediksi sebelumnya berapa banyak kapasitas yang Anda butuhkan. Dalam beberapa kasus, prediksi Anda mungkin tidak akurat atau kebutuhan kapasitas Anda mungkin berubah. Dalam kasus ini, klaster Anda dapat menjadi underprovisioned (lebih lambat dari yang Anda inginkan) atau overprovisioned (lebih mahal dari yang Anda inginkan).

Kapasitas penskalaan tanpa server Amazon DocumentDB

Satuan ukuran untuk Amazon DocumentDB serverless adalah DocumentDB Capacity Unit (DCU). Kapasitas penskalaan tanpa server DocumentDB tidak terkait dengan kelas instans yang Anda gunakan untuk klaster yang disediakan.

Setiap DCU adalah kombinasi dari sekitar 2 gibibytes (GiB) memori, CPU yang sesuai, dan jaringan. Anda menentukan rentang kapasitas basis data menggunakan unit ukur ini. Metrik

`DCUUtilization` CloudWatch dan `ServerlessDatabaseCapacity` membantu Anda menentukan berapa banyak kapasitas yang sebenarnya digunakan basis data Anda dan di mana kapasitas tersebut berada dalam rentang yang ditentukan.

Setiap saat, setiap penulis atau pembaca tanpa server DocumentDB memiliki kapasitas. Kapasitasnya adalah angka floating-point yang mewakili DCU. Kapasitas ini meningkat atau menurun setiap kali penulis atau pembaca diskalakan. Nilai ini diukur setiap detik. Untuk setiap cluster tempat Anda ingin menggunakan DocumentDB tanpa server, Anda menentukan rentang kapasitas: nilai kapasitas minimum dan maksimum yang dapat diskalakan oleh setiap penulis atau pembaca tanpa server DocumentDB. Rentang kapasitasnya sama untuk setiap penulis atau pembaca tanpa server DocumentDB dalam sebuah cluster. Setiap penulis atau pembaca tanpa server DocumentDB memiliki kapasitasnya sendiri, berada di suatu tempat dalam kisaran itu.

DocumentDB tanpa server didukung pada DocumentDB 5.0 dan 8.0 dengan rentang kapasitas 0,5 - 256 DCU.

Kapasitas tanpa server DocumentDB terkecil yang dapat Anda tentukan adalah 0,5 DCU. Anda dapat menentukan angka yang lebih tinggi jika kurang dari atau sama dengan nilai kapasitas maksimum yang didukung. Menyetel kapasitas minimum ke jumlah kecil memungkinkan cluster yang dimuat ringan mengkonsumsi sumber daya komputasi minimal. Pada saat yang sama, klaster tersebut tetap siap untuk menerima koneksi dengan segera dan menaikkan skalanya ketika menjadi sibuk.

Kami merekomendasikan pengaturan minimum ke nilai yang memungkinkan setiap penulis atau pembaca untuk memegang set kerja aplikasi di kolam buffer. Dengan begitu, konten pool buffer tidak dibuang selama periode idle. Untuk semua pertimbangan saat memilih rentang kapasitas penskalaan, lihat. Memilih rentang kapasitas penskalaan untuk cluster tanpa server DocumentDB

Bergantung pada bagaimana Anda mengkonfigurasi pembaca dalam Multi-AZ penyebaran, kapasitas mereka dapat dikaitkan dengan kapasitas penulis atau secara mandiri. Untuk detail tentang cara melakukannya, lihat Melihat dan memodifikasi tingkat promosi pembaca tanpa server.

Pemantauan DocumentDB tanpa server melibatkan pengukuran nilai kapasitas untuk penulis dan pembaca di cluster Anda dari waktu ke waktu. Jika basis data Anda tidak menurunkan skalanya ke kapasitas minimum, Anda dapat mengambil tindakan seperti menyesuaikan kapasitas minimum dan mengoptimalkan aplikasi basis data Anda. Jika basis data Anda secara konsisten mencapai kapasitas maksimumnya, Anda dapat mengambil tindakan seperti meningkatkan kapasitas maksimum. Anda juga dapat mengoptimalkan aplikasi basis data Anda dan menyebarkan beban kueri ke lebih banyak pembaca.

Biaya untuk kapasitas tanpa server DocumentDB diukur dalam hal. DCU-hours Untuk informasi tentang cara menghitung biaya tanpa server DocumentDB, lihat harga Amazon DocumentDB. Misalkan jumlah total penulis dan pembaca di cluster Anda adalah n. Dalam hal ini, cluster mengkonsumsi sekitar n x DCU minimum ketika Anda tidak menjalankan operasi database apa pun. Amazon DocumentDB sendiri mungkin menjalankan operasi pemantauan atau pemeliharaan yang menyebabkan sejumlah kecil beban. Cluster itu mengkonsumsi tidak lebih dari n x DCU maksimum ketika database berjalan pada kapasitas penuh.

Untuk detail selengkapnya tentang memilih nilai DCU minimum dan maksimum yang sesuai, lihatMemilih rentang kapasitas penskalaan untuk cluster tanpa server DocumentDB. Nilai DCU minimum dan maksimum yang Anda tentukan juga memengaruhi beberapa batas instans Amazon DocumentDB. Untuk detail tentang interaksi antara rentang kapasitas dan batas instans, lihatBatas instans tanpa server Amazon DocumentDB.

Penskalaan tanpa server Amazon DocumentDB

Untuk setiap penulis atau pembaca tanpa server DocumentDB, Amazon DocumentDB terus melacak pemanfaatan sumber daya seperti CPU, memori, dan jaringan. Pengukuran ini secara kolektif disebut beban. Beban mencakup operasi basis data yang dilakukan oleh aplikasi Anda. Ini juga mencakup pemrosesan latar belakang untuk server database dan tugas administratif Amazon DocumentDB. Ketika kapasitas dibatasi oleh salah satu dari ini, DocumentDB tanpa server meningkat. DocumentDB tanpa server juga meningkat ketika mendeteksi masalah kinerja yang dapat diselesaikan dengan melakukannya. Anda dapat memantau pemanfaatan sumber daya dan bagaimana pengaruhnya terhadap penskalaan tanpa server DocumentDB dengan menggunakan prosedur di. Pemantauan Amazon DocumentDB tanpa server

Beban dapat bervariasi di seluruh penulis dan pembaca di cluster Anda. Penulis menangani operasi menulis, dan melakukan semua modifikasi data pada volume cluster. Pembaca dapat memproses permintaan hanya-baca.

Scaling adalah operasi yang meningkatkan atau mengurangi kapasitas tanpa server DocumentDB untuk database Anda. Dengan DocumentDB tanpa server, setiap penulis dan pembaca memiliki nilai kapasitasnya sendiri saat ini, diukur dalam DCU. DocumentDB tanpa server menskalakan penulis atau pembaca hingga kapasitas yang lebih tinggi ketika kapasitas saat ini terlalu rendah untuk menangani beban. Operasi ini menurunkan skala penulis atau pembaca ke kapasitas yang lebih rendah ketika kapasitas saat ini lebih tinggi dari yang dibutuhkan.

DocumentDB tanpa server dapat meningkatkan kapasitas secara bertahap. Ketika permintaan beban kerja Anda mulai mencapai kapasitas database penulis atau pembaca saat ini, DocumentDB

tanpa server meningkatkan jumlah DCU untuk penulis atau pembaca itu. DocumentDB tanpa server menskalakan kapasitas dalam peningkatan yang diperlukan untuk memberikan kinerja terbaik bagi sumber daya yang dikonsumsi. Penskalaan terjadi secara bertahap sekecil 0,5 DCU. Semakin besar kapasitas saat ini, semakin besar inkremen penskalaan dan akibatnya semakin cepat penskalaan dapat terjadi.

Karena penskalaan tanpa server DocumentDB sangat sering, terperinci, dan tidak mengganggu, itu tidak menyebabkan peristiwa diskrit di file. Konsol Manajemen AWS Sebagai gantinya, Anda dapat mengukur CloudWatch metrik Amazon seperti `serverlessDatabaseCapacity` dan`DCUUtilization`, dan melacak nilai minimum, maksimum, dan rata-ratanya dari waktu ke waktu. Untuk mempelajari selengkapnya tentang pemantauan DocumentDB tanpa server, lihat. Pemantauan Amazon DocumentDB tanpa server

Penskalaan naik atau turun dapat disebabkan oleh hal-hal berikut:

- Pemanfaatan memori
- Penggunaan CPU
- Penggunaan jaringan
- Pemanfaatan penyimpanan

Anda dapat memantau penyebab penskalaan ini up/down pada instance tanpa server DocumentDB. Untuk informasi selengkapnya, lihat Pemantauan Amazon DocumentDB tanpa server.

Anda dapat memilih untuk membuat pembaca diskalakan pada saat yang sama dengan penulis terkait, atau secara independen dari penulis. Anda dapat melakukannya dengan menentukan tingkat promosi untuk pembaca tersebut.

- Pembaca tanpa server DocumentDB, dalam tingkatan promosi 0 dan 1, skala pada saat yang sama dengan penulis. Perilaku penskalaan tersebut membuat pembaca di tingkat prioritas 0 dan 1 cocok untuk mendukung ketersediaan. Hal ini karena pembaca tersebut selalu diatur ukurannya sesuai dengan kapasitas yang tepat untuk mengambil alih beban kerja dari penulis jika terjadi failover.
- Pembaca di tingkat promosi 2–15 akan diskalakan secara independen dari penulis. Setiap pembaca tetap berada dalam nilai DCU minimum dan maksimum yang Anda tentukan untuk cluster Anda. Ketika pembaca diskalakan secara independen dari DB penulis terkait, pembaca tersebut bisa menjadi idle dan menurunkan skalanya sementara penulis terus memproses volume transaksi yang tinggi. Pembaca tersebut masih tersedia sebagai target failover jika tidak ada pembaca lain

yang tersedia di tingkat promosi yang lebih rendah. Namun, jika dipromosikan menjadi penulis, pembaca tersebut mungkin perlu dinaikkan skalanya untuk menangani beban kerja penuh dari penulis.

Untuk detail tentang melihat dan mengubah tingkatan promosi instans tanpa server, lihat. Melihat dan memodifikasi tingkat promosi pembaca tanpa server

Penskalaan tanpa server DocumentDB dapat terjadi saat koneksi database terbuka, saat transaksi sedang dalam proses, dll. DocumentDB tanpa server tidak menunggu titik tenang untuk memulai penskalaan. Penskalaan tidak mengganggu operasi basis data apa pun yang sedang berlangsung.

Jika beban kerja Anda membutuhkan lebih banyak kapasitas baca daripada yang tersedia dengan satu penulis dan satu pembaca, Anda dapat menambahkan beberapa pembaca tanpa server DocumentDB ke cluster. Setiap pembaca tanpa server DocumentDB dapat menskalakan dalam kisaran nilai kapasitas minimum dan maksimum yang Anda tentukan untuk klaster Anda. Anda dapat menggunakan titik akhir pembaca klaster untuk mengarahkan sesi hanya baca ke pembaca dan mengurangi beban pada penulis.

Apakah DocumentDB tanpa server melakukan penskalaan, dan seberapa cepat penskalaan terjadi begitu dimulai, juga tergantung pada pengaturan DCU minimum dan maksimum untuk cluster. Selain itu, hal tersebut bergantung pada apakah pembaca dikonfigurasi untuk diskalakan bersama dengan penulis atau secara independen. Untuk detail tentang konfigurasi penskalaan, lihatKonfigurasi penskalaan tanpa server Amazon DocumentDB.

Keadaan menganggur (0,5 DCU)

Saat Amazon DocumentDB penulis atau pembaca tanpa server menganggur, instance tanpa server DocumentDB mendukung penskalaan ke status idle 0,5 DCU jika cluster dikonfigurasi menjadi 0,5. MinCapacity

Dalam keadaan idle, instance tanpa server DocumentDB tidak memiliki kapasitas komputasi CPU yang memadai untuk mendukung sebagian besar beban kerja produksi, tetapi siap untuk ditingkatkan dengan cepat untuk mendukung beban kerja baru. Dalam keadaan non-idle, instance tanpa server DocumentDB biasanya memerlukan setidaknya 1,0 - 2,5 DCU. Oleh karena itu, ketika instance tanpa server DocumentDB meningkat dari status idle ke status non-idle, mereka akan menskalakan langsung ke 1,0 - 2,5 DCU (atau nilai jika lebih rendah). MaxCapacity

Untuk mendukung penskalaan hingga 0,5 DCU saat idle, batas instance dibatasi jika MinCapacity dikonfigurasi menjadi kurang dari atau sama dengan 1,0 DCU. Untuk informasi selengkapnya tentang

bagaimana batas dipengaruhi oleh MinCapacity konfigurasi, lihatBatas instans tanpa server Amazon DocumentDB.

Persyaratan dan batasan untuk DocumentDB tanpa server

Persyaratan fitur

Ketersediaan wilayah

Jenis instans tanpa server Amazon DocumentDB tersedia di wilayah berikut:

AWS CLI Perintah berikut dapat digunakan untuk memverifikasi opsi instance tanpa server DocumentDB yang ditawarkan di wilayah tertentu:

```
aws docdb describe-orderable-db-instance-options \
        --region my_region \
        --db-instance-class db.serverless \
        --engine docdb
```

Ketersediaan versi mesin

DocumentDB tanpa server didukung oleh versi mesin Amazon DocumentDB 5.0 dan 8.0. Ini tidak tersedia pada versi mesin 3.6 atau 4.0.

> **Note**
>
> DocumentDB tanpa server hanya didukung pada versi patch 5.0.0 yang lebih baru. Pastikan klaster Anda diperbarui ke versi patch mesin terbaru. Untuk informasi selengkapnya tentang memelihara klaster Amazon DocumentDB, lihat Memelihara Amazon DocumentDB

Konfigurasi klaster

Sebelum menambahkan instance tanpa server Amazon DocumentDB ke cluster Amazon DocumentDB, cluster juga harus memiliki parameter yang ditetapkan. `ServerlessV2ScalingConfiguration` Ini mendefinisikan rentang kapasitas instance tanpa server DocumentDB di cluster. Untuk informasi selengkapnya tentang konfigurasi penskalaan, lihatKonfigurasi penskalaan tanpa server Amazon DocumentDB.

Pengaturan rentang kapasitas penskalaan minimum untuk fitur Amazon DocumentDB tertentu

Beberapa fitur Amazon DocumentDB bekerja dengan DocumentDB tanpa server, tetapi dapat menyebabkan masalah jika rentang kapasitas Anda lebih rendah dari yang dibutuhkan untuk persyaratan memori untuk fitur tersebut dengan beban kerja spesifik Anda. Dalam hal ini, basis data Anda mungkin tidak berfungsi sebaik biasanya, atau mungkin mengalami kesalahan kehabisan memori.

Fitur-fitur berikut memerlukan konfigurasi `MinCapacity` and/or `MaxCapacity` nilai yang lebih tinggi untuk operasi terbaik:

- Wawasan Performa
- Cluster global (hanya berlaku untuk primer Wilayah AWS)
- Pembuatan instance tanpa server pada cluster dengan volume data yang besar

  Ini termasuk pembuatan instance tanpa server sebagai bagian dari pemulihan cluster.

Untuk rekomendasi tentang pengaturan rentang kapasitas yang sesuai (jika Anda menggunakan fitur ini), lihatMemilih rentang kapasitas penskalaan untuk cluster tanpa server DocumentDB. Untuk informasi pemecahan masalah jika basis data Anda mengalami kesalahan kehabisan memori karena rentang kapasitas yang salah dikonfigurasi, lihat Menghindari kesalahan kehabisan memori.

Membuat cluster yang menggunakan Amazon DocumentDB tanpa server

Membuat klaster tanpa server Amazon DocumentDB

Dengan Amazon DocumentDB tanpa server, cluster Anda dapat dipertukarkan dengan cluster yang disediakan. Anda dapat memiliki cluster di mana beberapa instance menggunakan tanpa server dan beberapa instance disediakan.

Verifikasi bahwa wilayah dan versi engine yang Anda inginkan mendukung DocumentDB tanpa server. Lihat Persyaratan dan batasan untuk DocumentDB tanpa server.

Untuk membuat klaster Amazon DocumentDB tempat Anda dapat menambahkan instance tanpa server, ikuti prosedur yang sama seperti di. Membuat cluster Amazon DocumentDB Satu-

satunya perbedaan adalah bahwa `ServerlessV2ScalingConfiguration` argumen juga harus disediakan.

`ServerlessV2ScalingConfiguration`Argumen menentukan rentang kapasitas penskalaan instance tanpa server DocumentDB Anda. Ini terdiri dari nilai unit kapasitas DocumentDB (DCU) minimum dan maksimum yang berlaku untuk semua instance tanpa server DocumentDB di cluster:

- `MinCapacity`Nilai menentukan kapasitas penskalaan minimum.
- `MaxCapacity`Nilai menentukan kapasitas penskalaan maksimum.

Untuk informasi selengkapnya tentang penskalaan, lihat Konfigurasi penskalaan tanpa server Amazon DocumentDB.

Using the Konsol Manajemen AWS

Contoh Konsol Manajemen AWS konfigurasi berikut menunjukkan cara membuat cluster tanpa server DocumentDB.

1. Masuk ke Konsol Manajemen AWSdan buka konsol Amazon DocumentDB.
2. Di panel navigasi, pilih Klaster.

   > **Tip**
   >
   > Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu (≡) di pojok kiri atas halaman.

   Tabel Clusters muncul.
3. Pilih Buat.

   Halaman cluster Create Amazon DocumentDB muncul.
4. Pada halaman cluster Create Amazon DocumentDB, di bagian tipe Cluster, Instance-based pilih cluster (ini adalah opsi default).
5. Di bagian konfigurasi Cluster:

a. Untuk pengidentifikasi Cluster, masukkan nama unik, seperti**`myserverlesscluster`**. Perhatikan bahwa konsol akan mengubah semua nama cluster menjadi huruf kecil terlepas dari bagaimana mereka dimasukkan.

b. Untuk versi Engine, pilih 5.0.0 (ini adalah opsi default).

6. Di bagian konfigurasi penyimpanan Cluster, pilih Amazon DocumentDB Standard (ini adalah opsi default).

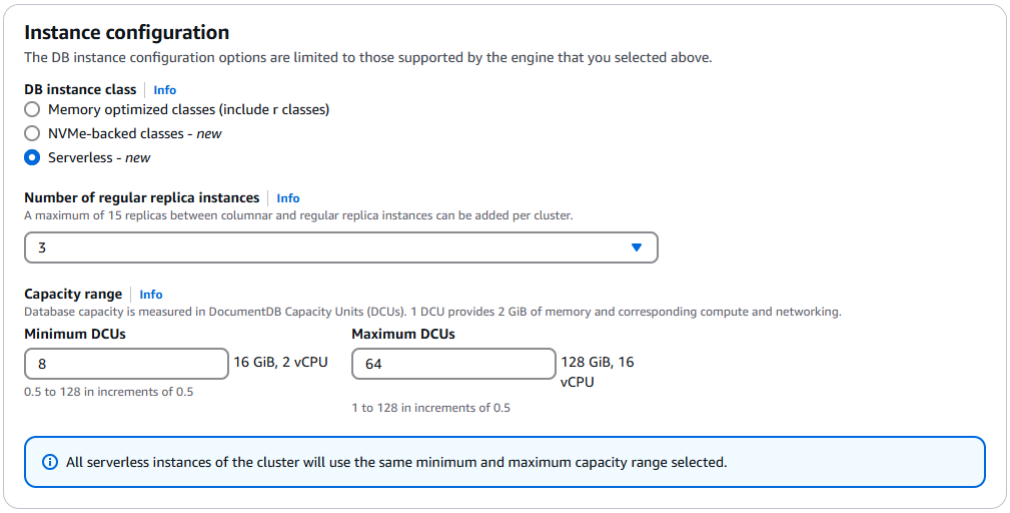
> **Note**
>
> Opsi lain dalam kategori ini adalah Amazon I/O-Optimized DocumentDB. Untuk mempelajari lebih lanjut tentang salah satu opsi, lihat Konfigurasi penyimpanan cluster Amazon DocumentDB

7. Di bagian konfigurasi Instans:

a. Untuk Kelas instans DB, pilih Nirserver.

b. Untuk Jumlah contoh replika biasa, pilih 3 (ini adalah opsi default).

c. Di bagian rentang Kapasitas, tinggalkan nilai default untuk DCU Minimum dan DCU Maksimum. Untuk informasi tentang pengaturan parameter ini, lihatBatas instans tanpa server Amazon DocumentDB.



8. Di bagian Konektivitas, biarkan pengaturan default Jangan sambungkan ke sumber daya komputasi EC2.

9. Di bagian Autentikasi, masukkan nama pengguna untuk pengguna utama, lalu pilih Self managed. Masukkan kata sandi, lalu konfirmasikan.

   Jika Anda memilih Managed in AWS Secrets Manager, lihat Manajemen kata sandi dengan Amazon DocumentDB dan AWS Secrets Manager untuk informasi selengkapnya.

10. Biarkan semua opsi lain sebagai default dan pilih Buat cluster.

Using the AWS CLI

Dalam contoh berikut, ganti masing-masing *user input placeholder* dengan informasi atau parameter konfigurasi Anda sendiri.

Untuk membuat cluster yang kompatibel dengan instance tanpa server DocumentDB menggunakan, lihat. AWS CLIMembuat cluster menggunakan AWS CLI

Sertakan parameter tambahan berikut dalam `create-db-cluster` perintah Anda:

```
--serverless-v2-scaling-configuration
     MinCapacity=minimum_capacity,MaxCapacity=maximum_capacity
```

Contoh:

```
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
```

Menambahkan instance tanpa server Amazon DocumentDB

Untuk menambahkan instance tanpa server DocumentDB, ikuti prosedur yang sama Menambahkan instance Amazon DocumentDB ke klaster di, pastikan untuk menentukan db.serverless sebagai kelas instance.

Menambahkan instance tanpa server menggunakan Konsol Manajemen AWS.

Untuk menambahkan instance tanpa server DocumentDB menggunakan konsol, Menambahkan instance Amazon DocumentDB ke klaster lihat dan pilih Menggunakan tab. Konsol Manajemen AWS

Menambahkan instance tanpa server menggunakan AWS CLI

Untuk menambahkan instance tanpa server DocumentDB menggunakan, Menambahkan instance Amazon DocumentDB ke klaster lihat dan pilih Menggunakan AWS CLI tab. AWS CLI

Gunakan argumen CLI kelas contoh berikut:

```
--db-instance-class db.serverless
```

Contoh:

```
aws docdb create-db-instance \
      --db-cluster-identifier sample-cluster \
      --db-instance-identifier sample-instance \
      --db-instance-class db.serverless \
      --engine docdb
```

Migrasi ke Amazon DocumentDB tanpa server

Topik

- Migrasi cluster DocumentDB yang ada ke tanpa server
- Migrasi dari MongoDB ke DocumentDB tanpa server

Migrasi cluster DocumentDB yang ada ke tanpa server

Memutakhirkan versi mesin cluster

Jika klaster yang disediakan menjalankan versi engine yang lebih rendah yang tidak mendukung DocumentDB tanpa server, Anda harus terlebih dahulu memutakhirkan klaster ke versi engine yang didukung. Untuk informasi selengkapnya, lihat Peningkatan versi utama Amazon DocumentDB di tempat.

Migrasi cluster yang disediakan ke DocumentDB tanpa server

Untuk mengganti klaster yang disediakan agar menggunakan DocumentDB tanpa server, ikuti langkah-langkah berikut:

1. Periksa apakah versi engine cluster yang disediakan perlu ditingkatkan untuk digunakan dengan DocumentDB tanpa server. Lihat Persyaratan dan batasan untuk DocumentDB tanpa server.

   > **Note**
   >
   > Jika klaster yang disediakan menjalankan versi engine yang tidak tersedia untuk DocumentDB tanpa server, tingkatkan versi mesin cluster. Lihat Peningkatan versi utama Amazon DocumentDB di tempat.

2. Konfigurasikan konfigurasi penskalaan untuk klaster. Untuk detail tentang memilih konfigurasi penskalaan, lihatMemilih rentang kapasitas penskalaan untuk cluster tanpa server DocumentDB. Untuk memodifikasi konfigurasi penskalaan klaster, lihatMelihat dan memodifikasi konfigurasi rentang kapasitas penskalaan klaster.
3. Konfigurasikan properti cluster lainnya untuk memperhitungkan persyaratan dan batasan tanpa server DocumentDB. Persyaratan dan batasan untuk DocumentDB tanpa server
4. Tambahkan satu atau beberapa instance tanpa server DocumentDB ke cluster. Ikuti prosedur di Menambahkan instance tanpa server Amazon DocumentDB.

   > **Note**
   >
   > Dalam beberapa kasus, Anda mungkin sudah memiliki satu atau lebih instance pembaca yang disediakan di cluster. Jika demikian, Anda dapat memilih untuk mengonversi salah satu pembaca ke instance tanpa server DocumentDB alih-alih membuat instance baru. Untuk melakukannya, ikuti prosedurnya di Mengubah kelas instance.

5. (Opsional) Lakukan operasi failover untuk menjadikan instance tanpa server DocumentDB sebagai penulis cluster. Lihat Failover Amazon DocumentDB.
6. (Opsional) Konversi instans Amazon DocumentDB yang tersisa yang disediakan ke instance Mengubah kelas instance tanpa server DocumentDB (lihat) atau hapus dari cluster (lihat). Menghapus instans Amazon DocumentDB

Using the Konsol Manajemen AWS

Contoh Konsol Manajemen AWS konfigurasi berikut menunjukkan proses migrasi menggunakan kluster yang disediakan Amazon DocumentDB yang menjalankan Amazon DocumentDB 5.0.0, yang tidak memerlukan upgrade versi engine untuk mulai menggunakan DocumentDB tanpa server. Cluster diberi nama`sample`, dan dimulai dengan tiga instance yang disediakan bernama`sample`,, `sample2` dan. `sample3` Dalam contoh ini, ketiga instance ini akan digantikan oleh tiga instance tanpa server.

1. Masuk ke Konsol Manajemen AWSdan buka konsol Amazon DocumentDB.
2. Di panel navigasi, pilih Klaster.

   > **Tip**
   >
   > Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu (≡) di pojok kiri atas halaman.

   Tabel Clusters muncul.
3. Dalam tabel Clusters, pilih kotak centang cluster yang ingin Anda tambahkan instance tanpa server.
4. Pilih Tindakan, dan kemudian pilih Tambahkan instans.
5. Dalam kotak dialog Add instance, pilih Tanpa Server di bagian kelas instans DB untuk setiap instance tanpa server baru yang ingin Anda buat.
6. Untuk pengaturan kapasitas tanpa server, atur konfigurasi penskalaan berdasarkan deskripsi kapasitas di kotak dialog.
7. (Opsional) Untuk menambahkan contoh lain, pilih Tambahkan instance. Lanjutkan menambahkan instance hingga Anda mencapai jumlah instans baru yang diinginkan.

   Dalam contoh ini, tiga instance tanpa server baru dibuat.
8. Pilih Buat.

   Dibutuhkan beberapa menit untuk membuat sebuah instans. Anda dapat menggunakan konsol atau AWS CLI untuk melihat status instans. Untuk informasi selengkapnya, lihat Memantau Status Cluster Amazon DocumentDB.
9. Kembali ke tabel Clusters, pilih kotak centang dari ketiga instance asli yang disediakan.

10. Pilih Tindakan, lalu pilih Hapus.

Selama penghapusan, failover secara otomatis dilakukan untuk mempromosikan salah satu contoh yang tersisa untuk menjadi penulis. Setelah beberapa menit, proses penghapusan selesai. Cluster yang ada sekarang berisi tiga instance tanpa server DocumentDB (seperti yang didefinisikan dalam kolom Ukuran).

Using the AWS CLI

Contoh AWS CLI konfigurasi berikut menunjukkan proses migrasi menggunakan kluster yang disediakan Amazon DocumentDB yang menjalankan Amazon DocumentDB 5.0.0, yang tidak memerlukan upgrade versi engine untuk mulai menggunakan DocumentDB tanpa server. Cluster diberi nama`sample`, dan dimulai dengan tiga instance yang disediakan bernama`sample`,, `sample2` dan. `sample3` Dalam contoh ini, ketiga instance ini akan digantikan oleh tiga instance tanpa server. Cluster diberi nama`sample-cluster`, dan dimulai dengan dua instance yang disediakan bernama `sample-provisioned-instance-1` dan`sample-provisioned-instance-2`, instance penulis dan instance pembaca.

Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi atau parameter konfigurasi Anda sendiri.

Gunakan `aws docdb describe-db-clusters` operasi untuk menentukan status cluster. Kode berikut menemukan status cluster `sample-cluster` dan output hasil dalam tabel:

```
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             |
+-------------------------------+---------------------------------
+-----------------+---------------+
```

Tambahkan konfigurasi penskalaan untuk cluster:

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --serverless-v2-scaling-configuration MinCapacity=0.5,MaxCapacity=16
```

Tambahkan instance tanpa server. Dalam contoh ini, instance tanpa server baru bernama `sample-serverless-instance-1` dan `sample-serverless-instance-2` ditambahkan:

```
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
```

Masukkan yang berikut ini untuk menunggu instans tanpa server tersedia sebelum melanjutkan:

```
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
```

Lakukan failover untuk menjadikan `sample-serverless-instance-1` instance baru sebagai penulis cluster:

```
aws docdb failover-db-cluster \
    --db-cluster-identifier sample-cluster \
    --target-db-instance-identifier sample-serverless-instance-1
```

Failover membutuhkan beberapa detik untuk diselesaikan, setelah itu sample-serverless-instance-1 menjadi penulis cluster. Verifikasi ini dengan masukan berikut:

```
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             |
+---------------------------------+---------------------------------
+------------------+----------------+
```

Terakhir, hapus instance asli yang disediakan:

```
aws docdb delete-db-instance \
    --db-instance-identifier sample-provisioned-instance-1

aws docdb delete-db-instance \
    --db-instance-identifier sample-provisioned-instance-2
```

Migrasi dari MongoDB ke DocumentDB tanpa server

Anda dapat memigrasikan database MongoDB Anda ke DocumentDB tanpa server, seperti halnya Amazon DocumentDB yang disediakan. Untuk informasi selengkapnya, lihat Migrasi ke Amazon DocumentDB.

Mengelola Amazon DocumentDB tanpa server

Melihat dan memodifikasi konfigurasi rentang kapasitas penskalaan klaster

`ServerlessV2ScalingConfiguration`Argumen menentukan rentang kapasitas penskalaan instance tanpa server DocumentDB Anda. Ini terdiri dari nilai unit kapasitas DocumentDB minimum dan maksimum (DCU) yang berlaku untuk semua instance tanpa server DocumentDB di cluster.

- **`MinCapacity`**- Kapasitas penskalaan minimum dari setiap instance tanpa server DocumentDB di cluster.
- **`MaxCapacity`**- Kapasitas penskalaan maksimum dari setiap instance tanpa server DocumentDB di cluster.

> **Note**
>
> Modifikasi konfigurasi penskalaan berikut memerlukan instans restart untuk mencerminkan `MaxCapacity` nilai baru `MinCapacity` dan:
>
> - Setiap perubahan pada `MaxCapacity` nilai
> - Mengubah `MinCapacity` nilai menjadi 1,0 atau lebih rendah dari nilai yang lebih tinggi
> - Mengubah `MinCapacity` nilai menjadi lebih besar dari 1,0 dari nilai yang lebih rendah

Untuk informasi selengkapnya tentang konfigurasi penskalaan dan cara memilih batas kapasitas penskalaan yang sesuai, lihat. Konfigurasi penskalaan tanpa server Amazon DocumentDB

Using the Konsol Manajemen AWS

Contoh Konsol Manajemen AWS konfigurasi berikut menunjukkan cara melihat dan mengedit pengaturan konfigurasi penskalaan cluster tanpa server DocumentDB.

1. Masuk ke Konsol Manajemen AWSdan buka konsol Amazon DocumentDB.

2. Di panel navigasi, pilih Klaster.

> **Tip**
>
> Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu (≡) di pojok kiri atas halaman.

Tabel Clusters muncul.

3. Dalam tabel Clusters, pilih kotak centang cluster yang ingin Anda ubah kapasitas penskalaan.
4. Pilih Tindakan, dan kemudian pilih Modifikasi.

Kotak dialog Modify cluster muncul.

5. Temukan bagian pengaturan kapasitas tanpa server dan atur konfigurasi penskalaan (Rentang kapasitas) berdasarkan deskripsi kapasitas di kotak dialog.

Untuk informasi selengkapnya tentang penskalaan dan rentang kapasitas, lihatKonfigurasi penskalaan tanpa server Amazon DocumentDB.

6. Pilih Lanjutkan.
7. Untuk Penjadwalan perubahan, pilih Terapkan langsung.
8. Pilih Ubah klaster.
9. Setelah modifikasi selesai, setiap instance tanpa server harus di-boot ulang. Untuk meminimalkan ketidaktersediaan penulis, lakukan urutan operasi berikut:

   a. Reboot setiap instance pembaca tanpa server.

      i. Pilih instance pembaca, pilih Actions, dan kemudian Reboot.

      ii. Tunggu status instance kembali ke Tersedia.

   b. Lakukan failover ke instance tanpa server yang di-boot ulang.

      i. Pilih cluster, pilih Tindakan, dan kemudian Failover.

      ii. Tunggu hingga operasi failover selesai.

   c. Reboot instance tanpa server yang tersisa.

      i. Pilih instance yang tersisa, pilih Actions, dan kemudian Reboot.

ii. Tunggu status instance kembali ke Tersedia.

Using the AWS CLI

Contoh AWS CLI konfigurasi berikut menunjukkan konfigurasi penskalaan saat ini.

Dalam contoh berikut, ganti masing-masing *user input placeholder* dengan informasi atau parameter konfigurasi Anda sendiri.

Konfigurasi penskalaan cluster saat ini dapat dilihat menggunakan `describe-db-clusters` AWS CLI perintah:

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[0].ServerlessV2ScalingConfiguration'
```

Berikut ini adalah output dari perintah ini:

```
{
    "MinCapacity": 0.5,
    "MaxCapacity": 16.0
}
```

Konfigurasi penskalaan cluster dapat dimodifikasi menggunakan `modify-db-cluster` perintah:

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --serverless-v2-scaling-configuration MinCapacity=0.5,MaxCapacity=32
```

Setelah selesai, setiap instance tanpa server harus di-boot ulang. Untuk meminimalkan ketidaktersediaan penulis, kami dapat melakukan urutan operasi berikut:

```
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
```

Melihat dan memodifikasi tingkat promosi pembaca tanpa server

Untuk cluster yang berisi beberapa instance tanpa server DocumentDB atau campuran instance yang disediakan dan tanpa server, perhatikan pengaturan tingkat promosi untuk setiap instans tanpa server. Setelan ini mengontrol lebih banyak perilaku untuk instance tanpa server daripada instance yang disediakan.

Untuk instance yang disediakan, pilihan tier 0—15 hanya menentukan urutan Amazon DocumentDB memilih instance pembaca mana yang akan dipromosikan kepada penulis selama operasi failover. Namun, untuk instance tanpa server, nomor tingkatan juga menentukan apakah instans menskalakan agar sesuai dengan kapasitas instance penulis atau menskalakan secara independen berdasarkan beban kerjanya sendiri. Instans pembaca tanpa server di tingkat 0 atau 1 disimpan pada kapasitas minimum setidaknya setinggi instance penulis. Dengan begitu, mereka siap untuk mengambil alih dari contoh penulis jika terjadi kegagalan. Jika instance writer adalah instance yang disediakan, Amazon DocumentDB memperkirakan kapasitas tanpa server DocumentDB yang setara. Ini menggunakan estimasi itu sebagai kapasitas minimum untuk instance pembaca tanpa server.

Instance pembaca tanpa server DocumentDB di tingkatan 2-15 tidak memiliki batasan yang sama pada kapasitas minimumnya. Ketika mereka menganggur, mereka dapat menurunkan ke nilai unit kapasitas DocumentDB minimum (DCU) yang ditentukan dalam rentang kapasitas cluster.

Using the Konsol Manajemen AWS

Contoh Konsol Manajemen AWS konfigurasi berikut menunjukkan cara melihat dan memodifikasi setelan tingkat promosi pembaca instans tanpa server DocumentDB.

1. Masuk ke Konsol Manajemen AWSdan buka konsol Amazon DocumentDB.

Tingkatan promosi setiap instans ditampilkan di kolom Tingkat Promosi di. Konsol Manajemen AWS

2. Di panel navigasi, pilih Klaster.

> **Tip**
>
> Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu (≡) di pojok kiri atas halaman.

Tabel Clusters muncul.

3. Di tabel Clusters, pilih kotak centang instance yang ingin Anda ubah tingkatan promosi.
4. Pilih Tindakan, dan kemudian pilih Modifikasi.

Kotak dialog Modify instance muncul.

5. Temukan bagian Failover dan atur tingkat Promosi ke level yang diinginkan.
6. Pilih Lanjutkan.
7. Untuk Penjadwalan perubahan, pilih Terapkan langsung.
8. Pilih Ubah instans.

Using the AWS CLI

Contoh AWS CLI konfigurasi berikut menunjukkan konfigurasi penskalaan saat ini.

Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi atau parameter konfigurasi Anda sendiri.

Tingkatan promosi dari semua instance dalam sebuah cluster dapat dilihat menggunakan perintah: `describe-db-clusters` AWS CLI

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[0].DBClusterMembers' \
    --output table
```

Berikut ini adalah output dari perintah ini:

```
-------------------------------------------------------------------------------------------------
|                                          DescribeDBClusters
                  |
+---------------------------------+---------------------------------
+------------------+----------------+
|  DBClusterParameterGroupStatus |      DBInstanceIdentifier       | IsClusterWriter
  | PromotionTier  |
+---------------------------------+---------------------------------
+------------------+----------------+
|  in-sync                       |  sample-serverless-instance-2   |  False
  |  1             |
|  in-sync                       |  sample-serverless-instance-1   |  True
  |  1             |
+---------------------------------+---------------------------------
+------------------+----------------+
```

Tingkat promosi dari instance tertentu dapat dimodifikasi menggunakan `modify-db-instance` perintah:

```
aws docdb modify-db-instance \
    --db-instance-identifier sample-serverless-instance-2 \
    --promotion-tier 3
```

Batas instans tanpa server Amazon DocumentDB

Untuk instance tanpa server DocumentDB, batas berikut per instance bergantung pada kapasitas penskalaan instans saat ini:

- Memori Instans (GiB)
- Koneksi (semua)
- Batas Kursor
- Buka Transaksi
- Koneksi (aktif)

Tabel berikut menjelaskan bagaimana skala batas per instance tanpa server dalam kaitannya dengan kapasitas penskalaan instans saat ini.

Perhatikan bahwa nilai batas berbeda tergantung pada apakah konfigurasi `MinCapacity` penskalaan cluster telah disetel ke lebih besar dari dua (2). Jika `MinCapacity` diatur ke kurang dari atau sama dengan 2, maka nilai batas untuk batas kursor, transaksi terbuka, dan koneksi (aktif) dibatasi ke nilai maksimum yang lebih rendah. (Alasan teknis untuk cap ini adalah untuk mendukung penskalaan ke status “idle”. Untuk informasi selengkapnya, lihat Keadaan menganggur (0,5 DCU).

db.serverless instance membatasi jika ≤1 MinCapacity

Kapasitas Saat Ini (DCU)	Memori Instans (GiB)	Koneksi (semua)	Batas Kursor	Buka Transaksi	Koneksi (aktif)
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

db.serverless instance membatasi jika >1 MinCapacity

Kapasitas Saat Ini (DCU)	Memori Instans (GiB)	Koneksi (semua)	Batas Kursor	Buka Transaksi	Koneksi (aktif)
1.5	3	7250	18	36	124

Kapasitas Saat Ini (DCU)	Memori Instans (GiB)	Koneksi (semua)	Batas Kursor	Buka Transaksi	Koneksi (aktif)
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

Anda dapat memantau dan alarm pada batas per instans menggunakan CloudWatch metrik berikut. Untuk selengkapnya tentang metrik Amazon CloudWatch DocumentDB, lihat. Memantau Amazon DocumentDB dengan CloudWatch

Sumber daya	CloudWatch batas metrik	CloudWatch metrik penggunaan (maks 1 menit)	CloudWatch metrik penggunaan
Memori Instans	-	-	FreeableMemory
Koneksi (semua)	DatabaseConnectionsLimit	DatabaseConnectionsMax	DatabaseConnections
Cursors	DatabaseCursorsLimit	DatabaseCursorsMax	DatabaseCursors
Transaksi	TransactionsOpenLimit	TransactionsOpenMax	TransactionsOpen

Konfigurasi penskalaan tanpa server Amazon DocumentDB

Topik

- Memilih rentang kapasitas penskalaan untuk cluster tanpa server DocumentDB
- Memilih MinCapacitypengaturan untuk cluster tanpa server DocumentDB
- Memilih MaxCapacitypengaturan untuk cluster tanpa server DocumentDB
- Menghindari kesalahan kehabisan memori
- Mengapa instance tanpa server saya tidak diperkecil?

Memilih rentang kapasitas penskalaan untuk cluster tanpa server DocumentDB

Sebelum menambahkan instance tanpa server DocumentDB ke cluster Amazon DocumentDB, cluster juga harus memiliki set parameter. `ServerlessV2ScalingConfiguration`

`ServerlessV2ScalingConfiguration`Parameter terdiri dari dua nilai yang menentukan rentang kapasitas penskalaan tanpa server dari setiap instance tanpa server di cluster:

- **`MinCapacity`**- Kapasitas penskalaan minimum dari setiap instance tanpa server DocumentDB di cluster.
- **`MaxCapacity`**- Kapasitas penskalaan maksimum dari setiap instance tanpa server DocumentDB di cluster.

Memilih **`MinCapacity`**pengaturan untuk cluster tanpa server DocumentDB

Sangat menggoda untuk selalu memilih 0,5 untuk`MinCapacity`. Nilai itu memungkinkan instance untuk menurunkan kapasitas terkecil saat benar-benar menganggur, sambil tetap aktif. Namun, tergantung pada bagaimana Anda menggunakan cluster itu dan pengaturan lain yang Anda konfigurasikan, kapasitas minimum yang berbeda mungkin yang paling efektif. Pertimbangkan faktor-faktor berikut saat memilih pengaturan kapasitas minimum:

- Tingkat penskalaan untuk instance tanpa server DocumentDB bergantung pada kapasitasnya saat ini. Semakin tinggi kapasitas saat ini, semakin cepat instans DB dapat dinaikkan skalanya. Jika Anda memerlukan instans untuk dengan cepat meningkatkan kapasitas yang sangat tinggi,

pertimbangkan untuk mengatur kapasitas minimum ke nilai di mana tingkat penskalaan memenuhi kebutuhan Anda.

- Jika Anda biasanya memodifikasi kelas instans instans Anda untuk mengantisipasi beban kerja yang sangat tinggi atau rendah, Anda dapat menggunakan pengalaman itu untuk membuat perkiraan kasar rentang kapasitas tanpa server DocumentDB yang setara. Untuk menentukan ukuran memori dari jenis instans Amazon DocumentDB yang disediakan, lihat. Batas instans

  Misalnya, anggaplah Anda menggunakan kelas `db.r6g.xlarge` instance ketika cluster Anda memiliki beban kerja yang rendah. Kelas instance itu memiliki 32 GiB memori. Dengan demikian, Anda dapat menentukan 16 untuk menyiapkan instance tanpa server yang dapat menurunkan skala hingga kira-kira kapasitas yang sama. `MinCapacity` Itu karena setiap DCU sesuai dengan sekitar 2 GiB memori. Anda dapat menentukan nilai yang agak lebih rendah untuk membiarkan skala instance lebih rendah jika `db.r6g.xlarge` instance Anda terkadang kurang dimanfaatkan.
- Jika aplikasi Anda bekerja paling efisien ketika instance memiliki sejumlah data dalam cache buffer, pertimbangkan untuk menentukan pengaturan DCU minimum di mana memori cukup besar untuk menampung data yang sering diakses. Jika tidak, beberapa data dikeluarkan dari cache buffer saat instance tanpa server diturunkan ke ukuran memori yang lebih rendah. Kemudian ketika instance diskalakan kembali, informasi dibaca kembali ke cache buffer dari waktu ke waktu. Jika jumlah I/O untuk membawa data kembali ke cache buffer cukup besar, mungkin lebih efektif untuk memilih nilai DCU minimum yang lebih tinggi. Untuk informasi selengkapnya, lihat Ukuran instans.
- Jika instance tanpa server DocumentDB Anda berjalan sebagian besar waktu pada kapasitas tertentu, pertimbangkan untuk menentukan pengaturan kapasitas minimum yang lebih rendah dari baseline tersebut, tetapi tidak terlalu rendah. Instans tanpa server dapat memperkirakan secara efektif berapa banyak dan seberapa cepat untuk ditingkatkan ketika kapasitas saat ini tidak jauh lebih rendah dari kapasitas yang dibutuhkan.
- Jika beban kerja Anda yang disediakan memiliki persyaratan memori yang terlalu tinggi untuk kelas instans kecil seperti T3 atau T4G, pilih pengaturan DCU minimum yang menyediakan memori yang sebanding dengan instans R5 atau R6g.
- Secara khusus, kami merekomendasikan minimum berikut `MinCapacity` untuk digunakan dengan fitur yang ditentukan (rekomendasi ini dapat berubah):
  - Performance Insights — 2 DCU
  - Cluster global — 8 DCU (hanya berlaku untuk primer) Wilayah AWS
- Di Amazon DocumentDB, replikasi terjadi pada lapisan penyimpanan, sehingga kapasitas pembaca tidak secara langsung memengaruhi replikasi. Namun, untuk instance pembaca tanpa server DocumentDB yang menskalakan secara independen, pastikan kapasitas minimum cukup

untuk menangani beban kerja selama periode intensif penulisan untuk menghindari latensi kueri. Jika instance pembaca dalam tingkatan promosi 2—15 mengalami masalah kinerja, pertimbangkan untuk meningkatkan kapasitas minimum klaster. Untuk detail tentang perubahan apakah skala instance pembaca bersama dengan penulis atau secara independen, lihatMelihat dan memodifikasi tingkat promosi pembaca tanpa server.

Jika Anda memiliki cluster dengan instance pembaca tanpa server DocumentDB, pembaca tidak menskalakan bersama dengan instance penulis ketika tingkat promosi pembaca bukan 0 atau 1. Dalam hal ini, pengaturan kapasitas minimum yang rendah dapat mengakibatkan lag replikasi yang berlebihan. Hal ini karena pembaca mungkin tidak memiliki kapasitas yang cukup untuk menerapkan perubahan dari penulis ketika basis data sibuk. Kami menyarankan Anda mengatur kapasitas minimum ke nilai yang mewakili jumlah memori dan CPU yang sebanding dengan instance penulis.

- Waktu yang dibutuhkan untuk instance tanpa server DocumentDB untuk menskalakan dari kapasitas minimumnya ke kapasitas maksimumnya tergantung pada perbedaan antara nilai DCU minimum dan maksimumnya. Ketika kapasitas instans saat ini besar, DocumentDB tanpa server meningkat dalam peningkatan yang lebih besar daripada saat instance dimulai dari kapasitas kecil. Jadi, jika Anda menentukan kapasitas maksimum yang relatif besar dan instans menghabiskan sebagian besar waktunya di dekat kapasitas itu, pertimbangkan untuk meningkatkan pengaturan DCU minimum. Dengan begitu, instans idle dapat menskalakan kembali ke kapasitas maksimum dengan lebih cepat.
- Batas instans tertentu ditentukan oleh kapasitas instans tanpa server saat ini, seperti batas koneksi, batas kursor, dan batas transaksi terbuka. Jika kapasitas instans saat ini kecil, maka batasnya juga akan kecil. Jika batasan ini menjadi masalah saat instance tanpa server Anda diperkecil ke `MinCapacity` nilainya, pertimbangkan untuk meningkatkan `MinCapacity` ke nilai yang lebih tinggi. Untuk informasi selengkapnya, lihat Batas instans tanpa server Amazon DocumentDB.
- Selanjutnya, batas instance tertentu dibatasi pada nilai maksimum yang lebih rendah jika `MinCapacity` disetel ke kurang dari atau sama dengan 1,0 DCU, seperti batas koneksi aktif, batas kursor, dan batas transaksi terbuka. Jika batas yang dibatasi ini tidak cukup untuk beban kerja Anda, gunakan `MinCapacity` nilai minimal 1,5 DCU. Untuk informasi selengkapnya, lihat Batas instans tanpa server Amazon DocumentDB.

Untuk petunjuk tentang cara memodifikasi konfigurasi penskalaan klaster, lihatMengelola Amazon DocumentDB tanpa server.

Memilih **`MaxCapacity`**pengaturan untuk cluster tanpa server DocumentDB

Sangat menggoda untuk selalu memilih beberapa nilai tinggi untuk pengaturan kapasitas tanpa server DocumentDB maksimum. Kapasitas maksimum yang besar memungkinkan instans untuk meningkatkan skala paling banyak saat menjalankan beban kerja intensif. Nilai rendah menghindari kemungkinan biaya tak terduga. Bergantung pada cara Anda menggunakan klaster tersebut dan pengaturan lain yang Anda konfigurasikan, nilai yang paling efektif mungkin lebih tinggi atau lebih rendah dari yang Anda kira. Pertimbangkan faktor-faktor berikut saat memilih pengaturan kapasitas maksimum:

- Kapasitas maksimum harus minimal setinggi kapasitas minimum. Anda dapat mengatur kapasitas minimum dan maksimum agar identik. Namun, dalam hal ini kapasitas tidak pernah naik atau turun. Dengan demikian, menggunakan nilai yang identik untuk kapasitas minimum dan maksimum tidak tepat untuk situasi di luar pengujian.
- Kapasitas maksimum harus minimal 1,0 DCU dan harus paling banyak 256 DCU.
- Sebaiknya Anda memantau penskalaan dan penggunaan sumber daya instans tanpa server Anda. Jika instans tanpa server Anda sering menskalakan ke kapasitas maksimum dan mencapai batasan sumber daya (misalnya saat `DCUUtilization` metrik berada di 100,0), sebaiknya pilih nilai yang lebih tinggi. `MaxCapacity` Untuk informasi selengkapnya, lihat Pemantauan Amazon DocumentDB tanpa server.
- Jika Anda biasanya memodifikasi kelas instans dari instance yang disediakan untuk mengantisipasi beban kerja yang sangat tinggi atau rendah, Anda dapat menggunakan pengalaman tersebut untuk memperkirakan rentang kapasitas tanpa server DocumentDB yang setara. Untuk menentukan ukuran memori instans Amazon DocumentDB yang disediakan, lihat. Batas instans

  Misalnya, anggaplah Anda menggunakan kelas `db.r6g.4xlarge` instance ketika cluster Anda memiliki beban kerja yang tinggi. Kelas instance itu memiliki 128 GiB memori. Dengan demikian, Anda dapat menentukan pengaturan DCU maksimum 64 untuk menyiapkan instance tanpa server yang dapat menskalakan hingga kira-kira kapasitas yang sama. Itu karena setiap DCU sesuai dengan sekitar 2 GiB memori. Anda dapat menentukan nilai yang sedikit lebih tinggi untuk membiarkan instance meningkat lebih jauh jika `db.r6g.4xlarge` instans Anda terkadang tidak memiliki kapasitas yang cukup untuk menangani beban kerja secara efektif.
- Jika Anda memiliki batasan anggaran pada penggunaan database Anda, pilih nilai yang tetap dalam batas itu meskipun semua instance tanpa server Anda berjalan pada kapasitas maksimum sepanjang waktu. Ingatlah bahwa ketika Anda memiliki n instance tanpa server di cluster Anda, kapasitas tanpa server maksimum teoritis yang dapat dikonsumsi klaster kapan saja adalah n kali

pengaturan DCU maksimum untuk cluster. (Jumlah sebenarnya yang dikonsumsi mungkin lebih sedikit, misalnya jika beberapa pembaca diskalakan secara terpisah dari penulis.)

- Jika Anda menggunakan instance pembaca tanpa server untuk menurunkan beberapa beban kerja hanya-baca dari instance penulis, Anda mungkin dapat memilih pengaturan kapasitas maksimum yang lebih rendah. Anda melakukan ini untuk mencerminkan bahwa setiap instance pembaca tidak perlu menskalakan setinggi jika cluster hanya berisi satu instance.
- Misalkan Anda ingin mencegah penggunaan yang berlebihan karena parameter basis data yang salah dikonfigurasi atau kueri yang tidak efisien dalam aplikasi Anda. Dalam hal ini, Anda dapat menghindari penggunaan berlebihan yang tidak disengaja dengan memilih pengaturan kapasitas maksimum yang lebih rendah dari pengaturan tertinggi absolut yang dapat Anda tetapkan.
- Jika lonjakan karena aktivitas pengguna sebenarnya jarang tetapi dapat terjadi, Anda dapat mempertimbangkan situasi tersebut saat memilih pengaturan kapasitas maksimum. Jika prioritasnya adalah agar aplikasi tetap berjalan dengan performa dan skalabilitas penuh, Anda dapat menentukan pengaturan kapasitas maksimum yang lebih tinggi daripada yang Anda amati dalam penggunaan normal. Jika Anda tidak memiliki masalah dengan aplikasi yang berjalan dengan throughput yang berkurang selama lonjakan aktivitas yang sangat ekstrem, Anda dapat memilih pengaturan kapasitas maksimum yang sedikit lebih rendah. Pastikan Anda memilih pengaturan yang masih memiliki cukup memori dan sumber daya CPU untuk menjaga aplikasi tetap berjalan.
- Jika Anda mengaktifkan pengaturan di klaster yang meningkatkan penggunaan memori untuk setiap instance, pertimbangkan memori tersebut saat memutuskan nilai DCU maksimum. Pengaturan tersebut mencakup pengaturan untuk Performance Insights dan klaster global. Pastikan bahwa nilai DCU maksimum memungkinkan instans tanpa server untuk meningkatkan skala cukup untuk menangani beban kerja saat fitur tersebut digunakan. Untuk informasi tentang pemecahan masalah yang disebabkan oleh kombinasi setelan DCU maksimum rendah dan fitur Amazon DocumentDB yang memaksakan overhead memori, lihat (di bawah). Menghindari kesalahan kehabisan memori
- Secara khusus, kami merekomendasikan minimum berikut `MaxCapacity` untuk digunakan dengan fitur yang ditentukan (rekomendasi ini dapat berubah):
  - Pembuatan instans tanpa server pada klaster dengan volume data yang besar — 2 DCU (ini termasuk pembuatan instance tanpa server sebagai bagian dari pemulihan klaster.)
- Batas instans tertentu ditentukan oleh kapasitas instans saat ini, seperti batas koneksi, batas kursor, dan batas transaksi terbuka. Saat memilih `MaxCapacity` nilai untuk beban kerja Anda, pastikan untuk mengingat batasan instance ini agar tidak terhambat oleh salah satu batasan ini. Untuk informasi selengkapnya, lihat Batas instans tanpa server Amazon DocumentDB.

Untuk petunjuk tentang cara memodifikasi konfigurasi penskalaan klaster, lihatMengelola Amazon DocumentDB tanpa server.

Menghindari kesalahan kehabisan memori

Jika salah satu instance tanpa server DocumentDB Anda secara konsisten mencapai batas kapasitas maksimumnya, Amazon DocumentDB menunjukkan kondisi ini dengan menyetel instance ke status parameter yang tidak kompatibel. Sementara instance memiliki status parameter yang tidak kompatibel, beberapa operasi diblokir. Misalnya, Anda tidak dapat meningkatkan versi mesin. Untuk informasi selengkapnya tentang status instans Amazon DocumentDB, lihat Memantau status instans Amazon DocumentDB.

Biasanya, instance Anda masuk ke status ini ketika sering restart karena kesalahan kehabisan memori. Amazon DocumentDB merekam peristiwa ketika jenis restart ini terjadi. Untuk melihat peristiwa sumber daya, lihatMelihat acara Amazon DocumentDB. Penggunaan memori yang luar biasa tinggi dapat terjadi karena overhead dari mengaktifkan pengaturan seperti Performance Insights. Ini juga bisa berasal dari beban kerja yang berat pada instance Anda atau dari mengelola metadata yang terkait dengan sejumlah besar objek skema.

Jika tekanan memori menjadi lebih rendah sehingga instans tidak mencapai kapasitas maksimumnya sangat sering, Amazon DocumentDB secara otomatis mengubah status instans kembali ke tersedia.

Untuk pulih dari kondisi ini, Anda dapat mengambil beberapa atau semua tindakan berikut:

- Tingkatkan batas bawah kapasitas untuk instans tanpa server dengan mengubah nilai unit kapasitas DocumentDB minimum (DCU) untuk cluster. Tindakan ini menghindari masalah saat basis data idle menurunkan skalanya ke kapasitas dengan memori lebih sedikit daripada yang dibutuhkan untuk fitur yang diaktifkan di klaster Anda. Setelah mengubah pengaturan DCU untuk cluster, reboot instance tanpa server. Melakukan hal itu mengevaluasi apakah Amazon DocumentDB dapat mengatur ulang status kembali ke tersedia.
- Tingkatkan batas atas kapasitas instans tanpa server dengan mengubah nilai DCU maksimum untuk cluster. Tindakan ini menghindari masalah saat basis data yang sibuk tidak dapat menaikkan skalanya ke kapasitas dengan memori yang cukup untuk fitur yang diaktifkan di klaster Anda dan beban kerja basis data. Setelah mengubah pengaturan DCU untuk cluster, reboot instance tanpa server. Melakukan hal itu mengevaluasi apakah Amazon DocumentDB dapat mengatur ulang status kembali ke tersedia.
- Nonaktifkan pengaturan konfigurasi yang memerlukan overhead memori. Misalnya, Anda memiliki fitur seperti Performance Insights yang diaktifkan tetapi tidak menggunakannya. Jika demikian,

Anda bisa mematikannya. Atau Anda dapat menyesuaikan nilai kapasitas minimum dan maksimum untuk cluster yang lebih tinggi untuk memperhitungkan memori yang digunakan oleh jenis fitur tersebut. Untuk panduan tentang memilih pengaturan kapasitas minimum dan maksimum, lihat Memilih rentang kapasitas penskalaan untuk cluster tanpa server DocumentDB.

- Kurangi beban kerja pada instance. Misalnya, Anda dapat menambahkan instance pembaca ke cluster untuk menyebarkan beban dari kueri hanya-baca di lebih banyak instance.

Mengapa instance tanpa server saya tidak diperkecil?

Dalam beberapa kasus, DocumentDB tanpa server tidak mengurangi kapasitas minimum, bahkan tanpa beban pada database. Hal ini dapat terjadi karena alasan berikut:

- Performance Insights dapat meningkatkan penggunaan sumber daya dan mencegah database mengurangi kapasitas minimum. Fitur-fitur ini mencakup hal-hal berikut:
- Jika instance pembaca tidak mengurangi ke minimum dan tetap pada kapasitas yang sama atau lebih tinggi dari instance penulis, maka periksa tingkat prioritas instance pembaca. Instance pembaca tanpa server DocumentDB di tingkat 0 atau 1 disimpan pada kapasitas minimum setidaknya setinggi instance penulis. Ubah tingkat prioritas pembaca menjadi 2 atau lebih tinggi agar dinaikkan dan diturunkan skalanya secara independen dari penulis. Untuk informasi selengkapnya, lihat Penskalaan tanpa server Amazon DocumentDB.
- Beban kerja basis data yang berat dapat meningkatkan penggunaan sumber daya.
- Volume basis data yang besar dapat meningkatkan penggunaan sumber daya. Amazon DocumentDB menggunakan memori dan sumber daya CPU untuk manajemen cluster. Amazon DocumentDB membutuhkan lebih banyak CPU dan memori untuk mengelola cluster dengan volume database yang lebih besar. Jika kapasitas minimum klaster Anda kurang dari minimum yang diperlukan untuk pengelolaan klaster, klaster Anda tidak akan menurunkan kapasitas minimum.
- Kegiatan pemeliharaan latar belakang dapat secara berkala meningkatkan penggunaan sumber daya.

Jika basis data masih tidak menurunkan skalanya ke kapasitas minimum yang dikonfigurasi, hentikan dan aktifkan ulang basis data untuk mengklaim kembali fragmen memori yang mungkin telah terakumulasi dari waktu ke waktu. Menghentikan dan memulai basis data akan menghasilkan waktu henti, jadi sebaiknya lakukan ini seperlunya.

Pemantauan Amazon DocumentDB tanpa server

Untuk mempelajari lebih lanjut tentang pemantauan di Amazon DocumentDB, lihat. Memantau Amazon DocumentDB

Topik

- Kehabisan Memori: status parameter yang tidak kompatibel
- CloudWatch Metrik Amazon untuk DocumentDB tanpa server
- Memantau kinerja tanpa server DocumentDB dengan Performance Insights

Kehabisan Memori: status parameter yang tidak kompatibel

Jika salah satu instans tanpa server Anda secara konsisten mencapai batas kapasitas maksimumnya, Amazon DocumentDB menunjukkan kondisi ini dengan menyetel instance ke status parameter yang tidak kompatibel. Untuk informasi selengkapnya, lihat Menghindari kesalahan kehabisan memori.

CloudWatch Metrik Amazon untuk DocumentDB tanpa server

Untuk mempelajari lebih lanjut tentang menggunakan CloudWatch Amazon DocumentDB, lihat. Memantau Amazon DocumentDB dengan CloudWatch

Anda dapat melihat instans tanpa server CloudWatch untuk memantau kapasitas yang dikonsumsi oleh setiap instance dengan metrik. `ServerlessDatabaseCapacity` Anda juga dapat memantau semua metrik CloudWatch DocumentDB standar, seperti dan Queries. `DatabaseConnections` Untuk daftar lengkap CloudWatch metrik yang dapat Anda pantau untuk Amazon Memantau Amazon DocumentDB dengan CloudWatch DocumentDB, lihat. Metrik CloudWatch tingkat instans berikut ini penting untuk dipantau agar Anda memahami bagaimana instans tanpa server DocumentDB Anda naik dan turun. Semua metrik ini dihitung setiap detik. Dengan begitu, Anda dapat memantau status instans tanpa server Anda saat ini. Anda dapat menyetel alarm untuk memberi tahu Anda jika ada instans tanpa server yang mendekati ambang batas untuk metrik yang terkait dengan kapasitas. Anda dapat menentukan apakah pengaturan kapasitas minimum dan maksimum sudah sesuai, atau apakah Anda perlu menyesuaikannya. Anda dapat menentukan di mana upaya Anda harus difokuskan untuk mengoptimalkan efisiensi basis data Anda.

- **`ServerlessDatabaseCapacity`**— Sebagai metrik tingkat instance, ia melaporkan jumlah DCU yang diwakili oleh kapasitas instans saat ini. Sebagai metrik tingkat cluster, ini mewakili rata-rata `ServerlessDatabaseCapacity` nilai semua instance tanpa server DocumentDB di cluster.
- **`DCUUtilization.`**— Metrik ini baru di DocumentDB tanpa server. Nilai ini direpresentasikan sebagai persentase. Ini dihitung sebagai nilai `ServerlessDatabaseCapacity` metrik dibagi dengan nilai DCU maksimum cluster. Pertimbangkan pedoman berikut untuk menafsirkan metrik ini dan mengambil tindakan:
  - Jika metrik ini mendekati nilai`100.0`, instance telah ditingkatkan setinggi mungkin. Pertimbangkan untuk meningkatkan pengaturan DCU maksimum untuk cluster. Dengan begitu, instance penulis dan pembaca dapat menskalakan ke kapasitas yang lebih tinggi.
  - Misalkan beban kerja hanya-baca menyebabkan instance pembaca mendekati a `DCUUtilization` of`100.0`, sedangkan instance penulis tidak mendekati kapasitas maksimumnya. Dalam hal ini, pertimbangkan untuk menambahkan instance pembaca tambahan ke cluster. Dengan begitu, Anda dapat menyebarkan bagian read-only dari beban kerja di lebih banyak instance, mengurangi beban pada setiap instance pembaca.
  - Misalkan Anda menjalankan aplikasi produksi, sehingga performa dan skalabilitas adalah pertimbangan utamanya. Dalam hal ini, Anda dapat mengatur nilai DCU maksimum untuk cluster ke angka tinggi. Tujuan Anda adalah agar `DCUUtilization` metrik selalu di bawah 100,0. Dengan nilai DCU maksimum yang tinggi, Anda dapat yakin bahwa ada cukup ruang jika ada lonjakan tak terduga dalam aktivitas database. Anda hanya dikenai biaya untuk kapasitas basis data yang benar-benar dikonsumsi.
- **`CPUUtilization`**— Metrik ini ditafsirkan secara berbeda di DocumentDB tanpa server daripada di instance yang disediakan. Untuk DocumentDB tanpa server, nilai ini adalah persentase yang dihitung sebagai jumlah CPU yang saat ini digunakan, dibagi dengan kapasitas CPU yang tersedia di bawah nilai DCU maksimum cluster. Amazon DocumentDB memonitor nilai ini secara otomatis dan meningkatkan skala instans tanpa server Anda ketika instans secara konsisten menggunakan proporsi kapasitas CPU-nya yang tinggi.

  Jika metrik ini mendekati nilai`100.0`, instance telah mencapai kapasitas CPU maksimumnya. Pertimbangkan untuk meningkatkan pengaturan DCU maksimum untuk cluster. Jika metrik ini mendekati nilai `100.0` pada instance pembaca, pertimbangkan untuk menambahkan instance pembaca tambahan ke cluster. Dengan begitu, Anda dapat menyebarkan bagian read-only dari beban kerja yang tersebar di lebih banyak instance, mengurangi beban pada setiap instance pembaca.
- **`FreeableMemory`**— Nilai ini mewakili jumlah memori yang tidak terpakai yang tersedia ketika instance tanpa server DocumentDB diskalakan ke kapasitas maksimumnya. Untuk setiap DCU

yang kapasitas saat ini di bawah kapasitas maksimum, nilai ini meningkat sekitar 2 GiB. Dengan demikian, metrik ini tidak mendekati nol sampai instance ditingkatkan setinggi mungkin.

Jika metrik ini mendekati nilai nol, instance telah ditingkatkan sebanyak mungkin dan mendekati batas memori yang tersedia. Pertimbangkan untuk meningkatkan pengaturan DCU maksimum untuk cluster. Jika metrik ini mendekati nilai nol pada instance pembaca, pertimbangkan untuk menambahkan instance pembaca tambahan ke cluster. Dengan begitu, bagian read-only dari beban kerja dapat tersebar di lebih banyak instance, mengurangi penggunaan memori pada setiap instance pembaca.

- **`TempStorageIops`**— Jumlah IOPS dilakukan pada penyimpanan lokal yang melekat pada instance. Ini termasuk IOPS untuk baca dan tulis. Metrik ini merepresentasikan hitungan dan diukur sekali per detik. Ini adalah metrik baru untuk DocumentDB tanpa server. Lihat perinciannya di Memantau Amazon DocumentDB dengan CloudWatch.
- **`TempStorageThroughput`**— Jumlah data yang ditransfer ke dan dari penyimpanan lokal yang terkait dengan instance. Metrik ini merepresentasikan byte dan diukur sekali per detik. Ini adalah metrik baru untuk DocumentDB tanpa server. Lihat perinciannya di Memantau Amazon DocumentDB dengan CloudWatch.

Biasanya, sebagian besar peningkatan skala untuk instance tanpa server DocumentDB disebabkan oleh penggunaan memori dan aktivitas CPU. Metrik `TempStorageIops` dan `TempStorageThroughput` metrik dapat membantu Anda mendiagnosis kasus yang jarang terjadi di mana aktivitas jaringan untuk transfer antara instans Anda dan perangkat penyimpanan lokal bertanggung jawab atas peningkatan kapasitas yang tidak terduga. Untuk memantau aktivitas jaringan lainnya, Anda dapat menggunakan metrik yang ada ini:

- `NetworkReceiveThroughput`
- `NetworkThroughput`
- `NetworkTransmitThroughput`
- `StorageNetworkReceiveThroughput`
- `StorageNetworkThroughput`
- `StorageNetworkTransmitThroughput`

Bagaimana metrik tanpa CloudWatch server DocumentDB berlaku untuk Anda AWS tagihan

Biaya tanpa server DocumentDB pada tagihan AWS Anda dihitung berdasarkan metrik yang `ServerlessDatabaseCapacity` sama yang dapat Anda pantau. Mekanisme penagihan dapat berbeda dari CloudWatch rata-rata yang dihitung untuk metrik ini jika Anda menggunakan kapasitas tanpa server DocumentDB hanya untuk sebagian jam. Ini juga dapat berbeda jika masalah sistem membuat CloudWatch metrik tidak tersedia untuk periode singkat. Dengan demikian, Anda mungkin melihat nilai tagihan yang DCU-hours sedikit berbeda daripada jika Anda menghitung sendiri angkanya dari nilai `ServerlessDatabaseCapacity` rata-rata.

Contoh perintah Amazon CloudWatch CLI untuk metrik tanpa server DocumentDB

AWS CLI Contoh berikut menunjukkan bagaimana Anda dapat memantau CloudWatch metrik terpenting yang terkait dengan DocumentDB tanpa server. Dalam setiap kasus, ganti `Value=` string untuk `--dimensions` parameter dengan pengidentifikasi instance tanpa server DocumentDB Anda sendiri.

Contoh Linux berikut menampilkan nilai kapasitas minimum, maksimum, dan rata-rata untuk sebuah instance, diukur setiap 10 menit selama satu jam. Perintah tanggal Linux menentukan waktu mulai dan akhir relatif terhadap tanggal dan waktu saat ini. Fungsi `sort_by` dalam parameter `--query` mengurutkan hasil secara kronologis berdasarkan bidang `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
```

Contoh Linux berikut menunjukkan pemantauan kapasitas instance dalam sebuah cluster.
Ini mengukur pemanfaatan kapasitas minimum, maksimum, dan rata-rata suatu instans.
Pengukuran dilakukan sekali setiap jam selama periode tiga jam. Contoh-contoh ini menggunakan `DCUUtilization` metrik yang mewakili persentase batas atas pada DCU, alih-alih

`ServerlessDatabaseCapacity` mewakili jumlah tetap DCU. Dengan begitu, Anda tidak perlu mengetahui angka aktual untuk nilai DCU minimum dan maksimum dalam rentang kapasitas. Anda dapat melihat persentase mulai dari 0 hingga 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
```

Contoh Linux berikut melakukan pengukuran yang sama seperti yang sebelumnya. Dalam hal ini, pengukuran ditujukan untuk metrik `CPUUtilization`. Pengukuran dilakukan setiap 10 menit selama periode 1 jam. Angka-angka tersebut mewakili persentase CPU yang tersedia yang digunakan, berdasarkan sumber daya CPU yang tersedia untuk pengaturan kapasitas maksimum untuk instance.

```
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
```

Contoh Linux berikut melakukan pengukuran yang sama seperti yang sebelumnya. Dalam hal ini, pengukuran ditujukan untuk metrik `FreeableMemory`. Pengukuran dilakukan setiap 10 menit selama periode 1 jam.

```
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
```

Memantau kinerja tanpa server DocumentDB dengan Performance Insights

Anda dapat menggunakan Performance Insights untuk memantau kinerja instance tanpa server DocumentDB. Untuk prosedur Wawasan Performa, lihat Memantau dengan Wawasan Kinerja.

Penghitung Performance Insights baru berikut ini berlaku untuk instance tanpa server DocumentDB:

- **`os.general.serverlessDBCapacity`**- Kapasitas instans saat ini di DCU. Nilai sesuai dengan `ServerlessDatabaseCapacity` CloudWatch metrik untuk contoh.
- **`os.general.dcuUtilization`**— Persentase kapasitas saat ini dari kapasitas maksimum yang dikonfigurasi. Nilai sesuai dengan `DCUUtilization` CloudWatch metrik untuk contoh.
- **`os.general.maxConfiguredDcu`**— Kapasitas maksimum yang Anda konfigurasikan untuk instance tanpa server DocumentDB ini. Ini diukur dalam DCU.
- **`os.general.minConfiguredDcu`**— Kapasitas minimum yang Anda konfigurasikan untuk instance tanpa server DocumentDB ini. Ini diukur dalam DCU.

Untuk daftar lengkap penghitung Wawasan Performa, lihat Performance Insights untuk metrik penghitung.

Ketika nilai vCPU ditampilkan untuk instance tanpa server DocumentDB di Performance Insights, nilai tersebut mewakili perkiraan berdasarkan nilai DCU untuk instance tersebut. Pada interval default satu menit, nilai vCPU fraksional apa pun dibulatkan ke bilangan bulat terdekat. Untuk interval yang lebih lama, nilai vCPU yang ditampilkan adalah rata-rata nilai vCPU bilangan bulat untuk setiap menit.

Melaksanakan Pengembangan dengan Amazon DocumentDB

Bagian ini mencakup pengembangan menggunakan Amazon DocumentDB.

Topik

- Connect ke Amazon DocumentDB
- Program dengan Amazon DocumentDB

Connect ke Amazon DocumentDB

Amazon DocumentDB menyediakan beberapa opsi koneksi, termasuk koneksi terprogram menggunakan driver MongoDB, menghubungkan dari luar VPC melalui AWS SSH tunneling atau VPN, dan alat populer seperti Studio 3T dan. DataGrip Layanan ini mendukung koneksi set replika untuk ketersediaan tinggi. Untuk analitik dan pelaporan, Anda dapat terhubung menggunakan driver JDBC atau ODBC. Instans EC2 di VPC yang sama dapat terhubung langsung ke cluster DocumentDB. Semua koneksi membutuhkan TLS dan tepat AWS credentials/authentication.

Topik

- Menghubungkan secara terprogram ke Amazon DocumentDB
- Menghubungkan ke cluster Amazon DocumentDB dari luar Amazon VPC
- Menghubungkan ke Amazon DocumentDB sebagai set replika
- Menghubungkan ke cluster Amazon DocumentDB dari Studio 3T
- Connect ke Amazon DocumentDB menggunakan DataGrip
- Connect menggunakan Amazon EC2
- Connect menggunakan driver Amazon DocumentDB JDBC
- Connect menggunakan driver Amazon DocumentDB ODBC

Menghubungkan secara terprogram ke Amazon DocumentDB

Bagian ini berisi contoh kode yang menunjukkan cara menghubungkan ke Amazon DocumentDB (dengan kompatibilitas MongoDB) menggunakan beberapa bahasa yang berbeda. Contoh dipisahkan menjadi dua bagian berdasarkan apakah Anda terhubung ke klaster yang memiliki Keamanan

Lapisan Pengangkutan (TLS) diaktifkan atau dinonaktifkan. Secara default, TLS diaktifkan di klaster Amazon DocumentDB. Namun, Anda dapat menonaktifkan TLS jika Anda mau. Untuk informasi selengkapnya, lihat Mengenkripsi data dalam perjalanan.

Jika Anda mencoba terhubung ke Amazon DocumentDB dari luar VPC tempat klaster Anda berada, lihat. Menghubungkan ke cluster Amazon DocumentDB dari luar Amazon VPC

Sebelum Anda terhubung ke klaster Anda, Anda harus tahu apakah TLS diaktifkan di klaster. Bagian berikutnya menunjukkan kepada Anda bagaimana untuk menentukan nilai parameter `tls` klaster Anda menggunakan Konsol Manajemen AWS atau AWS CLI. Setelah itu, Anda dapat melanjutkan dengan menemukan dan menerapkan contoh kode yang sesuai.

Topik

- Menentukan nilai parameter tls Anda
- Menghubungkan dengan TLS diaktifkan
- Menghubungkan dengan TLS dinonaktifkan

Menentukan nilai parameter **`tls`** Anda

Menentukan apakah klaster Anda mengaktifkan TLS adalah proses dua langkah yang dapat Anda lakukan menggunakan atau. Konsol Manajemen AWS AWS CLI

1. Tentukan grup parameter mana yang mengatur cluster Anda.

   Using the Konsol Manajemen AWS

   1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
   2. Pada panel navigasi sebelah kiri, pilih Klaster.
   3. Dalam daftar klaster, pilih nama klaster Anda.
   4. Halaman yang dihasilkan menunjukkan detail klaster yang Anda pilih. Pilih tab Konfigurasi. Di bagian Konfigurasi dan status, cari nama grup parameter di bawah grup parameter Cluster.

Using the AWS CLI

AWS CLI Kode berikut menentukan parameter mana yang mengatur cluster Anda. Pastikan Anda mengganti *sample-cluster* dengan nama klaster Anda.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].[DBClusterIdentifier,DBClusterParameterGroup]'
```

Keluaran dari operasi ini terlihat seperti berikut ini:

```
[
       [
           "sample-cluster",
           "sample-parameter-group"
       ]
]
```

2. Tentukan nilai **tls** parameter dalam grup parameter cluster Anda.

Using the Konsol Manajemen AWS

1. Di panel navigasi, pilih Grup parameter.
2. Di jendela grup parameter Cluster, pilih nama grup parameter cluster Anda dari Langkah 1d.
3. Halaman yang dihasilkan menunjukkan parameter grup parameter klaster Anda. Anda dapat melihat nilai dari parameter `tls` di sini. Untuk informasi tentang memodifikasi parameter ini, lihat Memodifikasi grup parameter cluster Amazon DocumentDB.

Using the AWS CLI

Anda dapat menggunakan `describe-db-cluster-parameters` AWS CLI perintah untuk melihat detail parameter dalam grup parameter cluster Anda.

- **`--describe-db-cluster-parameters`** — Untuk membuat daftar semua parameter di dalam grup parameter dan nilainya.
  - **`--db-cluster-parameter-group name`** — Diperlukan. Nama grup parameter klaster Anda.

Dalam contoh berikut, ganti masing-masing *user input placeholder* dengan informasi cluster Anda.

```
aws docdb describe-db-cluster-parameters \
    --db-cluster-parameter-group-name sample-parameter-group
```

Keluaran dari operasi ini terlihat seperti berikut ini:

```
{
        "Parameters": [
            {
                "ParameterName": "profiler_threshold_ms",
                "ParameterValue": "100",
                "Description": "Operations longer than profiler_threshold_ms
 will be logged",
                "Source": "system",
                "ApplyType": "dynamic",
                "DataType": "integer",
                "AllowedValues": "50-2147483646",
                "IsModifiable": true,
                "ApplyMethod": "pending-reboot"
            },
            {
                "ParameterName": "tls",
                "ParameterValue": "disabled",
                "Description": "Config to enable/disable TLS",
                "Source": "user",
                "ApplyType": "static",
                "DataType": "string",
                "AllowedValues": "disabled,enabled,fips-140-3",
                "IsModifiable": true,
                "ApplyMethod": "pending-reboot"
            }
        ]
}
```

> **Note**
>
> Amazon DocumentDB mendukung FIPS 140-3 endpoint dimulai dengan Amazon DocumentDB 5.0 (versi engine 3.0.3727) cluster di wilayah ini: ca-central-1, us-west-2, us-east-1, us-east-1, us-east-2, us-east-2, us-gov-east-1, us-gov-west-1.

Setelah menentukan nilai dari parameter `tls` Anda, lanjutkan menghubungkan ke klaster Anda dengan menggunakan salah satu contoh kode di bagian berikut.

- Menghubungkan dengan TLS diaktifkan
- Menghubungkan dengan TLS dinonaktifkan

Menghubungkan dengan TLS diaktifkan

Untuk melihat contoh kode untuk menghubungkan secara terprogram ke klaster Amazon TLS-enabled DocumentDB, pilih tab yang sesuai untuk bahasa yang ingin Anda gunakan.

Untuk mengenkripsi data dalam perjalanan, unduh kunci publik untuk Amazon `global-bundle.pem` DocumentDB bernama menggunakan operasi berikut.

```
wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
```

Jika aplikasi Anda berada di Microsoft Windows dan memerlukan file PKCS7, Anda dapat mengunduh paket sertifikat PKCS7. Bundel ini berisi sertifikat perantara dan root di https://truststore.pki.rds.amazonaws.com/global/global-bundle.p7b.

Python

Kode berikut menunjukkan cara menghubungkan ke Amazon DocumentDB menggunakan Python ketika TLS diaktifkan.

Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi cluster Anda.

```
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

Kode berikut menunjukkan cara terhubung ke Amazon Node.js DocumentDB menggunakan saat TLS diaktifkan.

> **Important**
>
> Ada batasan yang diketahui dengan Node.js driver yang lebih lama dari versi 6.13.1, yang saat ini tidak didukung oleh otentikasi identitas IAM untuk Amazon DocumentDB. Node.js driver dan alat yang menggunakan Node.js driver (misalnya, mongosh) harus ditingkatkan untuk menggunakan Node.js driver versi 6.13.1 atau lebih tinggi.

Dalam contoh berikut, ganti masing-masing *user input placeholder* dengan informasi cluster Anda.

```
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

Kode berikut menunjukkan cara menghubungkan ke Amazon DocumentDB menggunakan PHP ketika TLS diaktifkan.

Dalam contoh berikut, ganti masing-masing *user input placeholder* dengan informasi cluster Anda.

```
<?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

Kode berikut menunjukkan cara menghubungkan ke Amazon DocumentDB menggunakan Go ketika TLS diaktifkan.

> **Note**
>
> Pada versi 1.2.1, Driver MongoDB Go hanya akan menggunakan sertifikat server CA pertama yang ditemukan di `sslcertificateauthorityfile`. Contoh kode di bawah ini mengatasi batasan ini dengan menambahkan semua sertifikat server secara manual yang ditemukan di `sslcertificateauthorityfile` ke konfigurasi TLS kustom yang digunakan selama pembuatan klien.

Dalam contoh berikut, ganti masing-masing *`user input placeholder`* dengan informasi cluster Anda.

```
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

Saat menghubungkan ke klaster TLS-enabled Amazon DocumentDB dari aplikasi Java, program Anda harus menggunakan AWS file otoritas sertifikat (CA) yang disediakan untuk memvalidasi koneksi. Untuk menggunakan sertifikat CA Amazon RDS, lakukan hal berikut:

1. Unduh file CA Amazon RDS dari https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem .

2. Buat penyimpanan kepercayaan dengan sertifikat CA yang terdapat dalam file dengan melakukan perintah berikut. Pastikan untuk mengubah *truststore-password* ke sesuatu yang lain. Jika Anda mengakses penyimpanan kepercayaan yang berisi sertifikat CA lama (`rds-ca-2015-root.pem`) dan sertifikat CA baru (`rds-ca-2019-root.pem`), Anda dapat mengimpor paket sertifikat ke dalam penyimpanan kepercayaan.

   Berikut adalah contoh skrip shell yang mengimpor paket sertifikat ke trust store di sistem operasi Linux. Dalam contoh berikut, ganti masing-masing *user input placeholder* dengan informasi Anda sendiri. Terutama, di mana pun direktori contoh "*mydir*" berada di skrip, gantilah dengan direktori yang Anda buat untuk tugas ini.

```
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
```

Berikut adalah contoh skrip shell yang mengimpor paket sertifikat ke trust store di sistem operasi 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. Gunakan `keystore` di program Anda dengan mengatur properti sistem berikut di aplikasi Anda sebelum membuat koneksi ke klaster Amazon DocumentDB.

```
javax.net.ssl.trustStore: truststore
javax.net.ssl.trustStorePassword: truststore-password;
```

4. Kode berikut menunjukkan cara menghubungkan ke Amazon DocumentDB menggunakan Java ketika TLS diaktifkan.

   Dalam contoh berikut, ganti masing-masing *user input placeholder* dengan informasi cluster Anda.

```
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

Kode berikut menunjukkan cara menghubungkan ke Amazon DocumentDB menggunakan C# / .NET ketika TLS diaktifkan.

Dalam contoh berikut, ganti masing-masing *user input placeholder* dengan informasi cluster Anda.

```
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

Kode berikut menunjukkan cara menghubungkan ke dan kueri Amazon DocumentDB menggunakan versi terbaru, mongosh, atau versi shell mongo sebelumnya, saat TLS diaktifkan.

Connect ke Amazon DocumentDB dengan mongosh

> **Important**
>
> Ada batasan yang diketahui dengan Node.js driver yang lebih lama dari versi 6.13.1, yang saat ini tidak didukung oleh otentikasi identitas IAM untuk Amazon DocumentDB. Node.js driver dan alat yang menggunakan Node.js driver (misalnya, mongosh) harus ditingkatkan untuk menggunakan Node.js driver versi 6.13.1 atau lebih tinggi.

Dalam contoh berikut, ganti masing-masing *user input placeholder* dengan informasi cluster Anda.

```
mongosh --tls --host cluster-end-point:27017 --tlsCAFile  global-bundle.pem --
username sample-user --password password --retryWrites false
```

Connect ke Amazon DocumentDB dengan versi shell mongo sebelumnya

Jika Anda menggunakan IAM, Anda harus menggunakan versi mongo shell sebelumnya. Masukkan salah satu opsi perintah berikut:

```
mongo --ssl --host cluster-end-point:27017 --sslCAFile global-bundle.pem --
username sample-user --password password
```

Jika Anda menggunakan versi yang sama dengan atau lebih besar dari 4.2, gunakan kode berikut untuk menghubungkan. Tulisan yang dapat dicoba ulang tidak didukung di Amazon DocumentDB. Jika Anda menggunakan shell mongo lama (bukan mongosh), jangan sertakan `retryWrites=false` perintah dalam string kode apa pun. Secara default, penulisan yang dapat dicoba ulang dinonaktifkan. Termasuk `retryWrites=false` dapat menyebabkan kegagalan dalam perintah baca normal.

```
mongo --tls --host cluster-end-point:27017 --tlsCAFile global-bundle.pem --
username sample-user --password password
```

Uji koneksi

1. Masukkan dokumen tunggal.

   ```
   db.myTestCollection.insertOne({'hello':'Amazon DocumentDB'})
   ```

2. Temukan dokumen yang sebelumnya dimasukkan.

```
db.myTestCollection.find({'hello':'Amazon DocumentDB'})
```

R

Kode berikut menunjukkan cara menghubungkan ke Amazon DocumentDB dengan R menggunakan mongolite (https://jeroen.github.io/mongolite/) ketika TLS diaktifkan.

Dalam contoh berikut, ganti masing-masing *user input placeholder* dengan informasi cluster Anda.

```
#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

Kode berikut menunjukkan cara menghubungkan ke Amazon DocumentDB dengan Ruby ketika TLS diaktifkan.

Dalam contoh berikut, ganti masing-masing *user input placeholder* dengan informasi cluster Anda.

```
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
```

Menghubungkan dengan TLS dinonaktifkan

Untuk melihat contoh kode untuk menghubungkan secara terprogram ke klaster Amazon TLS-disabled DocumentDB, pilih tab untuk bahasa yang ingin Anda gunakan.

Python

Kode berikut menunjukkan cara menghubungkan ke Amazon DocumentDB menggunakan Python ketika TLS dinonaktifkan.

Dalam contoh berikut, ganti masing-masing *user input placeholder* dengan informasi cluster Anda.

```
## 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

Kode berikut menunjukkan cara terhubung ke Amazon Node.js DocumentDB menggunakan saat TLS dinonaktifkan.

> **Important**
>
> Ada batasan yang diketahui dengan Node.js driver yang lebih lama dari versi 6.13.1, yang saat ini tidak didukung oleh otentikasi identitas IAM untuk Amazon DocumentDB. Node.js driver dan alat yang menggunakan Node.js driver (misalnya, mongosh) harus ditingkatkan untuk menggunakan Node.js driver versi 6.13.1 atau lebih tinggi.

Dalam contoh berikut, ganti masing-masing *user input placeholder* dengan informasi cluster Anda.

```
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

Kode berikut menunjukkan cara menghubungkan ke Amazon DocumentDB menggunakan PHP ketika TLS dinonaktifkan.

Dalam contoh berikut, ganti masing-masing *user input placeholder* dengan informasi cluster Anda.

```
<?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

Kode berikut menunjukkan cara menghubungkan ke Amazon DocumentDB menggunakan Go ketika TLS dinonaktifkan.

Dalam contoh berikut, ganti masing-masing *user input placeholder* dengan informasi cluster Anda.

```
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

Kode berikut menunjukkan cara menghubungkan ke Amazon DocumentDB menggunakan Java ketika TLS dinonaktifkan.

Dalam contoh berikut, ganti masing-masing *user input placeholder* dengan informasi cluster Anda.

```
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

Kode berikut menunjukkan cara menghubungkan ke Amazon DocumentDB menggunakan C# / .NET ketika TLS dinonaktifkan.

Dalam contoh berikut, ganti masing-masing *user input placeholder* dengan informasi cluster Anda.

```
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

Kode berikut menunjukkan cara menghubungkan ke dan kueri Amazon DocumentDB menggunakan versi terbaru, mongosh, atau versi shell mongo sebelumnya, ketika TLS dinonaktifkan.

Connect ke Amazon DocumentDB dengan mongosh

> **Important**
>
> Ada batasan yang diketahui dengan Node.js driver yang lebih lama dari versi 6.13.1, yang saat ini tidak didukung oleh otentikasi identitas IAM untuk Amazon DocumentDB. Node.js driver dan alat yang menggunakan Node.js driver (misalnya, mongosh) harus ditingkatkan untuk menggunakan Node.js driver versi 6.13.1 atau lebih tinggi.

Dalam contoh berikut, ganti masing-masing *user input placeholder* dengan informasi cluster Anda.

```
mongosh --host cluster-end-point:27017 --username sample-user --password password --
retryWrites false
```

Connect ke Amazon DocumentDB dengan versi shell mongo sebelumnya

Jika Anda menggunakan IAM, Anda harus menggunakan versi mongo shell sebelumnya. Masukkan salah satu opsi perintah berikut:

```
mongo --host cluster-end-point:27017 --username sample-user --password password
```

Jika Anda menggunakan versi yang sama dengan atau lebih besar dari 4.2, gunakan kode berikut untuk menghubungkan. Tulisan yang dapat dicoba ulang tidak didukung di Amazon DocumentDB. Jika Anda menggunakan shell mongo lama (bukan mongosh), jangan sertakan `retryWrites=false` perintah dalam string kode apa pun. Secara default, penulisan yang dapat dicoba ulang dinonaktifkan. Termasuk `retryWrites=false` dapat menyebabkan kegagalan dalam perintah baca normal.

```
mongo --host cluster-end-point:27017 --username sample-user --password password
```

Uji koneksi

1. Masukkan dokumen tunggal.

   ```
   db.myTestCollection.insertOne({'hello':'Amazon DocumentDB'})
   ```

2. Temukan dokumen yang sebelumnya dimasukkan.

   ```
   db.myTestCollection.find({'hello':'Amazon DocumentDB'})
   ```

R

Kode berikut menunjukkan cara menghubungkan ke Amazon DocumentDB dengan R menggunakan mongolite (https://jeroen.github.io/mongolite/) ketika TLS dinonaktifkan.

Dalam contoh berikut, ganti masing-masing *user input placeholder* dengan informasi cluster Anda.

```
#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

Kode berikut menunjukkan cara menghubungkan ke Amazon DocumentDB dengan Ruby ketika TLS dinonaktifkan.

Dalam contoh berikut, ganti masing-masing *user input placeholder* dengan informasi cluster Anda.

```
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
```

Menghubungkan ke cluster Amazon DocumentDB dari luar Amazon VPC

Klaster Amazon DocumentDB (dengan kompatibilitas MongoDB) di-deploy dalam Amazon Virtual Private Cloud (Amazon VPC). Mereka dapat diakses langsung oleh instans Amazon EC2 atau AWS layanan lain yang digunakan di VPC Amazon yang sama. Selain itu, Amazon DocumentDB dapat diakses oleh instans EC2 atau layanan AWS lain di VPC yang berbeda di Wilayah yang sama Wilayah AWS atau lainnya melalui peering VPC.

Namun, anggaplah kasus penggunaan Anda mengharuskan Anda (atau aplikasi Anda) mengakses sumber daya Amazon DocumentDB Anda dari luar VPC klaster. Dalam hal ini, Anda dapat menggunakan tunneling SSH (juga dikenal sebagai penerusan port) untuk mengakses sumber daya Amazon DocumentDB Anda.

Hal ini berada di luar cakupan topik ini untuk membahas tunneling SSH secara mendalam. Untuk informasi lebih lanjut tentang tunneling SSH, lihat hal berikut:

- Terowongan SSH
- Contoh Penerusan Port SSH, khususnya bagian Penerusan Lokal

Untuk membuat terowongan SSH, Anda memerlukan instans Amazon EC2 yang berjalan di Amazon VPC yang sama dengan klaster Amazon DocumentDB Anda. Anda dapat menggunakan instans EC2 yang ada di VPC yang sama sebagai klaster Anda atau membuatnya. Untuk informasi lebih lanjut, lihat topik yang sesuai untuk sistem operasi Anda:

- Memulai Instans Linux Amazon EC2
- Memulai Instans Windows Amazon EC2

Biasanya Anda terhubung ke instans EC2 dengan menggunakan perintah berikut.

```
ssh -i "ec2Access.pem" ubuntu@ec2-34-229-221-164.compute-1.amazonaws.com
```

Jika demikian, Anda dapat mengatur terowongan SSH ke klaster Amazon DocumentDB `sample-cluster.node.us-east-1.docdb.amazonaws.com` dengan menjalankan perintah berikut di komputer lokal Anda. Bendera `-L` digunakan untuk meneruskan port lokal. Saat menggunakan terowongan SSH, kami rekomendasikan Anda menyambung ke klaster Anda menggunakan titik akhir klaster dan jangan mencoba untuk menyambung dalam mode set replika (yaitu, menentukan `replicaSet=rs0` dalam string koneksi Anda) karena akan menghasilkan kesalahan.

```
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
```

Setelah terowongan SSH dibuat, perintah apa pun yang Anda keluarkan ke `localhost:27017` diteruskan ke klaster Amazon DocumentDB `sample-cluster` yang berjalan di Amazon VPC. Jika Keamanan Lapisan Pengangkutan (TLS) diaktifkan pada klaster Amazon DocumentDB Anda, Anda perlu mengunduh kunci publik untuk Amazon DocumentDB dari https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem . Operasi berikut mengunduh file ini:

```
wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
```

> **Note**
>
> TLS diaktifkan secara default untuk klaster Amazon DocumentDB baru. Namun, Anda dapat menonaktifkannya. Untuk informasi selengkapnya, lihat Mengelola pengaturan TLS cluster Amazon DocumentDB.

Untuk terhubung ke klaster Amazon DocumentDB Anda dari luar Amazon VPC, gunakan perintah berikut.

```
mongo --sslAllowInvalidHostnames --ssl --sslCAFile global-bundle.pem --username
 <yourUsername> --password <yourPassword>
```

Menghubungkan ke Amazon DocumentDB sebagai set replika

Saat Anda mengembangkan pada Amazon DocumentDB (dengan kompatibilitas MongoDB), Anda sebaiknya membuat sambungan ke klaster Anda sebagai set replika dan mendistribusikan pembacaan ke instans replika menggunakan kemampuan preferensi baca bawaan driver Anda.

Bagian ini membahas lebih dalam apa artinya dan menjelaskan bagaimana Anda dapat terhubung ke klaster Amazon DocumentDB Anda sebagai set replika menggunakan SDK for Python sebagai contoh.

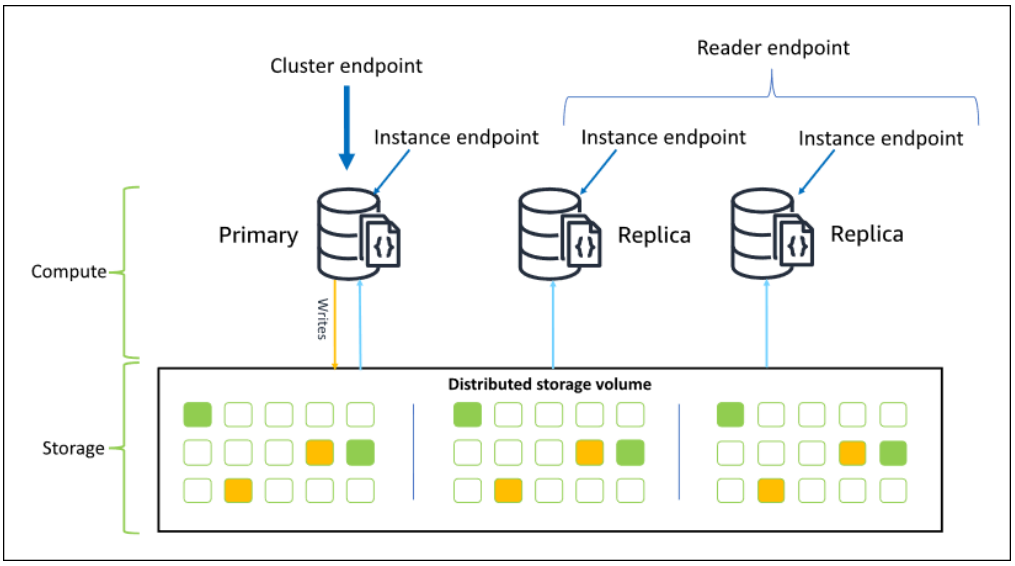
Amazon DocumentDB memiliki tiga titik akhir yang dapat Anda gunakan untuk terhubung ke klaster Anda:

- Titik akhir klaster
- Reader endpoint
- Endpoint instans

Dalam kebanyakan kasus saat Anda menghubungkan ke Amazon DocumentDB, kami merekomendasikan Anda menggunakan titik akhir klaster. Ini adalah CNAME yang menunjuk ke instans primer di klaster Anda, seperti yang ditunjukkan pada diagram berikut.

Saat menggunakan terowongan SSH, kami rekomendasikan Anda menyambung ke klaster Anda menggunakan titik akhir klaster dan jangan mencoba untuk menyambung dalam mode set replika (yaitu, menentukan `replicaSet=rs0` dalam string koneksi Anda) karena akan menghasilkan kesalahan.

> **Note**
>
> Untuk informasi selengkapnya tentang titik akhir Amazon DocumentDB, lihat Titik akhir Amazon DocumentDB.



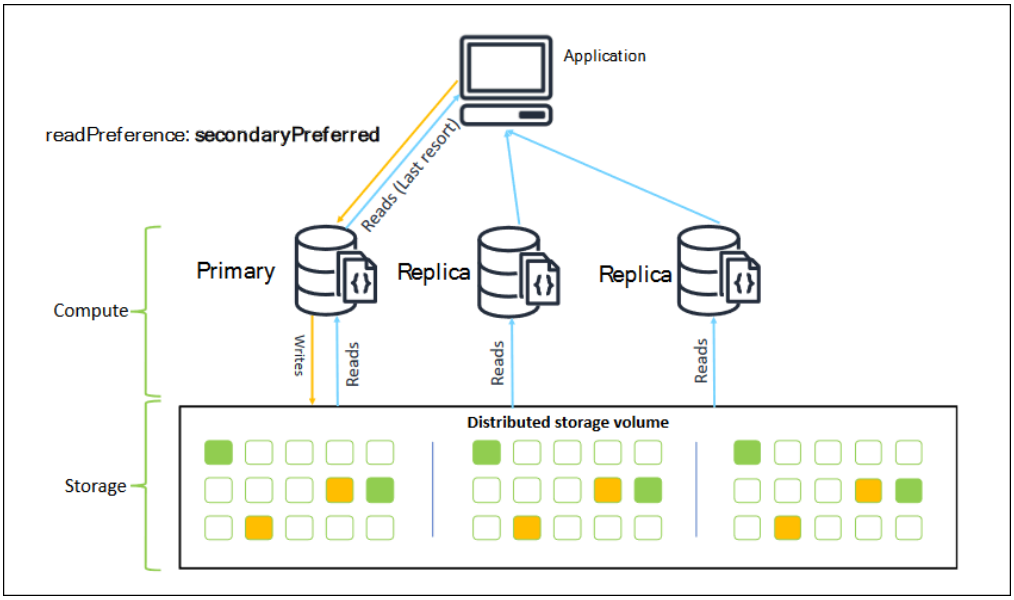
Dengan menggunakan titik akhir klaster, Anda dapat terhubung ke klaster Anda dalam mode set replika. Anda kemudian dapat menggunakan kemampuan driver preferensi baca bawaan. Dalam contoh berikut, menentukan `/?replicaSet=rs0` menandakan untuk SDK yang ingin Anda hubungkan sebagai set replika. Jika Anda menghilangkan `/?replicaSet=rs0'`, klien merutekan semua permintaan ke titik akhir klaster, yaitu instans primer Anda.

```
## 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')
```

Keuntungan menghubungkan sebagai set replika adalah memungkinkan SDK Anda menemukan topografi klaster secara otomatis, termasuk saat instans ditambahkan atau dihapus dari klaster. Anda kemudian dapat menggunakan klaster Anda secara lebih efisien dengan merutekan permintaan baca ke instans replika Anda.

Ketika Anda terhubung sebagai set replika, Anda dapat menentukan `readPreference` untuk koneksi tersebut. Jika Anda menentukan preferensi baca `secondaryPreferred`, klien merutekan kueri baca ke replika Anda dan menulis kueri ke instans primer Anda (seperti dalam diagram berikut). Ini adalah penggunaan yang lebih baik dari sumber daya klaster Anda. Untuk informasi selengkapnya, lihat Baca opsi preferensi.

```
## 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')
```



Pembacaan dari replika Amazon DocumentDB pada akhirnya adalah konsisten. Mereka mengembalikan data dalam urutan yang sama seperti yang tertulis di primer, dan sering kali ada jeda replikasi kurang dari 50 ms. Anda dapat memantau kelambatan replika untuk klaster Anda menggunakan CloudWatch metrik `DBInstanceReplicaLag` Amazon dan. `DBClusterReplicaLagMaximum` Untuk informasi selengkapnya, lihat Memantau Amazon DocumentDB dengan CloudWatch.

Tidak seperti arsitektur basis data monolitik tradisional, Amazon DocumentDB memisahkan penyimpanan dan komputasi. Dengan arsitektur modern ini, kami mendorong Anda untuk membaca skala pada instans replika. Membaca pada instans replika tidak memblokir penulisan yang direplikasi dari instans primer. Anda dapat menambahkan hingga 15 instans replika baca dalam klaster dan menskalakan keluar hingga jutaan pembacaan per detik.

Manfaat utama menghubungkan sebagai set replika dan mendistribusikan pembacaan replika adalah bahwa hal itu meningkatkan sumber daya secara keseluruhan dalam klaster Anda yang tersedia untuk melakukan pekerjaan bagi aplikasi Anda. Sebaiknya hubungkan sebagai set replika sebagai praktik terbaik. Lebih lanjut, kami merekomendasikannya paling sering dalam skenario berikut:

- Anda menggunakan CPU hampir 100 persen pada primer Anda.
- Rasio hit buffer cache mendekati nol.
- Anda mencapai batas koneksi atau kursor untuk masing-masing instans.

Menskalakan ke atas ukuran instans klaster adalah sebuah opsi, dan dalam beberapa kasus, itu bisa menjadi cara terbaik untuk menskalakan klaster. Tetapi Anda juga harus mempertimbangkan cara

menggunakan replika yang sudah Anda miliki di klaster dengan lebih baik. Hal ini memungkinkan Anda meningkatkan skala tanpa peningkatan biaya dari penggunaan tipe instans yang lebih besar. Kami juga menyarankan Anda memantau dan memperingatkan batas-batas ini (yaitu `CPUUtilization`, `DatabaseConnections`, dan `BufferCacheHitRatio`) menggunakan CloudWatch alarm sehingga Anda tahu kapan sumber daya sedang banyak digunakan.

Untuk informasi selengkapnya, lihat topik berikut:

- Praktik terbaik untuk Amazon DocumentDB
- Kuota dan batas Amazon DocumentDB

Menggunakan koneksi cluster

Pertimbangkan skenario menggunakan semua koneksi di klaster Anda. Sebagai contoh, sebuah instans `r5.2xlarge` memiliki batas 4.500 koneksi (dan 450 kursor terbuka). Jika Anda membuat tiga instans klaster Amazon DocumentDB dan terhubung hanya ke instans primer menggunakan titik akhir klaster, batas klaster Anda untuk koneksi terbuka dan kursor masing-masing adalah 4.500 dan 450. Anda mungkin mencapai batas ini jika Anda membangun aplikasi yang menggunakan banyak pekerja yang berputar dalam kontainer. Kontainer membuka sejumlah koneksi sekaligus dan memenuhi klaster.

Sebagai gantinya, Anda dapat terhubung ke klaster Amazon DocumentDB sebagai set replika dan mendistribusikan pembacaan Anda ke instans replika. Anda kemudian dapat secara efektif melipatgandakan jumlah koneksi dan kursor yang tersedia di klaster menjadi 13.500 dan 1.350 masing-masing. Menambahkan lebih banyak instans ke klaster hanya meningkatkan jumlah koneksi dan kursor untuk beban kerja baca. Jika Anda perlu meningkatkan jumlah koneksi untuk penulisan ke klaster Anda, sebaiknya tingkatkan ukuran instans.

> **Note**
>
> Jumlah koneksi untuk instans `large`, `xlarge`, dan `2xlarge` meningkat dengan ukuran instans hingga 4.500. Jumlah maksimum koneksi per instans untuk instans `4xlarge` atau lebih besar adalah 4.500. Untuk informasi selengkapnya tentang batasan menurut tipe instans, lihat Batas instans.

Biasanya kami tidak menyarankan Anda untuk terhubung ke klaster Anda menggunakan preferensi baca `secondary`. Hal ini karena jika tidak ada instans replika di klaster Anda, pembacaan akan

gagal. Misalnya, anggaplah bahwa Anda memiliki dua instans klaster Amazon DocumentDB dengan satu satu primer dan satu replika. Jika replika mengalami masalah, permintaan baca dari kolam koneksi yang ditetapkan sebagai `secondary` gagal. Keuntungan dari `secondaryPreferred` adalah bahwa jika klien tidak dapat menemukan instans replika yang cocok untuk terhubung ke, hal itu akan mengembalikan ke primer untuk pembacaan.

Beberapa kolam koneksi

Dalam beberapa skenario, pembacaan dalam aplikasi harus memiliki konsistensi baca-setelah-tulis, yang hanya dapat disajikan dari instans utama di Amazon DocumentDB. Dalam skenario ini, Anda dapat membuat dua kolam koneksi klien: satu untuk menulis dan satu untuk membaca yang memerlukan konsistensi baca-setelah-tulis. Untuk melakukan itu, kode Anda akan terlihat seperti berikut ini.

```
## 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')
```

Opsi lainnya adalah membuat kolam koneksi tunggal dan menimpa preferensi baca untuk koleksi tertentu.

```
##Specify the collection and set the read preference level for that collection
col = db.review.with_options(read_preference=ReadPreference.SECONDARY_PREFERRED)
```

Ringkasan

Untuk menggunakan sumber daya di klaster Anda dengan lebih baik, sebaiknya hubungkan ke klaster Anda menggunakan mode set replika. Jika cocok untuk aplikasi Anda, Anda dapat membaca skala aplikasi Anda dengan mendistribusikan pembacaan Anda ke instans replika.

Menghubungkan ke cluster Amazon DocumentDB dari Studio 3T

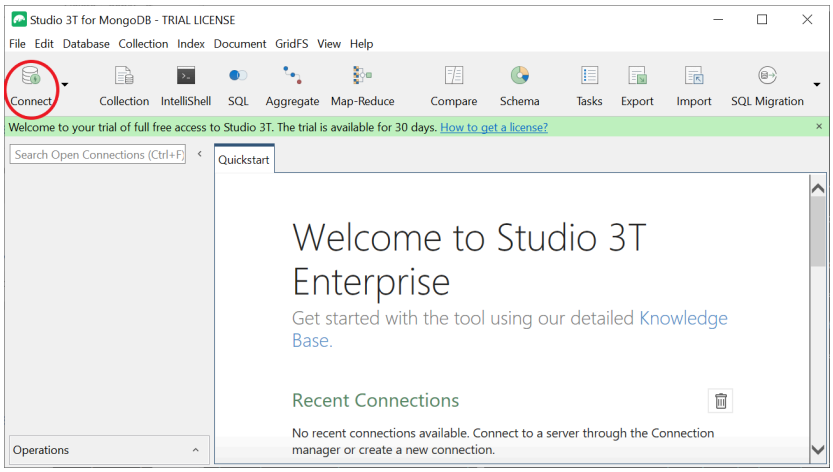
Studio 3T merupakan GUI dan IDE yang populer untuk developer dan teknisi data yang bekerja dengan MongoDB. Ini menawarkan beberapa kemampuan yang kuat Tampilan Tree, Table dan JSON dari data Anda, mudah import/export di CSV, JSON, SQL dan, opsi kueri fleksibel BSON/ mongodump, UI drag-and-drop visual, shell mongo bawaan dengan pelengkapan otomatis, editor pipeline agregasi, dan dukungan kueri SQL.

Prasyarat

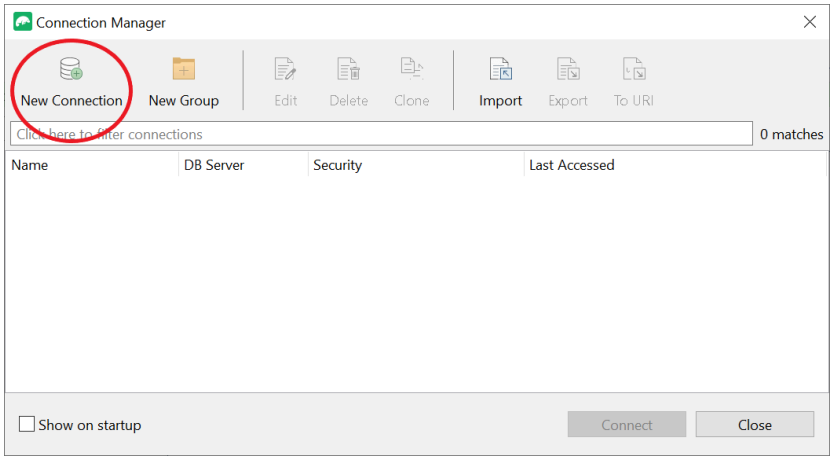
- Jika Anda belum memiliki cluster Amazon DocumentDB yang menggunakan Amazon EC2 sebagai bastion/jump host, ikuti petunjuk tentang cara Connect dengan Amazon EC2.
- Jika Anda tidak memiliki Studio 3T, unduh dan instal.

Connect dengan Studio 3T

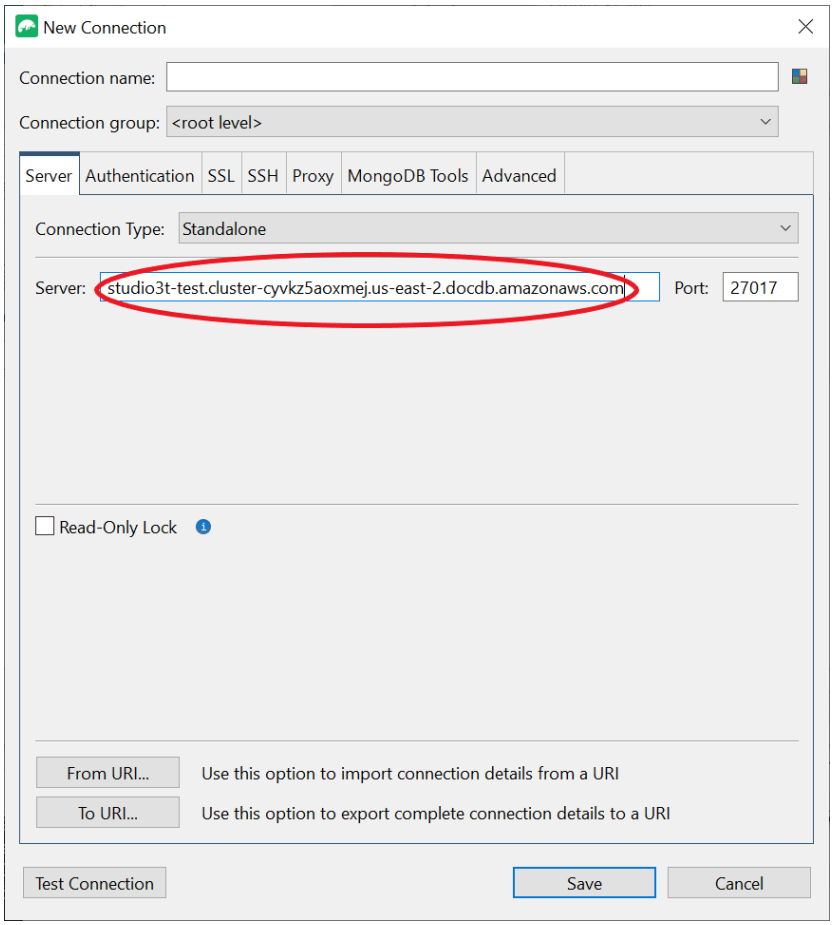
1. Pilih Connect di pojok kiri atas toolbar.



2. Pilih Koneksi Baru di pojok kiri atas toolbar.

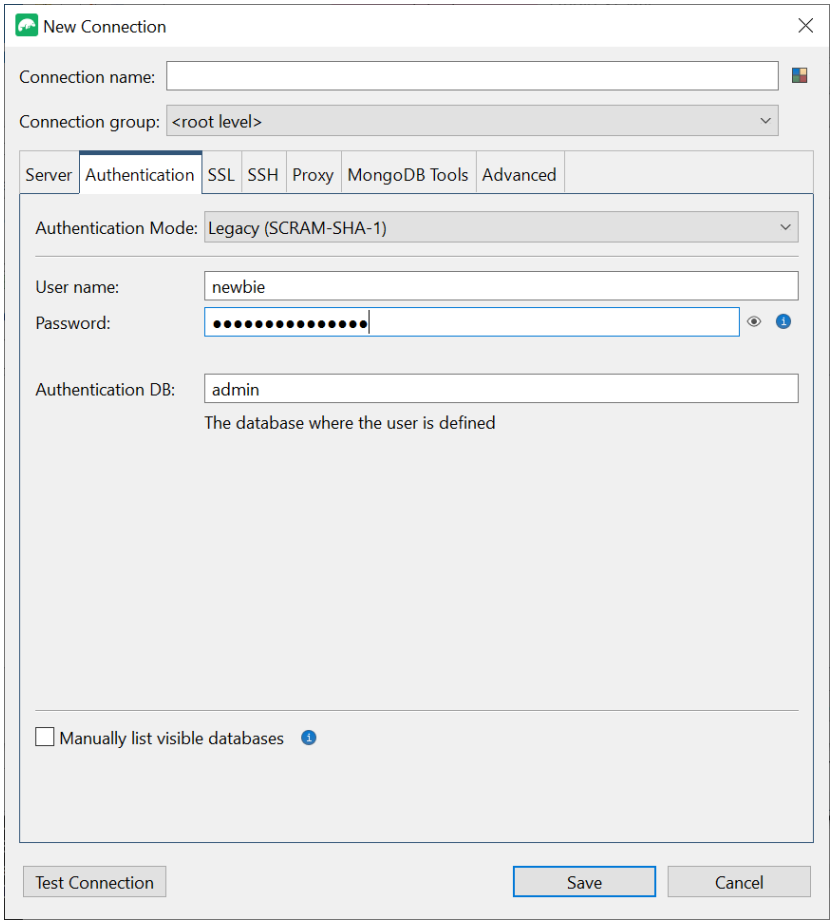


3. Pada tab Server, di bidang Server, masukkan informasi titik akhir klaster.

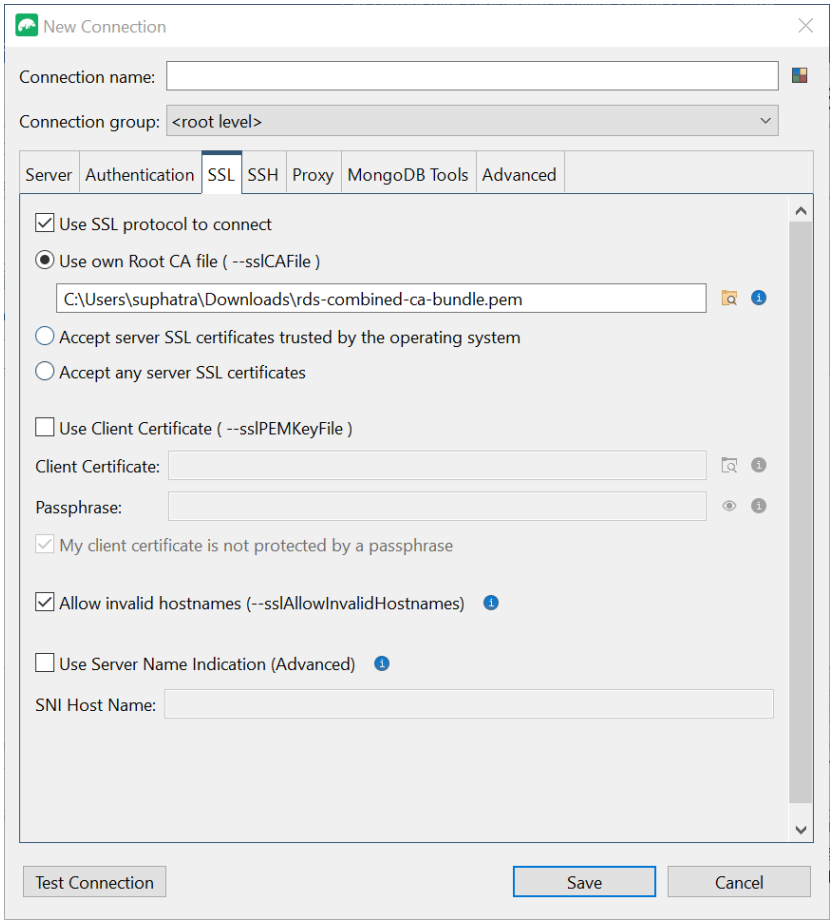


> **Note**
>
> Tidak dapat menemukan titik akhir klaster Anda? Ikuti saja langkah-langkahnya di sini.

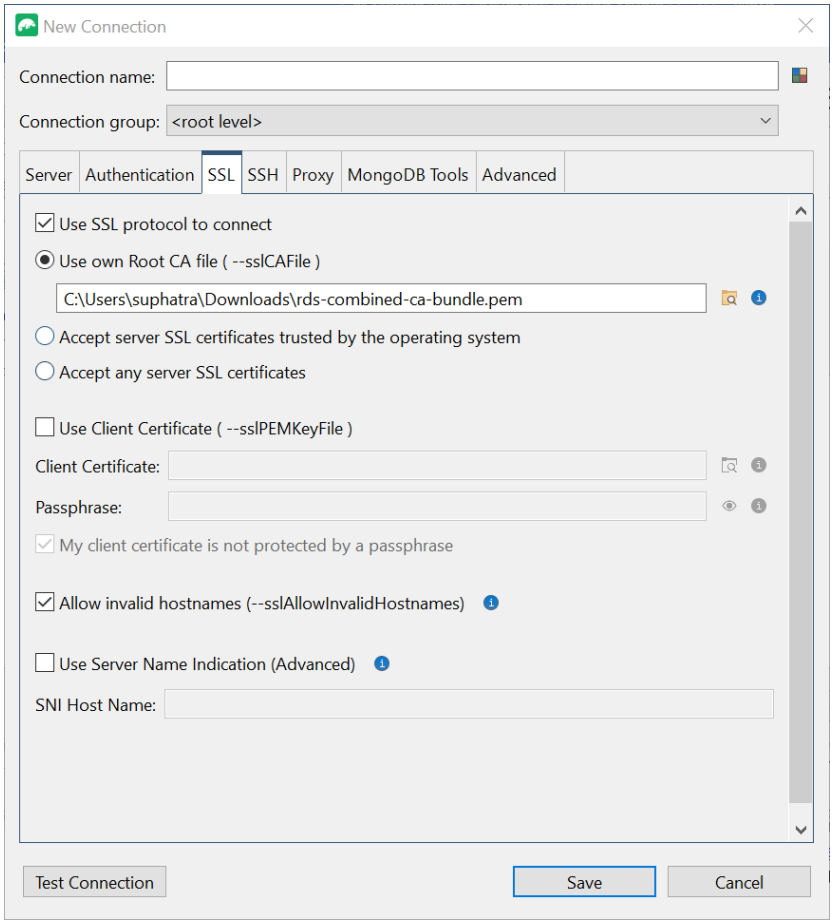
4. Pilih tab Autentikasi dan pilih Warisan di menu gulir ke bawah untuk Mode Autentikasi.



5. Masukkan nama pengguna dan kredensial Anda di bidang Nama pengguna dan Kata Sandi.
6. Pilih tab SSL dan centang kotak Gunakan protokol SSL untuk menghubungkan.



7. Pilih Gunakan file CA Root sendiri. Kemudian tambahkan sertifikat Amazon DocumentDB (Anda dapat melewatkan langkah ini jika SSL dinonaktifkan pada klaster DocumentDB Anda). Centang kotak untuk mengizinkan nama host tidak valid.



> **Note**
>
> Tidak memiliki sertifikat? Anda dapat mengunduhnya dengan perintah berikut ini: `wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem`

8. Jika Anda menghubungkan dari mesin klien di luar Amazon VPC, Anda perlu membuat terowongan SSH. Anda akan melakukannya di tab SSH.

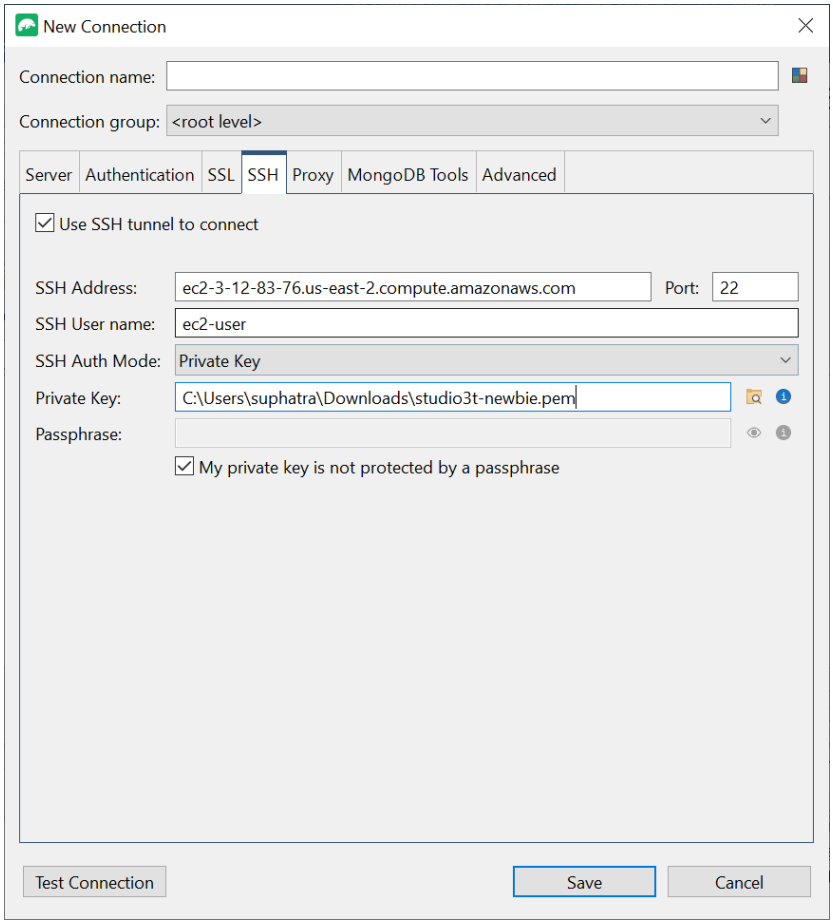
   a. Centang kotak untuk Gunakan terowongan SSH dan masukkan alamat SSH di bidang Alamat SSH. Ini adalah DNS Publik instans Anda (IPv4). Anda bisa mendapatkan URL ini dari Konsol Manajemen Amazon EC2.

   b. Masukkan nama pengguna Anda. Ini adalah nama pengguna instans Amazon EC2 Anda

   c. Untuk Mode Auten SSH, pilih Kunci Privat. Di bidang Kunci Privat, pilih ikon finder file untuk menemukan dan memilih kunci Privat dari instans Amazon EC2 Anda. Ini adalah file .pem (pasangan kunci) yang Anda simpan saat membuat instans Anda di Konsol Amazon EC2.

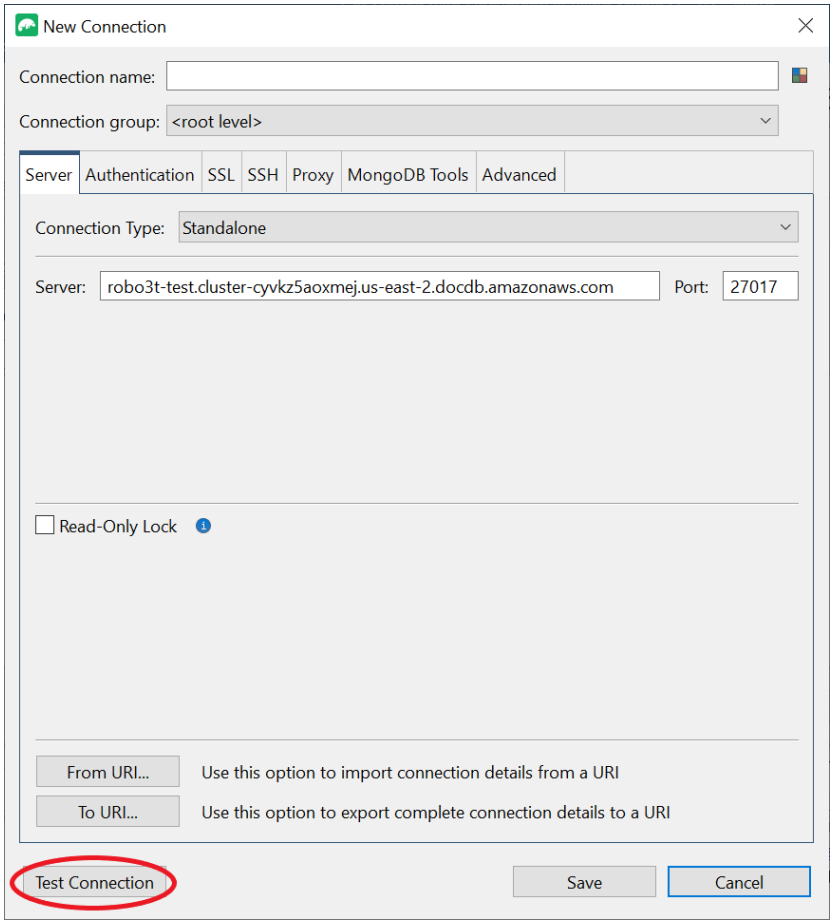
d. Jika Anda menggunakan mesin Linux/macOS klien, Anda mungkin harus mengubah izin kunci pribadi Anda menggunakan perintah berikut:

```
chmod 400 /fullPathToYourPemFile/<yourKey>.pem
```

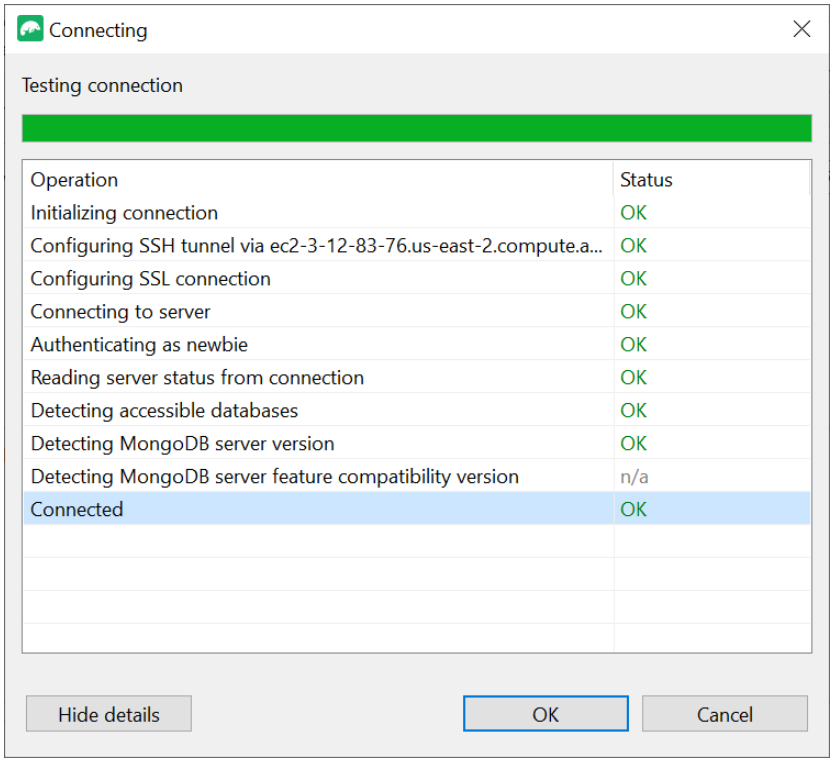


> **Note**
>
> Instans Amazon EC2 ini harus dalam Amazon VPC dan grup keamanan yang sama dengan klaster DocumentDB Anda. Anda bisa mendapatkan alamat SSH, nama pengguna, dan kunci privat dari Konsol Manajemen Amazon EC2.

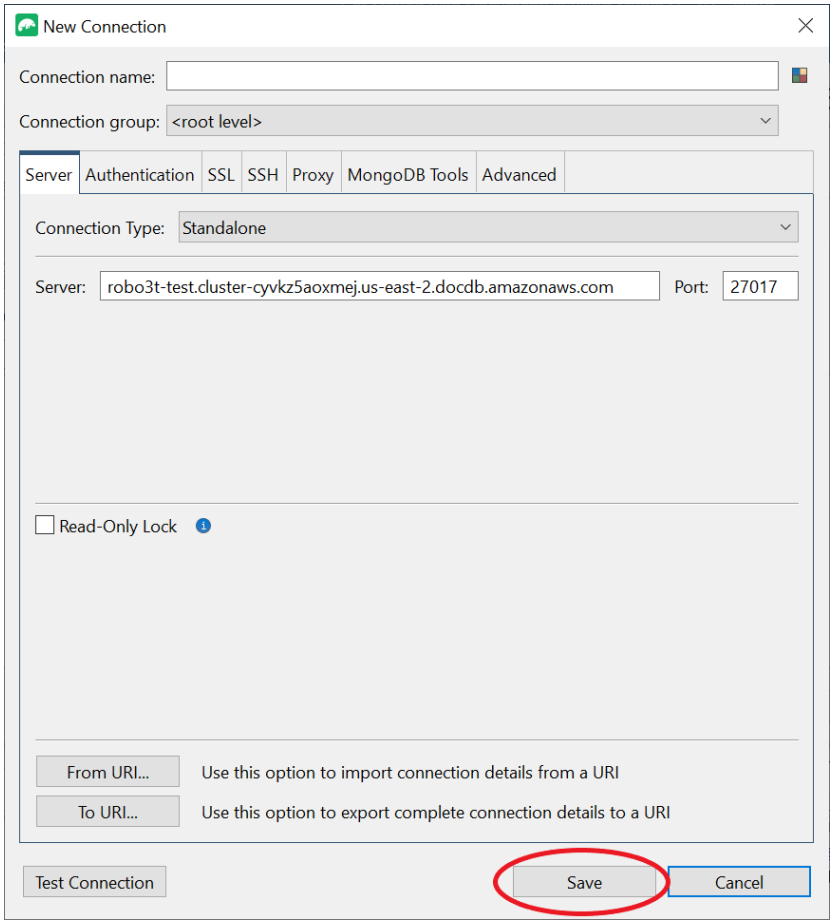
9. Sekarang uji konfigurasi Anda dengan memilih tombol Uji koneksi.



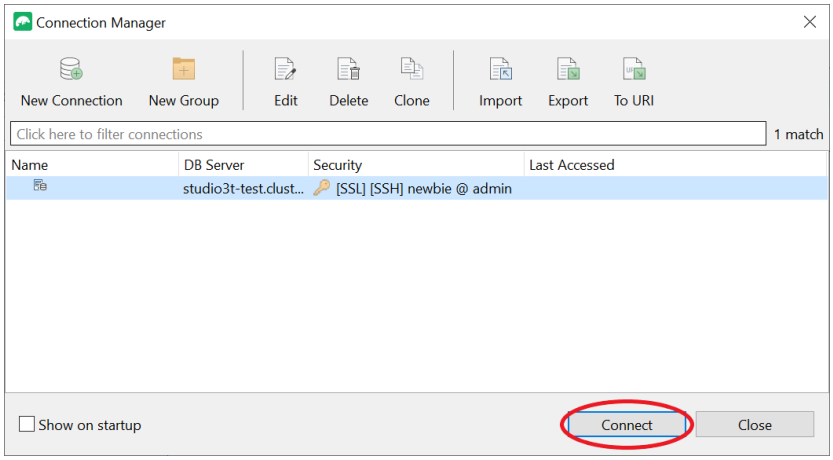
10. Jendela diagnostik harus memuat bar hijau untuk menunjukkan bahwa pengujian berhasil. Sekarang pilih OKE untuk menutup jendela diagnostik.



11. Pilih Simpan untuk menyimpan koneksi Anda untuk penggunaan masa depan.



12. Sekarang pilih klaster Anda dan pilih Hubungkan.



Selamat! Anda sekarang berhasil terhubung ke klaster Amazon DocumentDB Anda melalui Studio 3T.

Connect ke Amazon DocumentDB menggunakan DataGrip

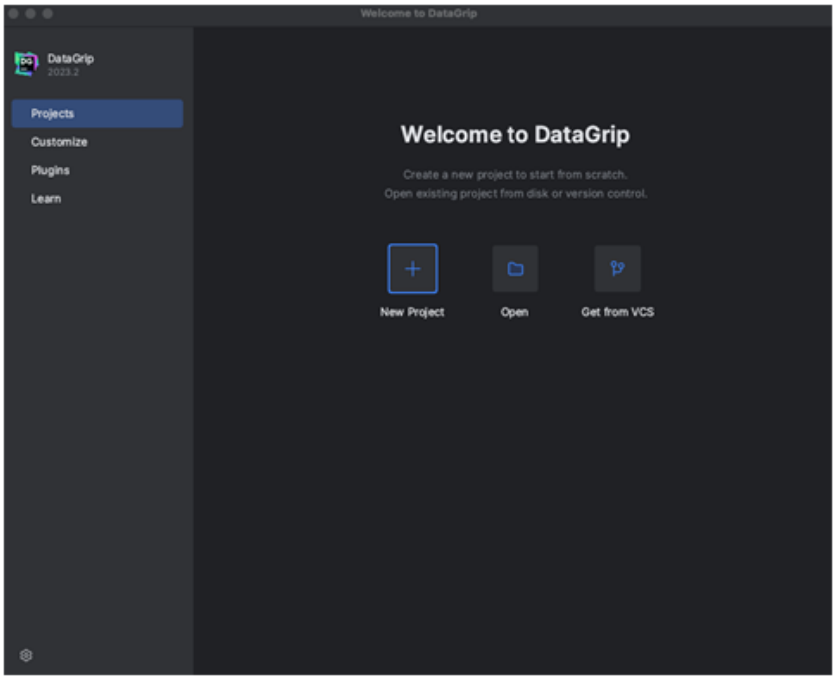
DataGripadalah lingkungan pengembangan terintegrasi (IDE) yang kuat yang mendukung berbagai sistem database, termasuk Amazon DocumentDB. Bagian ini memandu Anda melalui langkah-langkah untuk terhubung ke cluster Amazon DocumentDB Anda DataGrip menggunakan, memungkinkan Anda untuk dengan mudah mengelola dan menanyakan data Anda menggunakan antarmuka grafis.

Prasyarat

- DataGrip IDE diinstal pada mesin Anda. Anda dapat mengunduhnya dari JetBrains.
- Instans Amazon EC2 yang berjalan di VPC yang sama dengan cluster Amazon DocumentDB Anda. Anda akan menggunakan instance ini untuk membuat terowongan aman dari mesin lokal Anda ke Amazon DocumentDBCluster. Ikuti instruksi tentang cara melakukannyaConnect menggunakan Amazon EC2.
- Alternatif untuk instans Amazon EC2, koneksi VPN, atau jika Anda sudah mengakses AWS infrastruktur Anda menggunakan VPN aman. Jika Anda lebih suka opsi ini, ikuti petunjuk untuk mengakses Amazon DocumentDB dengan aman menggunakan. AWS Client VPN

Connect menggunakan DataGrip

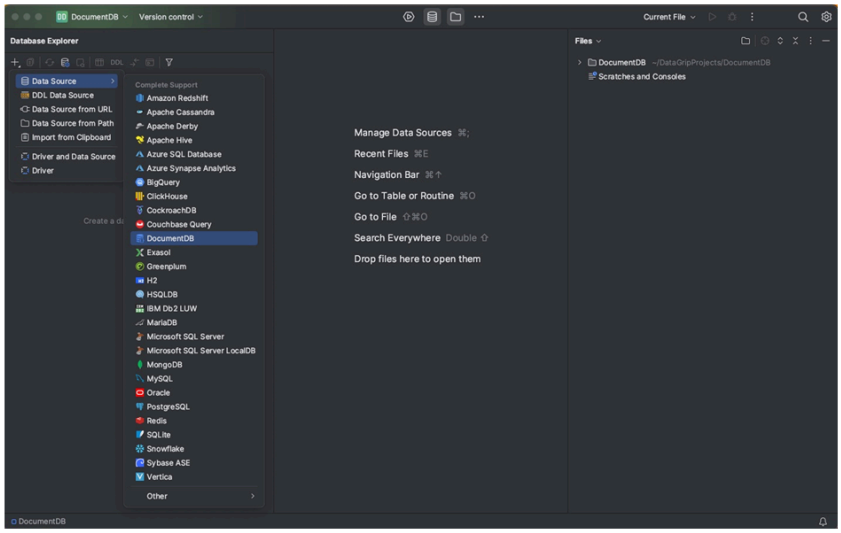
1. Luncurkan DataGrip di komputer Anda dan buat Proyek Baru.



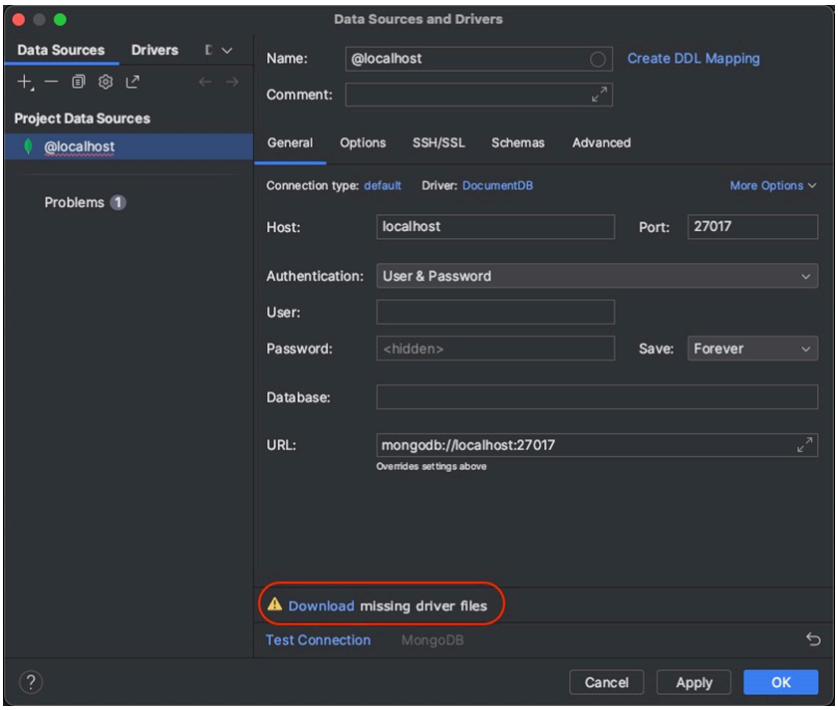
2. Tambahkan sumber data baru menggunakan salah satu cara berikut:

   a. Dari menu utama, navigasikan ke File — New — Data Source dan pilih DocumentDB

   b. Di Database Explorer, pilih ikon baru (+) di toolbar. Arahkan ke Sumber Data dan pilih DocumentDB.



3. Pada halaman Sumber Data di tab Umum, periksa apakah ada tautan Unduh file driver yang hilang di bagian bawah area pengaturan koneksi. Pilih tautan ini untuk mengunduh driver yang diperlukan untuk berinteraksi dengan database. Untuk tautan unduhan langsung, lihat driver JetBrains JDBC.



4. Di tab Umum, tentukan detail koneksi:

a. Di bidang Host, tentukan titik akhir cluster Amazon DocumentDB.

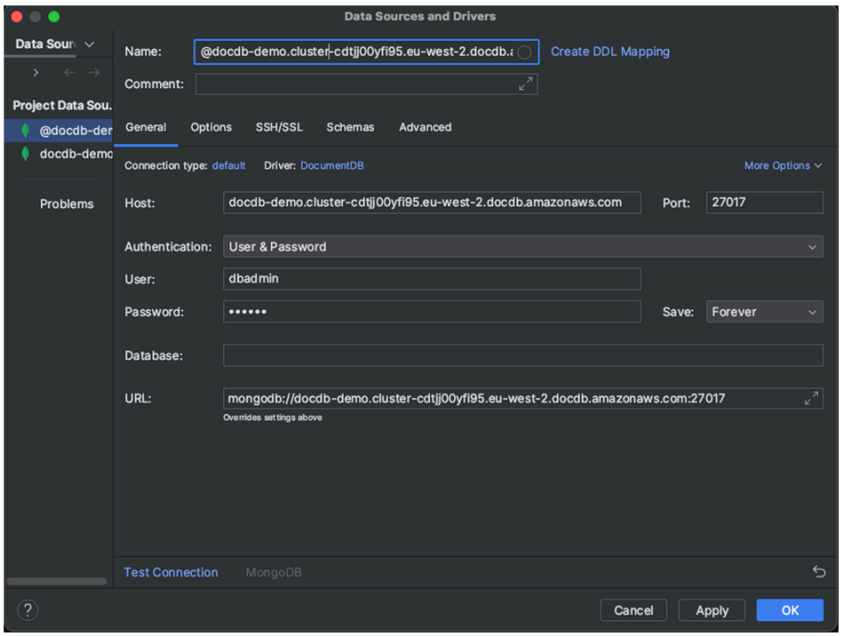
b. Port sudah diatur ke 27017. Ubah jika cluster Anda diterapkan pada port yang berbeda.

c. Untuk Otentikasi, pilih User & Password.

d. Masukkan nama pengguna dan informasi kata sandi Anda.

e. Bidang Database adalah opsional. Anda dapat menentukan database yang ingin Anda sambungkan.

f. Bidang URL otomatis selesai saat Anda menambahkan detail sebelumnya.



5. Di SSH/SSLtab, aktifkan Gunakan terowongan SSH, lalu pilih ikon untuk membuka dialog Konfigurasi SSH. Masukkan informasi berikut:

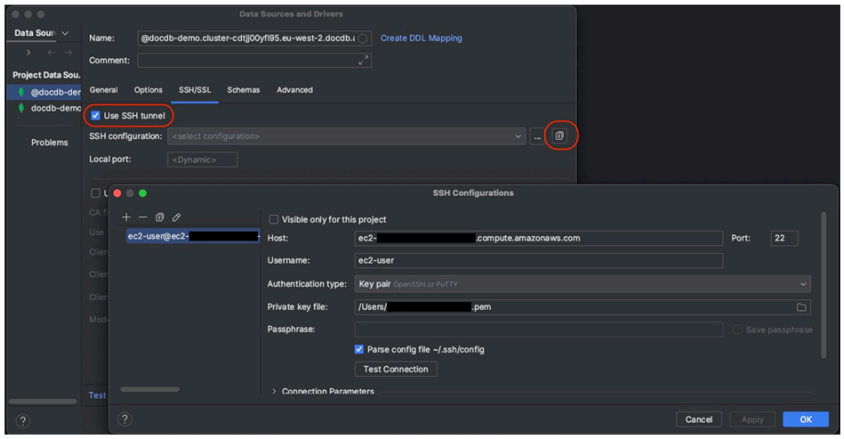
a. di bidang Host, masukkan nama host instans Amazon EC2 Anda.

b. Masukkan nama pengguna dan kata sandi untuk instans Amazon EC2 Anda.

c. Untuk jenis Otentikasi, pilih Pasangan kunci.

d. Masukkan file kunci pribadi Anda.

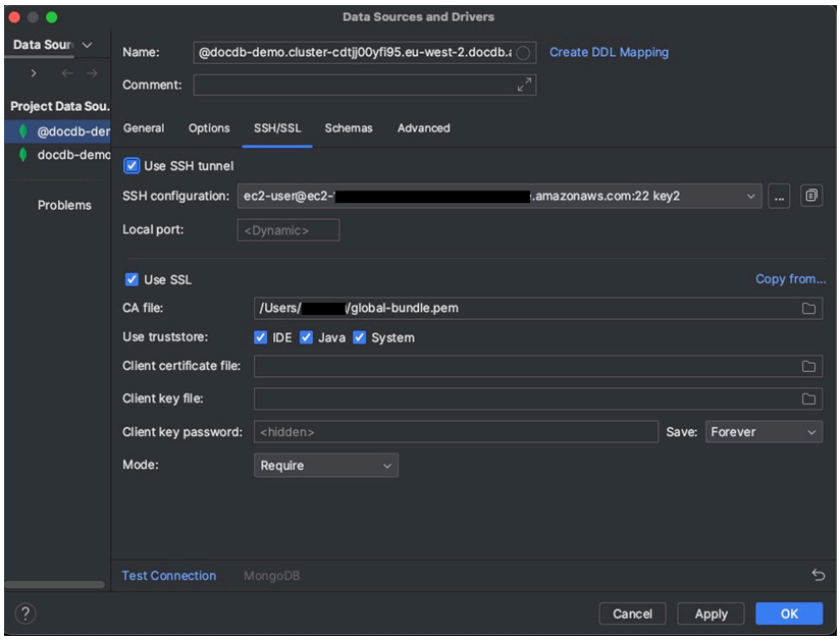
> **Note**
>
> Jika Anda menggunakan opsi VPN, tidak perlu mengkonfigurasi terowongan SSH.



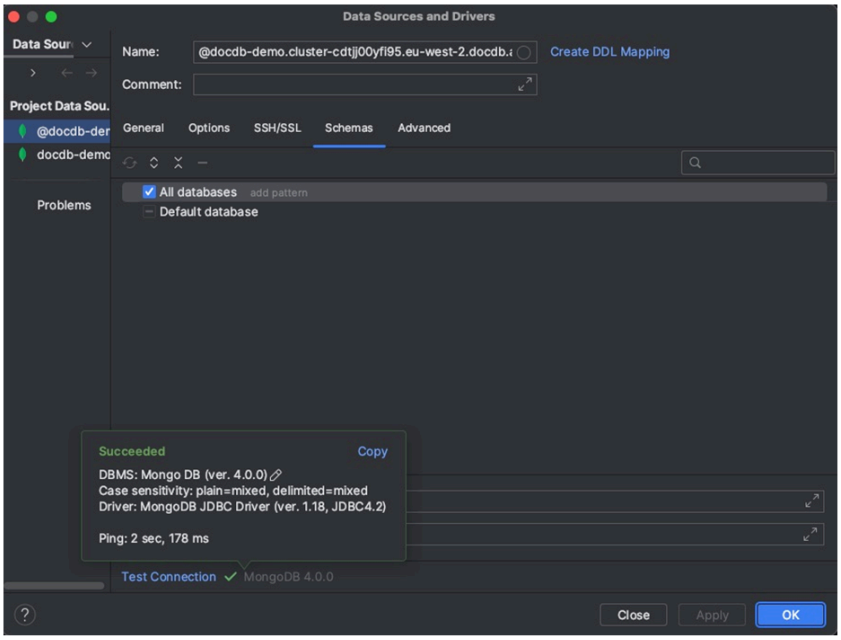
6. Di SSH/SSLtab, aktifkan Gunakan SSL. Di bidang file CA, masukkan lokasi ke `global-bundle.pem` file di komputer Anda. Untuk Mode, tinggalkan opsi Memerlukan.

> **Note**
>
> Anda dapat mengunduh sertifikat dari lokasi ini atau dengan perintah ini: wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem

> **Note**
>
> Jika Anda terhubung ke Amazon DocumentDB cluster elastis, Anda tidak perlu menentukan file CA. Biarkan opsi Gunakan SSL dicentang dan semua opsi lainnya pada nilai defaultnya.



7. Di tab Skema, pilih Semua database atau masukkan filter “*: *” di bidang Pola skema. Pilih Uji Koneksi untuk menguji koneksi.



8. Setelah koneksi berhasil diuji, pilih OK untuk menyimpan konfigurasi sumber data.

DataGrip fitur

DataGrip menyediakan berbagai fitur untuk membantu Anda bekerja dengan Amazon DocumentDB secara efisien:

- SQL Editor - Tulis dan jalankan SQL-like kueri pada koleksi DocumentDB Anda menggunakan editor SQL di. DataGrip
- Visual Query Builder - Gunakan pembangun kueri visual untuk membuat kueri secara grafis tanpa menulis kode SQL.
- Manajemen Skema - Kelola skema database Anda dengan mudah, termasuk membuat, mengubah, dan menjatuhkan koleksi.
- Visualisasi Data — Melihat dan menganalisis data Anda menggunakan berbagai alat visualisasi yang tersedia di. DataGrip
- Ekspor dan Impor Data - Transfer data antara Amazon DocumentDB dan database lain DataGrip menggunakan fitur ekspor dan impor.

Lihat DataGrip dokumentasi resmi untuk fitur dan kiat yang lebih canggih tentang bekerja dengan Amazon DocumentDB dan sistem basis data lainnya.

Connect menggunakan Amazon EC2

Bagian ini menjelaskan cara mengatur konektivitas antara cluster Amazon DocumentDB dan Amazon EC2 dan mengakses cluster Amazon DocumentDB dari instans Amazon EC2.

Ada dua opsi untuk mengkonfigurasi koneksi EC2:

- Sambungkan instans EC2 Anda secara otomatis ke database Amazon DocumentDB — Gunakan fitur koneksi otomatis di konsol EC2 untuk secara otomatis mengonfigurasi koneksi antara instans EC2 Anda dan database Amazon DocumentDB baru atau yang sudah ada. Koneksi ini memungkinkan lalu lintas untuk melakukan perjalanan antara instans EC2 dan database Amazon DocumentDB. Opsi ini biasanya digunakan untuk menguji dan membuat grup keamanan baru.
- Hubungkan instans EC2 Anda secara manual ke database Amazon DocumentDB Anda — Konfigurasikan koneksi antara instans EC2 Anda ke database Amazon DocumentDB Anda dengan mengonfigurasi dan menetapkan grup keamanan secara manual untuk mereproduksi konfigurasi yang dibuat oleh fitur koneksi otomatis. Opsi ini biasanya digunakan untuk mengubah pengaturan yang lebih maju dan menggunakan grup keamanan yang ada.

Prasyarat

Terlepas dari opsi, dan sebelum Anda membuat cluster Amazon DocumentDB pertama Anda, Anda harus melakukan hal berikut:

Buatlah akun Amazon Web Services (AWS)

Sebelum Anda dapat mulai menggunakan Amazon DocumentDB, Anda harus memiliki akun Amazon Web Services (AWS). AWS Akun ini gratis. Anda hanya membayar untuk layanan dan sumber daya yang Anda gunakan.

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar untuk Akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.
2. Ikuti petunjuk online.

   Bagian dari prosedur pendaftaran melibatkan menerima panggilan telepon atau pesan teks dan memasukkan kode verifikasi pada keypad telepon.

   Saat Anda mendaftar untuk sebuah Akun AWS, sebuah Pengguna root akun AWSdibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya di akun. Sebagai praktik keamanan terbaik, tetapkan akses administratif ke pengguna, dan gunakan hanya pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

Secara opsional, atur izin yang diperlukan AWS Identity and Access Management (IAM).

Akses untuk mengelola sumber daya Amazon DocumentDB seperti cluster, instance, dan grup parameter cluster memerlukan kredensil yang dapat digunakan untuk AWS mengautentikasi permintaan Anda. Untuk informasi selengkapnya, lihat Identity and Access Management untuk Amazon DocumentDB.

1. Di bilah pencarian Konsol Manajemen AWS, ketik IAM dan pilih IAM di menu tarik-turun yang muncul.
2. Setelah Anda berada di konsol IAM, pilih Pengguna dari panel navigasi.
3. Pilih nama pengguna Anda.
4. Pilih Tambahkan izin.

5. Pilih Lampirkan kebijakan yang ada secara langsung.
6. Ketik `AmazonDocDBFullAccess` di bilah pencarian dan pilih itu setelah muncul di hasil pencarian.
7. Pilih Berikutnya: Tinjauan.
8. Pilih Tambahkan izin.

Buat Amazon Virtual Private Cloud (Amazon VPC)

Tergantung di mana Wilayah AWS Anda berada, Anda mungkin atau mungkin tidak memiliki VPC default yang sudah dibuat. Jika Anda tidak memiliki VPC default, selesaikan langkah 1 dari Memulai Amazon VPC di Panduan Pengguna Amazon VPC. Ini akan memakan waktu kurang dari lima menit.

Hubungkan Amazon EC2 secara otomatis

Topik

- Secara otomatis menghubungkan instans EC2 ke database Amazon DocumentDB baru
- Secara otomatis menghubungkan instans EC2 ke database Amazon DocumentDB yang ada
- Ikhtisar konektivitas otomatis dengan instans EC2
- Melihat sumber daya komputasi terhubung

Sebelum menyiapkan sambungan antara instans EC2 dan database Amazon DocumentDB baru, pastikan Anda memenuhi persyaratan yang dijelaskan dalam. Ikhtisar konektivitas otomatis dengan instans EC2 Jika Anda membuat perubahan pada grup keamanan setelah mengonfigurasi konektivitas, perubahan tersebut dapat memengaruhi koneksi antara instans EC2 dan database Amazon DocumentDB.

> **Note**
>
> Anda hanya dapat mengatur koneksi antara instans EC2 dan database Amazon DocumentDB secara otomatis dengan menggunakan file. Konsol Manajemen AWS Anda tidak dapat mengatur koneksi secara otomatis dengan Amazon DocumentDB API AWS CLI atau Amazon.

Secara otomatis menghubungkan instans EC2 ke database Amazon DocumentDB baru

Proses berikut mengasumsikan Anda telah menyelesaikan langkah-langkah dalam Prasyarat topik.

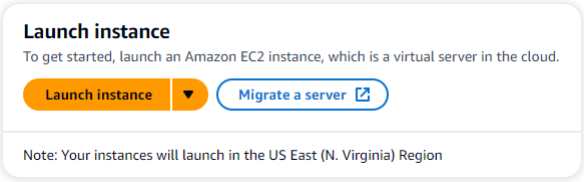
Langkah-langkah

- Langkah 1: Buat instans Amazon EC2
- Langkah 2: Buat cluster Amazon DocumentDB
- Langkah 3: Connect ke instans Amazon EC2
- Langkah 4: Instal MongoDB Shell
- Langkah 5: Kelola Amazon DocumentDB TLS
- Langkah 6: Connect ke cluster Amazon DocumentDB Anda
- Langkah 7: Masukkan dan kueri data
- Langkah 8: Jelajahi

Langkah 1: Buat instans Amazon EC2

Pada langkah ini, Anda akan membuat instans Amazon EC2 di Wilayah yang sama dan Amazon VPC yang nantinya akan Anda gunakan untuk menyediakan klaster Amazon DocumentDB Anda.

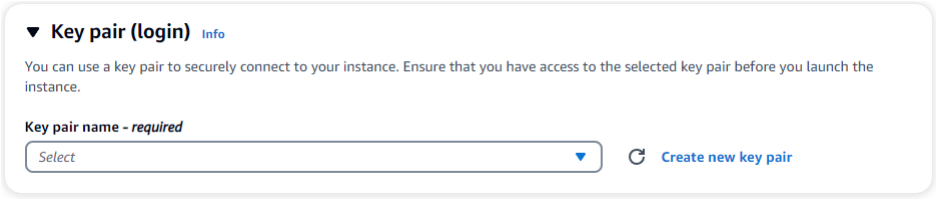
1. Pada konsol Amazon EC2, pilih Luncurkan instans.



2. Masukkan nama atau pengenal di bidang Nama yang terletak di bagian Nama dan tag.
3. Di daftar drop-down Amazon Machine Image (AMI), cari Amazon Linux 2 AMI dan pilih.

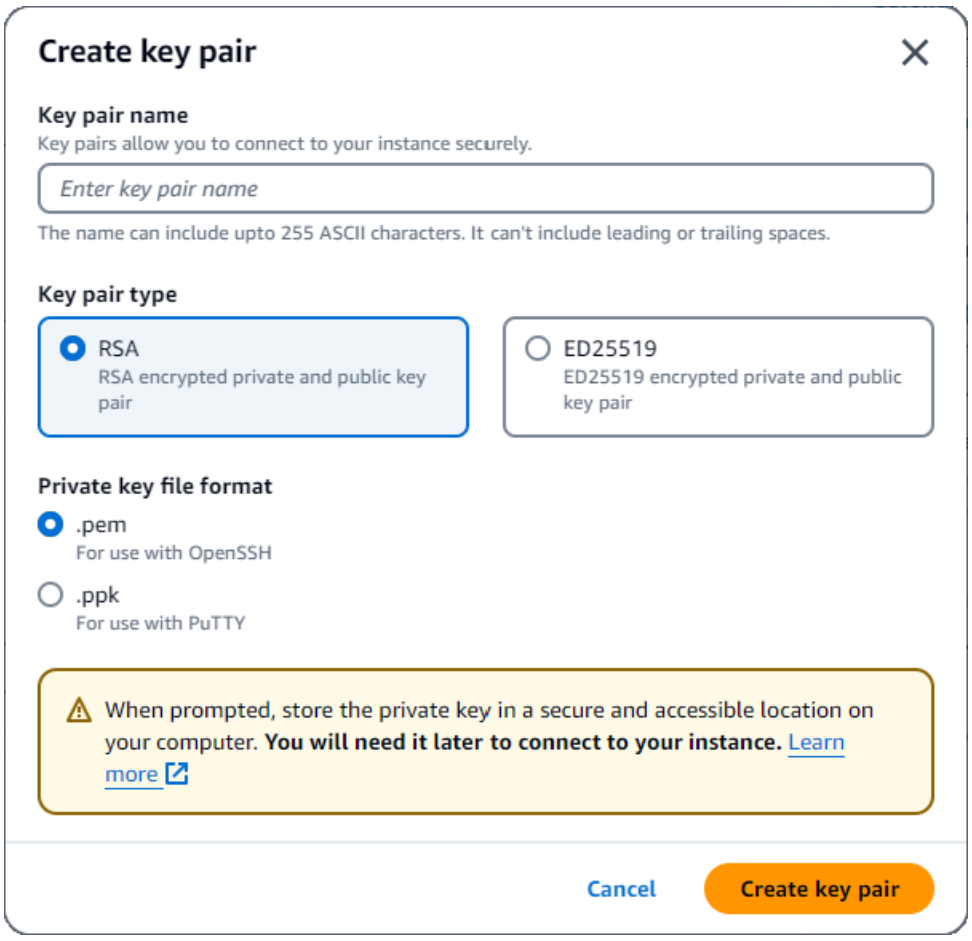


4. Cari dan pilih t3.micro dalam daftar drop-down tipe Instance.
5. Di bagian Key pair (login), masukkan identifier dari key-pair yang ada, atau pilih Create new key pair.



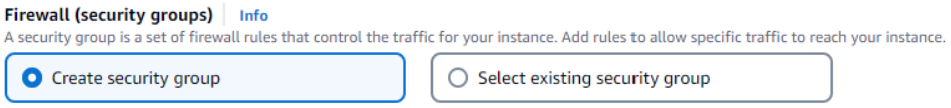
Anda harus menyediakan pasangan kunci Amazon EC2.

- Jika Anda memiliki pasangan kunci Amazon EC2:
  1. Pilih pasangan kunci, pilih pasangan kunci Anda dari daftar.
  2. Anda harus sudah memiliki file kunci pribadi (file.pem atau .ppk) yang tersedia untuk masuk ke instans Amazon EC2 Anda.
- Jika Anda tidak memiliki pasangan kunci Amazon EC2:
  1. Pilih Create new key pair, kotak dialog Create key pair muncul.
  2. Masukkan nama di bidang Nama pasangan kunci.
  3. Pilih jenis pasangan kunci dan format file kunci pribadi.
  4. Pilih Buat pasangan kunci.



> **Note**
>
> Untuk tujuan keamanan, kami sangat menyarankan menggunakan pasangan kunci untuk konektivitas SSH dan internet ke instans EC2 Anda.

6. Opsional: Di bagian Pengaturan jaringan, di bawah Firewall (grup keamanan), pilih Buat grup keamanan.



Pilih Buat grup keamanan (periksa semua aturan izin lalu lintas yang berlaku untuk konektivitas EC2 Anda).

> **Note**
>
> Jika Anda ingin menggunakan grup keamanan yang ada, ikuti instruksi diHubungkan Amazon EC2 secara manual.

7. Di bagian Ringkasan, tinjau konfigurasi EC2 Anda dan pilih Launch instance jika benar.

Langkah 2: Buat cluster Amazon DocumentDB

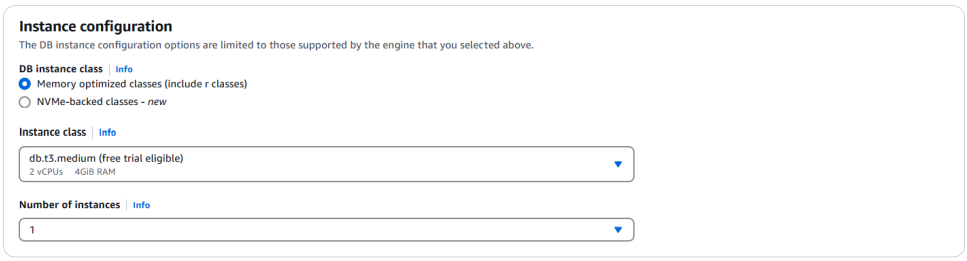
Saat instans Amazon EC2 sedang disediakan, buat klaster Amazon DocumentDB Anda.

1. Navigasikan ke konsol Amazon DocumentDB dan pilih Klaster dari panel navigasi.
2. Pilih Buat.
3. Biarkan pengaturan tipe Cluster pada defaultnya dari Instance Based Cluster.
4. Dalam konfigurasi Cluster, untuk pengidentifikasi Cluster, masukkan nama unik. Perhatikan bahwa konsol akan mengubah semua nama cluster menjadi huruf kecil terlepas dari bagaimana mereka dimasukkan.

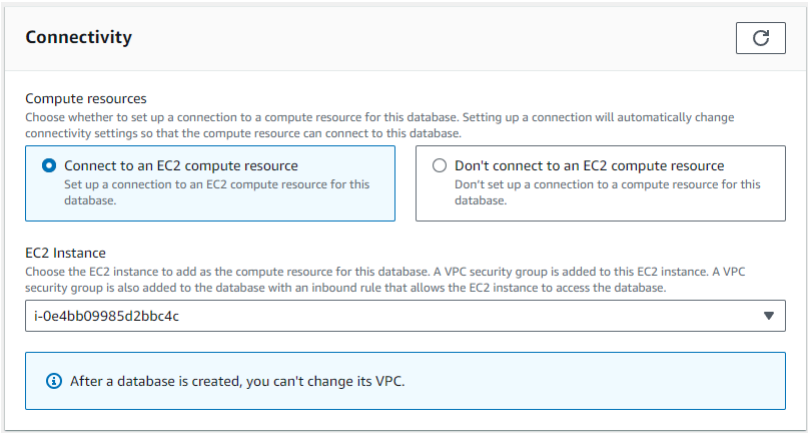
   Biarkan versi Engine pada nilai defaultnya 5.0.0.
5. Untuk konfigurasi penyimpanan Cluster, tinggalkan pengaturan default Amazon DocumentDB Standard.
6. Dalam konfigurasi Instance:

   - Untuk kelas instans DB, pilih kelas yang dioptimalkan memori (termasuk kelas r) (ini adalah default).

     Pilihan contoh lainnya adalah NVMe-backed kelas. Untuk mempelajari selengkapnya, lihat NVMe-backed contoh.
   - Untuk kelas Instance, pilih jenis instance yang sesuai dengan kebutuhan Anda. Untuk penjelasan lebih rinci tentang kelas instance, lihatSpesifikasi kelas instans.
   - Untuk jumlah contoh, pilih nomor yang paling mencerminkan kebutuhan Anda. Ingat, semakin rendah angkanya, semakin rendah biayanya, dan semakin rendah read/write volume yang dapat dikelola oleh cluster.



7. Untuk Konektivitas, pilih Connect ke sumber daya komputasi EC2. Ini adalah contoh EC2 yang Anda buat di Langkah 1.



> **Note**
>
> Menyambung ke sumber daya komputasi EC2 secara otomatis membuat grup keamanan untuk koneksi sumber daya komputasi EC2 Anda ke cluster Amazon DocumentDB Anda. Setelah selesai membuat klaster dan ingin melihat grup keamanan yang baru dibuat, navigasikan ke daftar klaster dan pilih pengenal klaster Anda. Di tab Konektivitas & keamanan, buka Grup Keamanan dan temukan grup Anda di bawah Nama grup keamanan (ID). Ini akan berisi nama cluster Anda dan terlihat mirip dengan ini:`docdb-ec2-docdb-2023-12-11-21-33-41:i-0e4bb09985d2bbc4c (sg-0238e0b0bf0f73877)`.

8. Di bagian Autentikasi, masukkan nama pengguna untuk pengguna utama, lalu pilih Self managed. Masukkan kata sandi, lalu konfirmasikan.

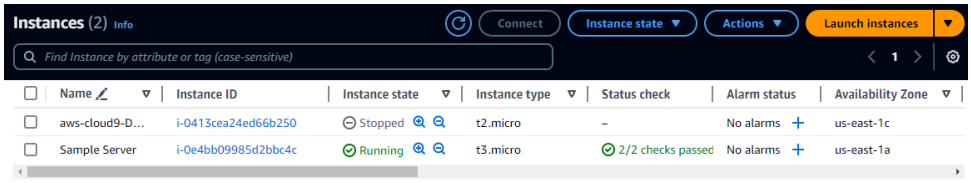
   Jika Anda memilih Managed in AWS Secrets Manager, lihat Manajemen kata sandi dengan Amazon DocumentDB dan AWS Secrets Manager untuk informasi selengkapnya.

9. Pilih Buat klaster.

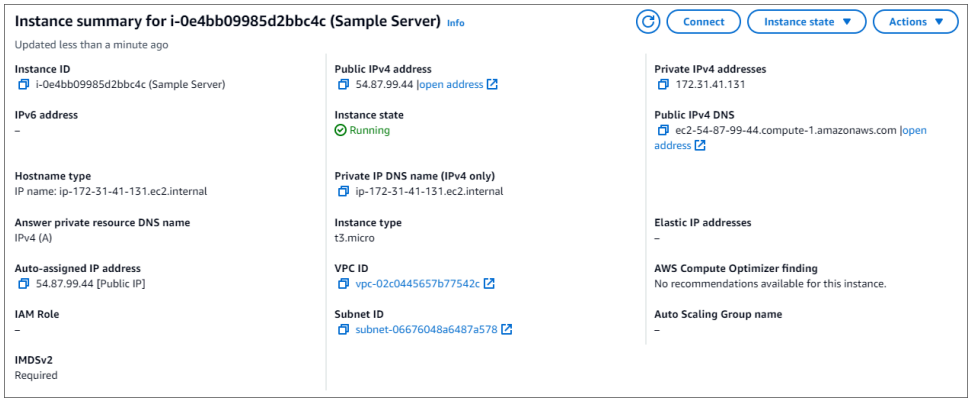
Langkah 3: Connect ke instans Amazon EC2

Untuk menginstal shell mongo, Anda harus terlebih dahulu terhubung ke instans Amazon EC2 Anda. Instalasi shell mongo memungkinkan Anda untuk terhubung ke dan kueri klaster Amazon DocumentDB Anda. Selesaikan langkah-langkah berikut:

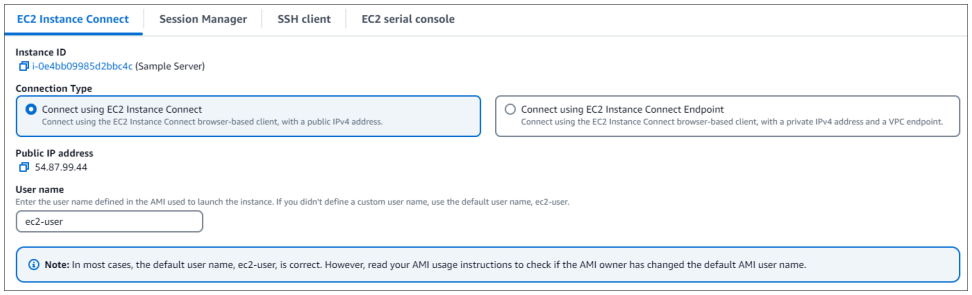
1. Pada konsol Amazon EC2, navigasikan ke instans Anda dan lihat apakah instans yang baru saja Anda buat berjalan. Jika ya, pilih instance dengan mengklik ID instance.



2. Pilih Hubungkan.



3. Ada empat opsi tab untuk metode koneksi Anda: Amazon EC2 Instance Connect, Session Manager, SSH client, atau konsol serial EC2. Anda harus memilih satu dan mengikuti instruksinya. Setelah selesai, pilih Connect.



> **Note**
>
> Jika alamat IP Anda berubah setelah Anda memulai penelusuran ini, atau Anda kembali ke lingkungan Anda di lain waktu, Anda harus memperbarui aturan masuk grup `demoEC2` keamanan Anda untuk mengaktifkan lalu lintas masuk dari alamat API baru Anda.

Langkah 4: Instal MongoDB Shell

Anda sekarang dapat menginstal shell MongoDB, yang merupakan utilitas baris perintah yang Anda gunakan untuk menghubungkan dan menanyakan cluster Amazon DocumentDB Anda. Saat ini ada dua versi shell MongoDB: versi terbaru, mongosh, dan versi sebelumnya, mongo shell.

> **Important**
>
> Ada batasan yang diketahui dengan Node.js driver yang lebih lama dari versi 6.13.1, yang saat ini tidak didukung oleh otentikasi identitas IAM untuk Amazon DocumentDB. Node.js driver dan alat yang menggunakan Node.js driver (misalnya, mongosh) harus ditingkatkan untuk menggunakan Node.js driver versi 6.13.1 atau lebih tinggi.

Ikuti instruksi di bawah ini untuk menginstal MongoDB shell untuk sistem operasi Anda.

On Amazon Linux

Untuk menginstal shell MongoDB di Amazon Linux

Jika Anda tidak menggunakan otentikasi IAM dan ingin menggunakan shell MongoDB terbaru (mongosh) untuk terhubung ke cluster Amazon DocumentDB Anda, ikuti langkah-langkah berikut:

1. Buat file repositori. Pada baris perintah instans EC2 yang Anda buat, jalankan perintah berikut:

```
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. Setelah selesai, instal mongosh dengan salah satu dari dua opsi perintah berikut di prompt perintah:

Opsi 1 - Jika Anda memilih default Amazon Linux 2023 selama konfigurasi Amazon EC2, masukkan perintah ini:

```
sudo yum install -y mongodb-mongosh-shared-openssl3
```

Opsi 2 - Jika Anda memilih Amazon Linux 2 selama konfigurasi Amazon EC2, masukkan perintah ini:

```
sudo yum install -y mongodb-mongosh
```

Jika Anda menggunakan otentikasi IAM, Anda harus menggunakan shell MongoDB versi sebelumnya (5.0) untuk terhubung ke cluster Amazon DocumentDB Anda, ikuti langkah-langkah berikut:

1. Buat file repositori. Pada baris perintah instans EC2 yang Anda buat, jalankan perintah berikut:

```
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. Setelah selesai, instal shell mongodb 5.0 dengan opsi perintah berikut di prompt perintah:

```
sudo yum install -y mongodb-org-shell
```

On Ubuntu

Untuk menginstal mongosh di Ubuntu

1. Impor kunci publik yang akan digunakan oleh sistem manajemen paket.

```
curl -fsSL https://pgp.mongodb.com/server-5.0.asc | sudo gpg --dearmor -o /usr/
share/keyrings/mongodb-server-5.0.gpg
```

2. Buat file daftar `mongodb-org-5.0.list` untuk MongoDB menggunakan perintah yang sesuai untuk versi Ubuntu Anda.

```
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. Impor dan perbarui database paket lokal menggunakan perintah berikut:

```
sudo apt-get update
```

4. Instal mongosh.

```
sudo apt-get install -y mongodb-mongosh
```

Untuk informasi tentang menginstal versi sebelumnya dari MongoDB pada sistem Ubuntu Anda, lihat Instal MongoDB Community Edition di Ubuntu.

On other operating systems

Untuk menginstal shell mongo pada sistem operasi lain, lihat Instal MongoDB Community Edition dalam dokumentasi MongoDB.

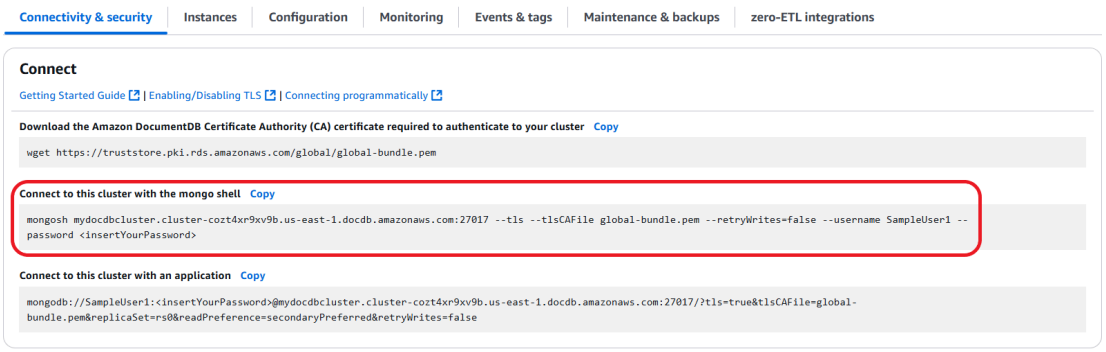
Langkah 5: Kelola Amazon DocumentDB TLS

Unduh sertifikat CA untuk Amazon DocumentDB dengan kode berikut: `wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem`

> **Note**
>
> Keamanan Lapisan Pengangkutan (TLS) diaktifkan secara default untuk setiap klaster Amazon DocumentDB baru. Untuk informasi selengkapnya, lihat Mengelola setelan TLS cluster Amazon DocumentDB.

Langkah 6: Connect ke cluster Amazon DocumentDB Anda

1. Di konsol Amazon DocumentDB, di bawah Clusters, temukan klaster Anda. Pilih cluster yang Anda buat dengan mengklik pengenal Cluster untuk cluster tersebut.
2. Di tab Konektivitas dan keamanan, cari Connect to this cluster dengan shell mongo di kotak Connect:



Salin string koneksi yang disediakan dan tempel ke terminal Anda.

Buat perubahan berikut untuk itu:

a. Pastikan Anda memiliki nama pengguna yang benar dalam string.

b. Hilangkan `<insertYourPassword>` sehingga Anda diminta kata sandi oleh shell mongo saat Anda terhubung.

c. Opsional: Jika Anda menggunakan otentikasi IAM, atau menggunakan shell MongoDB versi sebelumnya, ubah string koneksi Anda sebagai berikut:

```
mongo --ssl --host mydocdbcluster.cluster-cozt4xr9xv9b.us-east-1.docdb.amazonaws.com:27017 --sslCAFile global-bundle.pem --username SampleUser1 --password
```

Ganti `mydocdbcluster.cluster-cozt4xr9xv9b.us-east-1` dengan informasi yang sama dari cluster Anda.

3. Tekan enter di terminal Anda. Anda sekarang diminta untuk kata sandi Anda. Masukkan kata sandi Anda.
4. Saat Anda memasukkan kata sandi dan dapat melihat perintah `rs0 [direct: primary] <env-name>>`, Anda berhasil terhubung ke klaster Amazon DocumentDB Anda.

Mengalami masalah saat menghubungkan? Lihat Pemecahan Masalah Amazon DocumentDB.

Langkah 7: Masukkan dan kueri data

Sekarang setelah Anda terhubung ke klaster, Anda dapat menjalankan beberapa kueri untuk mengenali penggunaan basis data dokumen.

1. Untuk memasukkan satu dokumen, masukkan hal berikut ini:

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

Anda mendapatkan output berikut ini:

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

2. Anda dapat membaca dokumen yang Anda tulis dengan perintah `findOne()` (karena hanya mengembalikan satu dokumen). Masukkan yang berikut ini:

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

Anda mendapatkan output berikut ini:

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

3. Untuk mengajukan beberapa kueri lagi, pertimbangkan kasus penggunaan profil game. Pertama, masukkan beberapa entri ke dalam koleksi berjudul `profiles`. Masukkan yang berikut ini:

```
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 }
])
```

Anda mendapatkan output berikut ini:

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

4. Gunakan perintah `find()` untuk mengembalikan semua dokumen dalam koleksi profil. Masukkan berikut ini:

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

Anda akan mendapatkan output yang akan cocok dengan data yang Anda ketik di Langkah 3.

5. Gunakan kueri untuk satu dokumen dengan menggunakan filter. Masukkan yang berikut ini:

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

Anda mendapatkan output berikut ini:

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

6. Sekarang mari kita coba untuk menemukan profil dan memodifikasinya dengan menggunakan perintah `findAndModify`. Kami akan memberi pengguna Matt tambahan 10 poin dengan kode berikut:

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

Anda mendapatkan output berikut (perhatikan bahwa skornya belum meningkat):

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

7. Anda dapat memverifikasi bahwa skornya telah berubah dengan kueri berikut:

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

Anda mendapatkan output berikut ini:

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

Langkah 8: Jelajahi

Selamat! Anda telah berhasil menyelesaikan Panduan Quick Start untuk Amazon DocumentDB.

Apa selanjutnya? Pelajari cara memanfaatkan sepenuhnya basis data yang kuat ini dengan beberapa fitur populernya:

- Mengelola Amazon DocumentDB
- Penskalaan

- Mencadangkan dan memulihkan

> **Note**
>
> Untuk menghemat biaya, Anda dapat menghentikan klaster Amazon DocumentDB untuk mengurangi biaya atau menghapus klaster. Secara default, setelah 30 menit tidak aktif, AWS Cloud9 lingkungan Anda akan menghentikan instans Amazon EC2 yang mendasarinya.

Secara otomatis menghubungkan instans EC2 ke database Amazon DocumentDB yang ada

Prosedur berikut mengasumsikan Anda memiliki klaster Amazon DocumentDB yang sudah ada dan instans Amazon EC2 yang sudah ada.

Akses kluster Amazon DocumentDB Anda dan atur koneksi Amazon EC2

1. Akses kluster Amazon DocumentDB Anda.

   a. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb

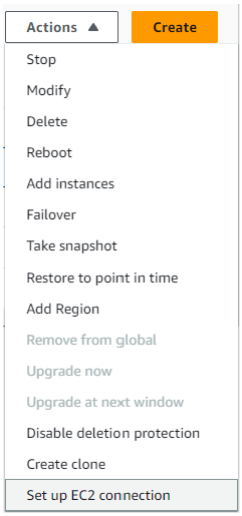
   b. Di panel navigasi, pilih Klaster.

      > **Tip**
      >
      > Jika Anda tidak melihat panel navigasi di sisi kiri layar, pilih ikon menu (≡) di pojok kiri atas halaman.

   c. Tentukan cluster yang Anda inginkan dengan memilih tombol di sebelah kiri nama cluster.

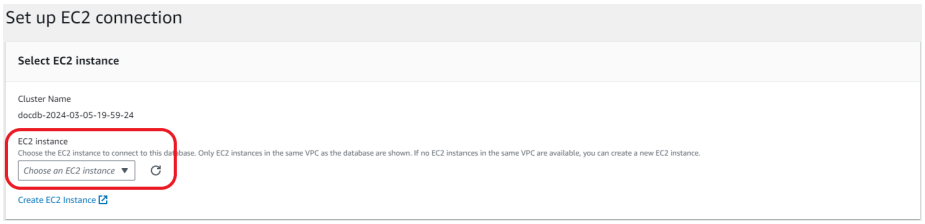
2. Siapkan koneksi Amazon EC2.

   a. Pilih Tindakan, lalu pilih Siapkan koneksi EC2.



Dialog Mengatur koneksi EC2 muncul.

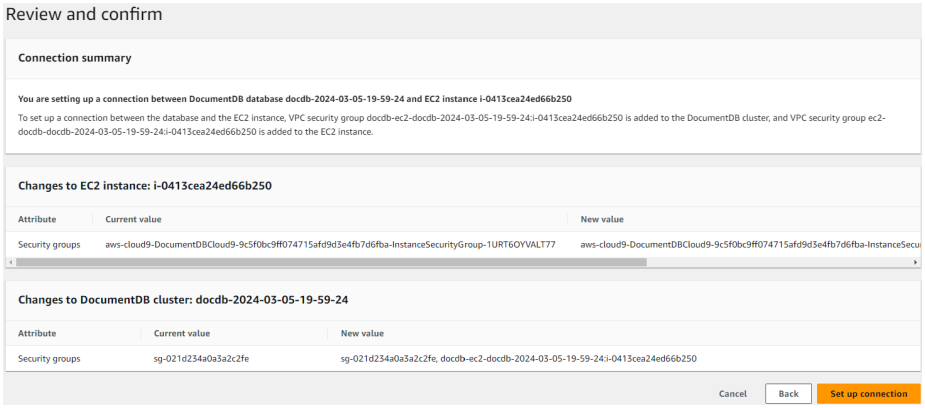
b. Di bidang instans EC2, pilih instans EC2 yang ingin Anda sambungkan ke cluster Anda.



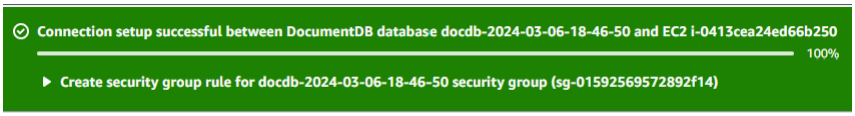
c. Pilih Lanjutkan.

Dialog Tinjau dan konfirmasi muncul.

d. Pastikan perubahannya benar. Kemudian pilih Siapkan koneksi.



Jika berhasil, verifikasi berikut akan muncul:



Ikhtisar konektivitas otomatis dengan instans EC2

Saat Anda mengatur koneksi antara instans EC2 dan database Amazon DocumentDB, Amazon DocumentDB secara otomatis mengonfigurasi grup keamanan VPC untuk instans EC2 Anda dan untuk database Amazon DocumentDB Anda.

Berikut ini adalah persyaratan untuk menghubungkan instans EC2 dengan database Amazon DocumentDB:

- Instans EC2 harus ada di VPC yang sama dengan database Amazon DocumentDB.

  Jika tidak ada instans EC2 di VPC yang sama, maka konsol menyediakan tautan untuk membuatnya.
- Pengguna yang menyiapkan konektivitas harus memiliki izin untuk melakukan operasi-operasi Amazon EC2 berikut:
  - `ec2:AuthorizeSecurityGroupEgress`
  - `ec2:AuthorizeSecurityGroupIngress`
  - `ec2:CreateSecurityGroup`
  - `ec2:DescribeInstances`
  - `ec2:DescribeNetworkInterfaces`
  - `ec2:DescribeSecurityGroups`
  - `ec2:ModifyNetworkInterfaceAttribute`
  - `ec2:RevokeSecurityGroupEgress`

Jika instans basis data dan instans EC2 berada di Zona Ketersediaan yang berbeda, akun Anda mungkin dikenakan biaya lintas Zona Ketersediaan.

Saat Anda menyiapkan sambungan ke instans EC2, Amazon DocumentDB bertindak sesuai dengan konfigurasi grup keamanan saat ini yang terkait dengan database Amazon DocumentDB dan instans EC2, seperti yang dijelaskan dalam tabel berikut:

Konfigurasi grup keamanan Amazon DocumentDB saat ini	Konfigurasi grup keamanan EC2 saat ini	Tindakan Amazon DocumentDB
Ada satu atau lebih grup keamanan yang terkait dengan database Amazon DocumentDB dengan nama yang cocok dengan polanya. `DocumentDB-ec2-n` Grup keamanan yang cocok dengan pola belum diubah. Grup keamanan ini memiliki hanya satu aturan masuk dengan grup keamanan VPC instans EC2 sebagai sumbernya.	Ada satu atau lebih grup keamanan yang terkait dengan instans EC2 dengan nama yang cocok dengan pola `DocumentDB-ec2-n` (di mana n adalah angka). Grup keamanan yang cocok dengan pola belum diubah. Grup keamanan ini hanya memiliki satu aturan keluar dengan grup keamanan VPC dari database Amazon DocumentDB sebagai sumbernya.	Amazon DocumentDB tidak mengambil tindakan. Koneksi sudah dikonfigurasi secara otomatis antara instans EC2 dan database Amazon DocumentDB. Karena koneksi sudah ada antara instans EC2 dan database Amazon DocumentDB, grup keamanan tidak dimodifikasi.
Salah satu syarat berikut berlaku: • Tidak ada grup keamanan yang terkait dengan database Amazon DocumentDB dengan nama yang cocok dengan polanya. `DocumentDB-ec2-n` • Ada satu atau lebih grup keamanan yang terkait dengan Amazon DocumentDB dengan nama yang cocok dengan polanya. `DocumentDB-ec2-n` Namun, Amazon DocumentDB tidak dapat menggunakan salah satu grup keamanan ini untuk	Salah satu syarat berikut dipenuhi: • Tidak ada grup keamanan yang terkait dengan instans EC2 dengan nama yang cocok dengan pola `ec2-DocumentDB-n` . • Ada satu atau beberapa grup keamanan yang terkait dengan instans EC2 dengan nama yang cocok dengan pola `ec2-DocumentDB-n`. Namun, Amazon DocumentDB tidak dapat menggunakan salah satu grup keamanan ini untuk koneksi dengan database Amazon DocumentDB. Amazon DocumentDB	Tindakan Amazon DocumentDB: buat grup keamanan baru

Konfigurasi grup keamanan Amazon DocumentDB saat ini	Konfigurasi grup keamanan EC2 saat ini	Tindakan Amazon DocumentDB
koneksi dengan instans EC2. Amazon DocumentDB tidak dapat menggunakan grup keamanan yang tidak memiliki satu aturan masuk dengan grup keamanan VPC dari instans EC2 sebagai sumbernya. Amazon DocumentDB juga tidak dapat menggunakan grup keamanan yang telah dimodifikasi. Contoh-contoh perubahan meliputi penambahan aturan atau pengubahan port aturan yang ada.	tidak dapat menggunakan grup keamanan yang tidak memiliki satu aturan keluar dengan grup keamanan VPC dari database Amazon DocumentDB sebagai sumbernya. Amazon DocumentDB juga tidak dapat menggunakan grup keamanan yang telah dimodifikasi.	

Konfigurasi grup keamanan Amazon DocumentDB saat ini	Konfigurasi grup keamanan EC2 saat ini	Tindakan Amazon DocumentDB
Ada satu atau lebih grup keamanan yang terkait dengan database Amazon DocumentDB dengan nama yang cocok dengan polanya. `DocumentDB-ec2-n` Grup keamanan yang cocok dengan pola belum diubah. Grup keamanan ini memiliki hanya satu aturan masuk dengan grup keamanan VPC instans EC2 sebagai sumbernya.	Ada satu atau beberapa grup keamanan yang terkait dengan instans EC2 dengan nama yang cocok dengan pola `ec2-DocumentDB-n` . Namun, Amazon DocumentDB tidak dapat menggunakan salah satu grup keamanan ini untuk koneksi dengan database Amazon DocumentDB. Amazon DocumentDB tidak dapat menggunakan grup keamanan yang tidak memiliki satu aturan keluar dengan grup keamanan VPC dari database Amazon DocumentDB sebagai sumbernya. Amazon DocumentDB juga tidak dapat menggunakan grup keamanan yang telah dimodifikasi.	Tindakan Amazon DocumentDB: buat grup keamanan baru

Konfigurasi grup keamanan Amazon DocumentDB saat ini	Konfigurasi grup keamanan EC2 saat ini	Tindakan Amazon DocumentDB
Ada satu atau lebih grup keamanan yang terkait dengan database Amazon DocumentDB dengan nama yang cocok dengan polanya. `DocumentDB-ec2-n` Grup keamanan yang cocok dengan pola belum diubah. Grup keamanan ini memiliki hanya satu aturan masuk dengan grup keamanan VPC instans EC2 sebagai sumbernya.	Ada grup keamanan EC2 yang valid untuk koneksi, tetapi tidak terkait dengan instans EC2. Grup keamanan ini memiliki nama yang cocok dengan pola `DocumentDB-ec2-n`. Grup tersebut belum diubah. Ini hanya memiliki satu aturan keluar dengan grup keamanan VPC dari database Amazon DocumentDB sebagai sumbernya.	Tindakan Amazon DocumentDB: asosiasikan grup keamanan EC2

Konfigurasi grup keamanan Amazon DocumentDB saat ini	Konfigurasi grup keamanan EC2 saat ini	Tindakan Amazon DocumentDB
Salah satu syarat berikut berlaku: • Tidak ada grup keamanan yang terkait dengan database Amazon DocumentDB dengan nama yang cocok dengan polanya. `DocumentDB-ec2-n` • Ada satu atau lebih grup keamanan yang terkait dengan database Amazon DocumentDB dengan nama yang cocok dengan polanya. `DocumentDB-ec2-n` Namun, Amazon DocumentDB tidak dapat menggunakan salah satu grup keamanan ini untuk koneksi dengan instans EC2. Amazon DocumentDB tidak dapat menggunakan grup keamanan yang tidak memiliki satu aturan masuk dengan grup keamanan VPC dari instans EC2 sebagai sumbernya. Amazon DocumentDB juga tidak dapat menggunakan grup keamanan yang telah dimodifikasi.	Ada satu atau beberapa grup keamanan yang terkait dengan instans EC2 dengan nama yang cocok dengan pola `DocumentDB-ec2-n` . Grup keamanan yang cocok dengan pola belum diubah. Grup keamanan ini hanya memiliki satu aturan keluar dengan grup keamanan VPC dari database Amazon DocumentDB sebagai sumbernya.	Tindakan Amazon DocumentDB: buat grup keamanan baru

Tindakan Amazon DocumentDB: buat grup keamanan baru

Amazon DocumentDB mengambil tindakan berikut:

- Membuat grup keamanan baru yang cocok dengan pola `DocumentDB-ec2-n`. Grup keamanan ini memiliki aturan masuk dengan grup keamanan VPC instans EC2 sebagai sumbernya. Grup keamanan ini dikaitkan dengan database Amazon DocumentDB dan memungkinkan instans EC2 untuk mengakses database Amazon DocumentDB.
- Membuat grup keamanan baru yang cocok dengan pola `ec2-DocumentDB-n`. Grup keamanan ini memiliki aturan keluar dengan grup keamanan VPC dari database Amazon DocumentDB sebagai sumbernya. Grup keamanan ini dikaitkan dengan instans EC2 dan memungkinkan instans EC2 mengirim lalu lintas ke database Amazon DocumentDB.

Tindakan Amazon DocumentDB: asosiasikan grup keamanan EC2

Amazon DocumentDB mengaitkan grup keamanan EC2 yang valid dan sudah ada dengan instans EC2. Grup keamanan ini memungkinkan instans EC2 untuk mengirim lalu lintas ke database Amazon DocumentDB.

Melihat sumber daya komputasi terhubung

Anda dapat menggunakan Konsol Manajemen AWS untuk melihat sumber daya komputasi yang terhubung ke database Amazon DocumentDB. Sumber daya yang ditampilkan meliputi koneksi sumber daya komputasi yang disiapkan secara otomatis. Anda dapat menyiapkan secara otomatis konektivitas dengan sumber daya komputasi dengan cara berikut:

- Anda dapat memilih sumber daya komputasi saat membuat basis data. Untuk informasi selengkapnya, lihat Membuat cluster Amazon DocumentDB dan Membuat cluster Multi-AZ DB.
- Anda dapat menyiapkan konektivitas antara basis data yang ada dan sumber daya komputasi. Untuk informasi selengkapnya, lihat Hubungkan Amazon EC2 secara otomatis.

Sumber daya komputasi yang tercantum tidak menyertakan sumber daya yang dihubungkan secara manual dengan basis data. Misalnya, Anda dapat mengizinkan sumber daya komputasi untuk mengakses basis data secara manual dengan menambahkan aturan ke grup keamanan VPC yang terkait dengan basis data.

Agar sumber daya komputasi tercantum, syarat-syarat berikut harus dipenuhi:

- Nama grup keamanan yang terkait dengan sumber daya komputasi cocok dengan pola `ec2-DocumentDB-n` (di mana n adalah angka).
- Grup keamanan yang terkait dengan sumber daya komputasi memiliki aturan keluar dengan rentang port yang disetel ke port yang digunakan database Amazon DocumentDB.
- Grup keamanan yang terkait dengan sumber daya komputasi memiliki aturan keluar dengan sumber yang disetel ke grup keamanan yang terkait dengan database Amazon DocumentDB.
- Nama grup keamanan yang terkait dengan database Amazon DocumentDB cocok dengan `DocumentDB-ec2-n` pola (di mana n adalah angka).
- Grup keamanan yang terkait dengan database Amazon DocumentDB memiliki aturan masuk dengan rentang port yang disetel ke port yang digunakan database Amazon DocumentDB.
- Grup keamanan yang terkait dengan database Amazon DocumentDB memiliki aturan masuk dengan sumber yang disetel ke grup keamanan yang terkait dengan sumber daya komputasi.

Untuk melihat sumber daya komputasi yang terhubung ke database Amazon DocumentDB

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon DocumentDB di. https://console.aws.amazon.com/docdb
2. Di panel navigasi, pilih Databases, lalu pilih nama database Amazon DocumentDB.
3. Pada tab Konektivitas & keamanan, lihat sumber daya komputasi di bagian Sumber daya komputasi yang terhubung.

Hubungkan Amazon EC2 secara manual

Topik

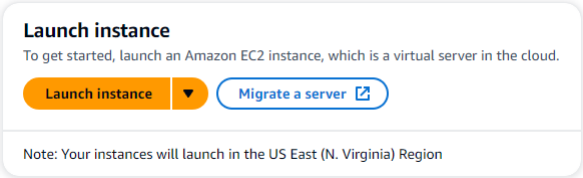
- Langkah 1: Buat instans Amazon EC2
- Langkah 2: Buat grup keamanan
- Langkah 3: Buat klaster Amazon DocumentDB
- Langkah 4: Hubungkan ke instans Amazon EC2 Anda
- Langkah 5: Instal MongoDB Shell
- Langkah 6: Kelola Amazon DocumentDB TLS
- Langkah 7: Hubungkan ke klaster Amazon DocumentDB Anda
- Langkah 8: Masukkan dan kueri data
- Langkah 9: Jelajahi

Langkah-langkah berikut mengasumsikan Anda telah menyelesaikan langkah-langkah dalam Prasyarat topik.

Langkah 1: Buat instans Amazon EC2

Pada langkah ini, Anda akan membuat instans Amazon EC2 di Wilayah yang sama dan Amazon VPC yang nantinya akan Anda gunakan untuk menyediakan klaster Amazon DocumentDB Anda.

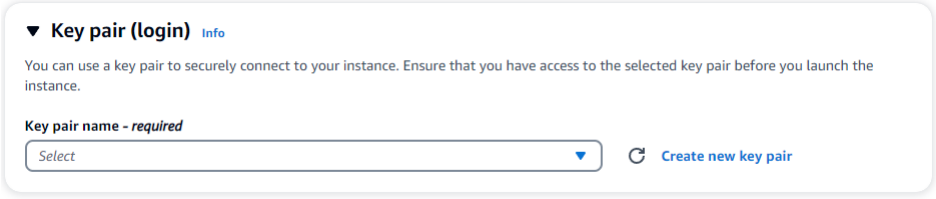
1. Pada konsol Amazon EC2, pilih Luncurkan instans.



2. Masukkan nama atau pengenal di bidang Nama yang terletak di bagian Nama dan tag.
3. Di daftar drop-down Amazon Machine Image (AMI), cari Amazon Linux 2 AMI dan pilih.



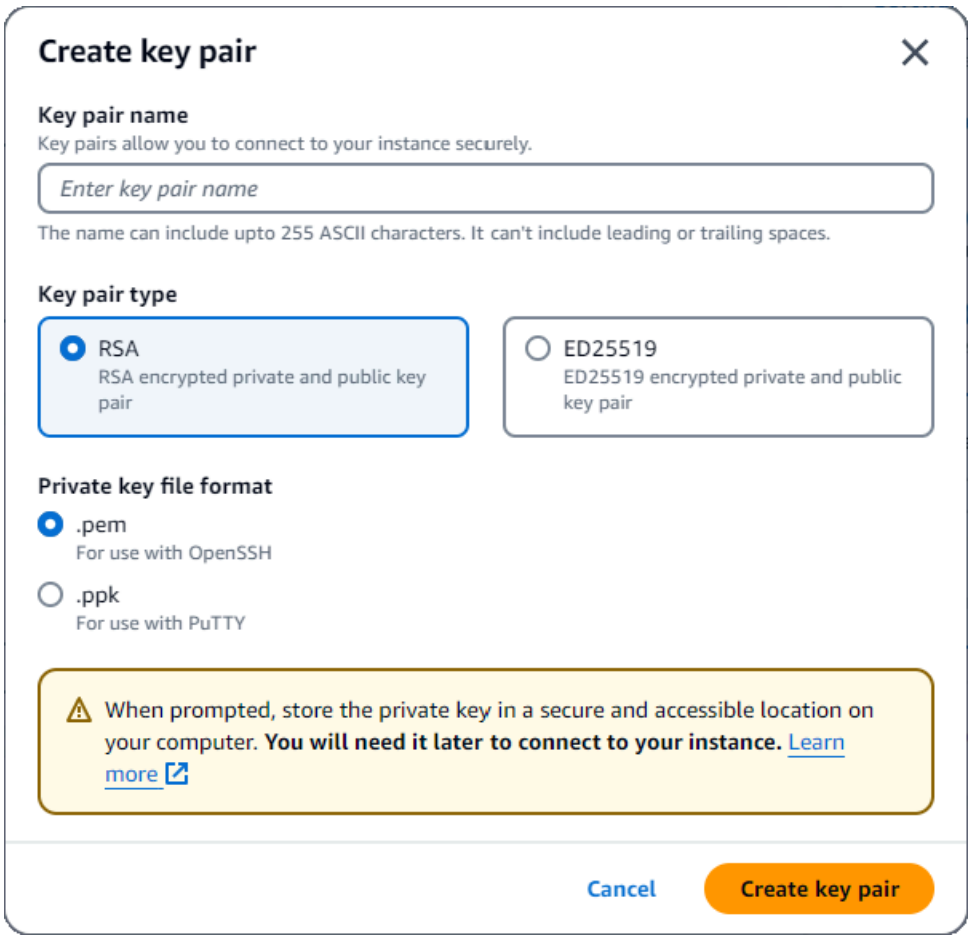
4. Cari dan pilih t3.micro dalam daftar drop-down tipe Instance.
5. Di bagian Key pair (login), masukkan identifier dari key-pair yang ada, atau pilih Create new key pair.



   Anda harus menyediakan pasangan kunci Amazon EC2.

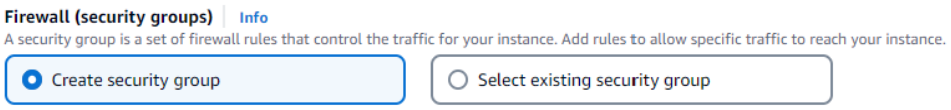
   - Jika Anda memiliki pasangan kunci Amazon EC2:
     1. Pilih pasangan kunci, pilih pasangan kunci Anda dari daftar.
     2. Anda harus sudah memiliki file kunci pribadi (file.pem atau .ppk) yang tersedia untuk masuk ke instans Amazon EC2 Anda.
   - Jika Anda tidak memiliki pasangan kunci Amazon EC2:
     1. Pilih Create new key pair, kotak dialog Create key pair muncul.
     2. Masukkan nama di bidang Nama pasangan kunci.
     3. Pilih jenis pasangan kunci dan format file kunci pribadi.

4. Pilih Buat pasangan kunci.



> **Note**
>
> Untuk tujuan keamanan, kami sangat menyarankan menggunakan pasangan kunci untuk konektivitas SSH dan internet ke instans EC2 Anda.

6. Di bagian Pengaturan jaringan, di bawah Firewall (grup keamanan), pilih Buat grup keamanan atau Pilih grup keamanan yang ada.



Jika Anda memilih untuk memilih grup keamanan yang ada, pilih salah satu dari daftar drop-down grup keamanan umum.

Jika Anda memilih untuk membuat grup keamanan baru, lakukan hal berikut:

a. Periksa semua aturan izin lalu lintas yang berlaku untuk konektivitas EC2 Anda.

b. Di bidang IP, pilih IP Saya atau pilih Kustom untuk memilih dari daftar blok CIDR, daftar awalan, atau grup keamanan. Kami tidak merekomendasikan Anywhere sebagai pilihan, kecuali instans EC2 Anda berada di jaringan yang terisolasi, karena memungkinkan akses alamat IP apa pun ke instans EC2 Anda.

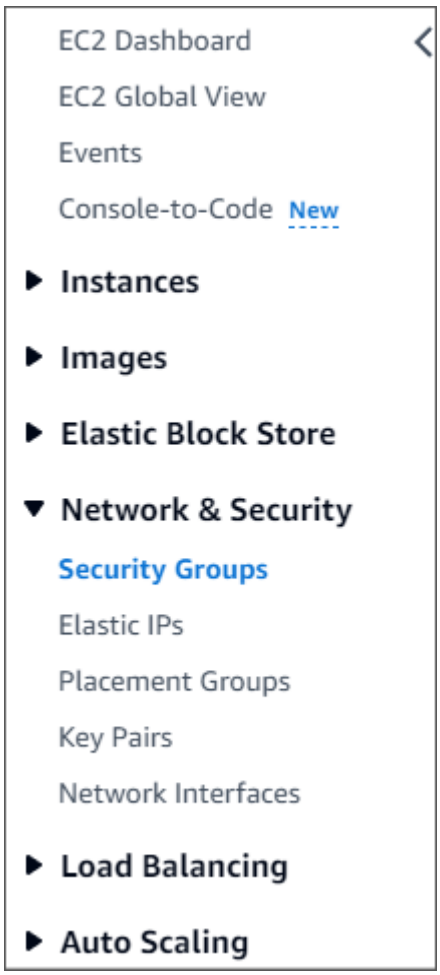


7. Di bagian Ringkasan, tinjau konfigurasi EC2 Anda dan pilih Launch instance jika benar.

Langkah 2: Buat grup keamanan

Anda sekarang akan membuat grup keamanan baru di Amazon VPC default Anda. Grup keamanan `demoDocDB` memungkinkan Anda untuk terhubung ke klaster Amazon DocumentDB Anda pada port 27017 (port default untuk Amazon DocumentDB) dari instans Amazon EC2 Anda.

1. Di Konsol Manajemen Amazon EC2, di bawah Jaringan dan Keamanan, pilih Grup keamanan.



2. Pilih Buat grup keamanan.



3. Di bagian Detail dasar:

   a. Untuk Nama grup keamanan, masukkan `demoDocDB`.

   b. Untuk Deskripsi, masukkan deskripsi.

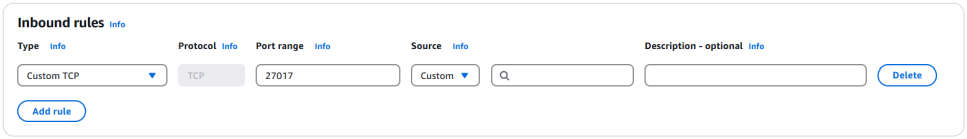
   c. Untuk VPC, terima penggunaan VPC default Anda.

4. Di bagian Aturan masuk, pilih Tambahkan aturan.

   a. Untuk Jenis, pilih Aturan TCP Kustom (default).

   b. Untuk Rentang port, masukkan `27017`.

   c. Untuk Sumber, pilih Kustom. Di bidang di sebelahnya, cari grup keamanan yang baru saja Anda buat di langkah 1. Anda mungkin perlu untuk me-refresh peramban Anda bagi konsol Amazon EC2 untuk mengisi otomatis nama sumber .



5. Terima semua default lainnya dan pilih Buat grup keamanan.



Langkah 3: Buat klaster Amazon DocumentDB

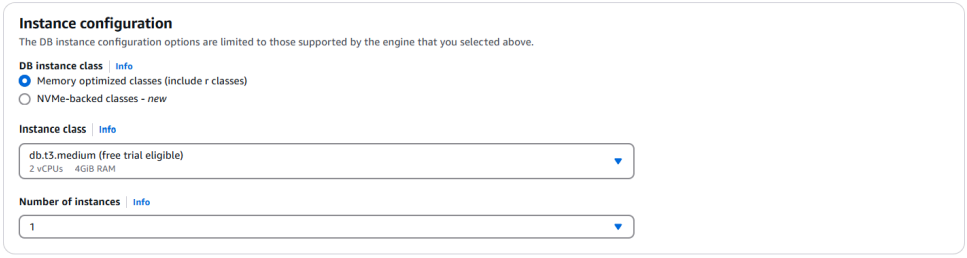
Saat instans Amazon EC2 sedang disediakan, Anda akan membuat klaster Amazon DocumentDB Anda.

1. Navigasikan ke konsol Amazon DocumentDB dan pilih Klaster dari panel navigasi.
2. Pilih Buat.
3. Biarkan pengaturan tipe Cluster pada defaultnya dari Instance Based Cluster.
4. Dalam konfigurasi Cluster, untuk pengidentifikasi Cluster, masukkan nama unik. Perhatikan bahwa konsol akan mengubah semua nama cluster menjadi huruf kecil terlepas dari bagaimana mereka dimasukkan.

   Biarkan versi Engine pada nilai defaultnya 5.0.0.
5. Untuk konfigurasi penyimpanan Cluster, tinggalkan pengaturan default Amazon DocumentDB Standard.
6. Dalam konfigurasi Instance:

   - Untuk kelas instans DB, pilih kelas yang dioptimalkan memori (termasuk kelas r) (ini adalah default).

     Pilihan contoh lainnya adalah NVMe-backed kelas. Untuk mempelajari selengkapnya, lihat NVMe-backed contoh.
   - Untuk kelas Instance, pilih jenis instance yang sesuai dengan kebutuhan Anda. Untuk penjelasan lebih rinci tentang kelas instance, lihatSpesifikasi kelas instans.
   - Untuk jumlah contoh, pilih nomor yang paling mencerminkan kebutuhan Anda. Ingat, semakin rendah angkanya, semakin rendah biayanya, dan semakin rendah read/write volume yang dapat dikelola oleh cluster.



7. Untuk Konektivitas, biarkan pengaturan default Jangan sambungkan ke sumber daya komputasi EC2.

> **Note**
>
> Menyambung ke sumber daya komputasi EC2 secara otomatis membuat grup keamanan untuk koneksi Anda ke klaster Anda. Karena Anda membuat grup keamanan ini secara manual pada langkah sebelumnya, Anda harus memilih Jangan sambungkan ke sumber daya komputasi EC2 agar tidak membuat kumpulan grup keamanan kedua.

8. Di bagian Autentikasi, masukkan nama pengguna untuk pengguna utama, lalu pilih Self managed. Masukkan kata sandi, lalu konfirmasikan.

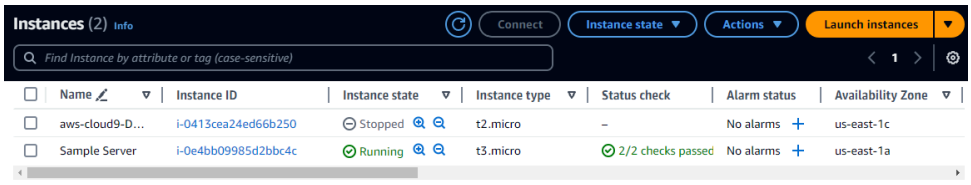
   Jika Anda memilih Managed in AWS Secrets Manager, lihat Manajemen kata sandi dengan Amazon DocumentDB dan AWS Secrets Manager untuk informasi selengkapnya.

9. Pilih Buat klaster.

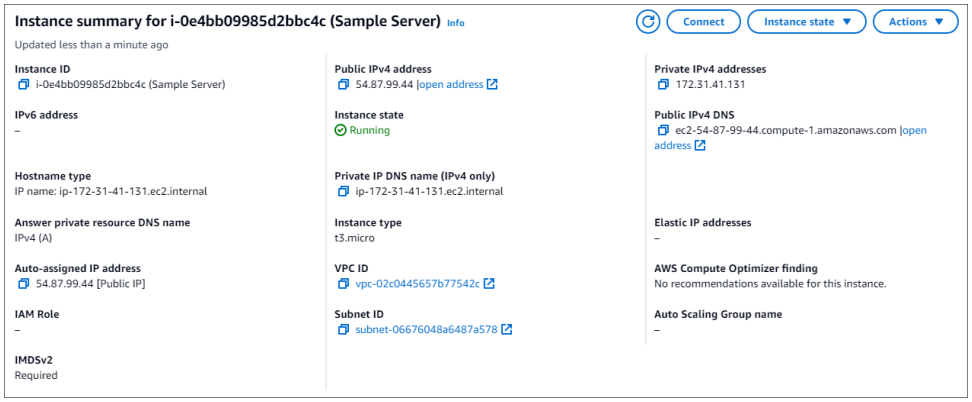
Langkah 4: Hubungkan ke instans Amazon EC2 Anda

Menghubungkan ke instans Amazon EC2 Anda akan memungkinkan Anda menginstal shell MongoDB. Instalasi shell mongo memungkinkan Anda untuk terhubung ke dan kueri klaster Amazon DocumentDB Anda. Selesaikan langkah-langkah berikut:

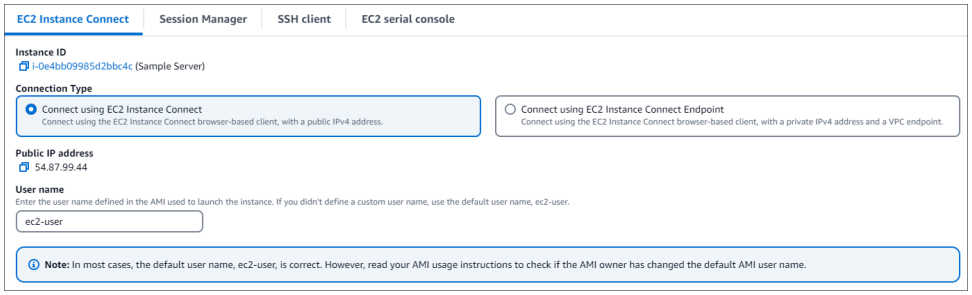
1. Pada konsol Amazon EC2, navigasikan ke instans Anda dan lihat apakah instans yang baru saja Anda buat berjalan. Jika ya, pilih instance dengan mengklik ID instance.



2. Pilih Hubungkan.



3. Ada empat opsi tab untuk metode koneksi Anda: Amazon EC2 Instance Connect, Session Manager, SSH client, atau konsol serial EC2. Anda harus memilih satu dan mengikuti instruksinya. Setelah selesai, pilih Connect.



> **Note**
>
> Jika alamat IP Anda berubah setelah Anda memulai penelusuran ini, atau Anda kembali ke lingkungan Anda di lain waktu, Anda harus memperbarui aturan masuk grup `demoEC2` keamanan Anda untuk mengaktifkan lalu lintas masuk dari alamat API baru Anda.

Langkah 5: Instal MongoDB Shell

Anda sekarang dapat menginstal shell MongoDB, yang merupakan utilitas baris perintah yang Anda gunakan untuk menghubungkan dan menanyakan cluster Amazon DocumentDB Anda. Saat ini ada dua versi shell MongoDB: versi terbaru, mongosh, dan versi sebelumnya, mongo shell.

> **Important**
>
> Ada batasan yang diketahui dengan Node.js driver yang lebih lama dari versi 6.13.1, yang saat ini tidak didukung oleh otentikasi identitas IAM untuk Amazon DocumentDB. Node.js

driver dan alat yang menggunakan Node.js driver (misalnya, mongosh) harus ditingkatkan untuk menggunakan Node.js driver versi 6.13.1 atau lebih tinggi.

Ikuti instruksi di bawah ini untuk menginstal MongoDB shell untuk sistem operasi Anda.

On Amazon Linux

Untuk menginstal shell MongoDB di Amazon Linux

Jika Anda tidak menggunakan IAM dan ingin menggunakan shell MongoDB terbaru (mongosh) untuk terhubung ke cluster Amazon DocumentDB Anda, ikuti langkah-langkah berikut:

1. Buat file repositori. Pada baris perintah instans EC2 yang Anda buat, jalankan perintah berikut:

```
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. Setelah selesai, instal mongosh dengan salah satu dari dua opsi perintah berikut di prompt perintah:

   Opsi 1 - Jika Anda memilih default Amazon Linux 2023 selama konfigurasi Amazon EC2, masukkan perintah ini:

   ```
   sudo yum install -y mongodb-mongosh-shared-openssl3
   ```

   Opsi 2 - Jika Anda memilih Amazon Linux 2 selama konfigurasi Amazon EC2, masukkan perintah ini:

   ```
   sudo yum install -y mongodb-mongosh
   ```

Jika Anda menggunakan IAM, Anda harus menggunakan shell MongoDB versi sebelumnya (5.0) untuk terhubung ke cluster Amazon DocumentDB Anda, ikuti langkah-langkah ini:

1. Buat file repositori. Pada baris perintah instans EC2 yang Anda buat, jalankan perintah berikut:

```
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. Setelah selesai, instal shell mongodb 5.0 dengan opsi perintah berikut di prompt perintah:

```
sudo yum install -y mongodb-org-shell
```

On Ubuntu

Untuk menginstal mongosh di Ubuntu

1. Impor kunci publik yang akan digunakan oleh sistem manajemen paket.

```
curl -fsSL https://pgp.mongodb.com/server-5.0.asc | sudo gpg --dearmor -o /usr/
share/keyrings/mongodb-server-5.0.gpg
```

2. Buat file daftar `mongodb-org-5.0.list` untuk MongoDB menggunakan perintah yang sesuai untuk versi Ubuntu Anda.

```
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. Impor dan perbarui database paket lokal menggunakan perintah berikut:

```
sudo apt-get update
```

4. Instal mongosh.

```
sudo apt-get install -y mongodb-mongosh
```

Untuk informasi tentang menginstal versi sebelumnya dari MongoDB pada sistem Ubuntu Anda, lihat Instal MongoDB Community Edition di Ubuntu.

On other operating systems

Untuk menginstal shell mongo pada sistem operasi lain, lihat Instal MongoDB Community Edition dalam dokumentasi MongoDB.

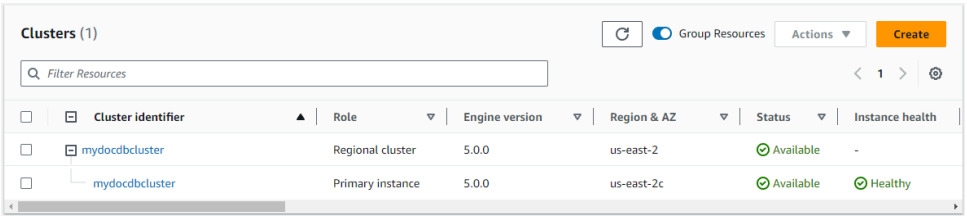
Langkah 6: Kelola Amazon DocumentDB TLS

Unduh sertifikat CA untuk Amazon DocumentDB dengan kode berikut: `wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem`

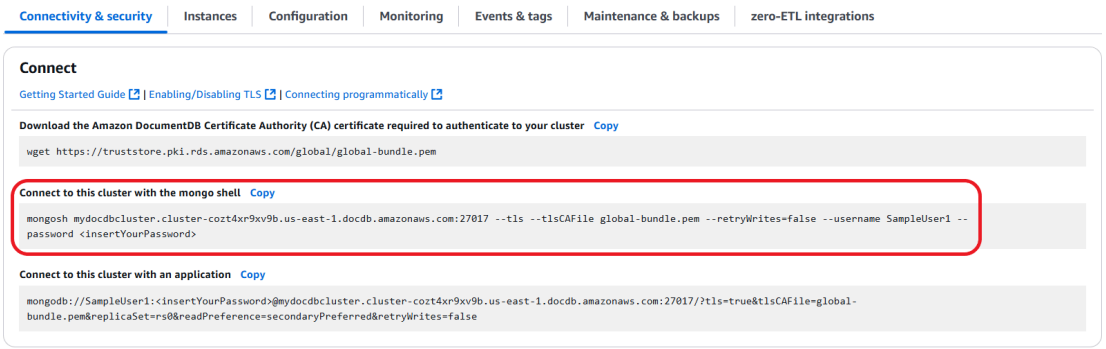
> **Note**
>
> Keamanan Lapisan Pengangkutan (TLS) diaktifkan secara default untuk setiap klaster Amazon DocumentDB baru. Untuk informasi selengkapnya, lihat Mengelola setelan TLS cluster Amazon DocumentDB.

Langkah 7: Hubungkan ke klaster Amazon DocumentDB Anda

1. Di konsol Amazon DocumentDB, di bawah Clusters, temukan klaster Anda. Pilih cluster yang Anda buat dengan mengklik pengenal Cluster untuk cluster tersebut.



2. Di tab Konektivitas dan keamanan, cari Connect to this cluster dengan shell mongo di kotak Connect:



Salin string koneksi yang disediakan dan tempel ke terminal Anda.

Buat perubahan berikut untuk itu:

a. Pastikan Anda memiliki nama pengguna yang benar dalam string.

b. Hilangkan `<insertYourPassword>` sehingga Anda diminta kata sandi oleh shell mongo saat Anda terhubung.

c. Opsional: Jika Anda menggunakan otentikasi IAM, atau menggunakan shell MongoDB versi sebelumnya, ubah string koneksi Anda sebagai berikut:

```
mongo --ssl --host mydocdbcluster.cluster-cozt4xr9xv9b.us-
east-1.docdb.amazonaws.com:27017 --sslCAFile global-bundle.pem --
username SampleUser1 --password
```

Ganti `mydocdbcluster.cluster-cozt4xr9xv9b.us-east-1` dengan informasi yang sama dari cluster Anda.

3. Tekan enter di terminal Anda. Anda sekarang diminta untuk kata sandi Anda. Masukkan kata sandi Anda.
4. Saat Anda memasukkan kata sandi dan dapat melihat perintah `rs0 [direct: primary] <env-name>>`, Anda berhasil terhubung ke klaster Amazon DocumentDB Anda.

Mengalami masalah saat menghubungkan? Lihat Pemecahan Masalah Amazon DocumentDB.

Langkah 8: Masukkan dan kueri data

Sekarang setelah Anda terhubung ke klaster, Anda dapat menjalankan beberapa kueri untuk mengenali penggunaan basis data dokumen.

1. Untuk memasukkan satu dokumen, masukkan hal berikut ini:

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

Anda mendapatkan output berikut ini:

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

2. Anda dapat membaca dokumen yang Anda tulis dengan perintah `findOne()` (karena hanya mengembalikan satu dokumen). Masukkan yang berikut ini:

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

Anda mendapatkan output berikut ini:

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

3. Untuk mengajukan beberapa kueri lagi, pertimbangkan kasus penggunaan profil game. Pertama, masukkan beberapa entri ke dalam koleksi berjudul `profiles`. Masukkan yang berikut ini:

```
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 }
])
```

Anda mendapatkan output berikut ini:

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

4. Gunakan perintah `find()` untuk mengembalikan semua dokumen dalam koleksi profil. Masukkan berikut ini:

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

Anda akan mendapatkan output yang akan cocok dengan data yang Anda ketik di Langkah 3.

5. Gunakan kueri untuk satu dokumen dengan menggunakan filter. Masukkan yang berikut ini:

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

Anda mendapatkan output berikut ini:

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

6. Sekarang mari kita coba untuk menemukan profil dan memodifikasinya dengan menggunakan perintah `findAndModify`. Kami akan memberi pengguna Matt tambahan 10 poin dengan kode berikut:

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

Anda mendapatkan output berikut (perhatikan bahwa skornya belum meningkat):

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

7. Anda dapat memverifikasi bahwa skornya telah berubah dengan kueri berikut:

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

Anda mendapatkan output berikut ini:

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

Langkah 9: Jelajahi

Selamat! Anda telah berhasil menyelesaikan Panduan Quick Start untuk Amazon DocumentDB.

Apa selanjutnya? Pelajari cara memanfaatkan sepenuhnya basis data yang kuat ini dengan beberapa fitur populernya:

- Mengelola Amazon DocumentDB
- Penskalaan
- Mencadangkan dan memulihkan

> **Note**
>
> Untuk menghemat biaya, Anda dapat menghentikan klaster Amazon DocumentDB untuk mengurangi biaya atau menghapus klaster. Secara default, setelah 30 menit tidak aktif, AWS Cloud9 lingkungan Anda akan menghentikan instans Amazon EC2 yang mendasarinya.

Connect menggunakan driver Amazon DocumentDB JDBC

Driver JDBC untuk Amazon DocumentDB menyediakan SQL-relational antarmuka untuk pengembang dan memungkinkan konektivitas dari alat BI seperti Tableau dan. DbVisualizer

Untuk informasi lebih rinci, lihat dokumentasi Driver Amazon DocumentDB JDBC di. GitHub

Topik

- Memulai
- Connect ke Amazon DocumentDB dari Tableau Desktop
- Connect ke Amazon DocumentDB dari DbVisualizer
- Pembuatan skema otomatis JDBC
- Dukungan dan keterbatasan SQL
- Pemecahan masalah

Memulai

Langkah 1. Buat Cluster Amazon DocumentDB

Jika Anda tidak memiliki kluster Amazon DocumentDB yang dibuat, maka buat cluster menggunakan petunjuk di bagian Memulai Panduan Pengembang Amazon DocumentDB.

> **Note**
>
> Amazon DocumentDB adalah layanan Virtual Private Cloud (VPC) saja. Jika Anda terhubung dari mesin lokal, di luar VPC cluster, Anda harus membuat koneksi SSH ke instans Amazon EC2. Dalam hal ini, luncurkan cluster Anda menggunakan instruksi di Connect with EC2. Lihat Menggunakan Terowongan SSH untuk Menyambung ke Amazon DocumentDB untuk informasi selengkapnya tentang tunneling SSH dan kapan Anda mungkin membutuhkannya.

Langkah 2. Instalasi JRE atau JDK

Tergantung pada aplikasi BI Anda, Anda mungkin perlu memastikan instalasi JRE atau JDK 64-bit versi 8 atau yang lebih baru diinstal pada komputer Anda. Anda dapat mengunduh Java SE Runtime Environment 8 di sini.

Langkah 3. Unduh Driver DocumentDB JDBC

Unduh driver DocumentDB JDBC dari sini. Driver dikemas sebagai file JAR tunggal (misalnya documentdb-jdbc-1.0.0-all.jar).

Langkah 4. Menggunakan Terowongan SSH untuk Connect ke Amazon DocumentDB

Klaster Amazon DocumentDB (dengan kompatibilitas MongoDB) di-deploy dalam Amazon Virtual Private Cloud (Amazon VPC). Mereka dapat diakses langsung oleh instans Amazon EC2 atau AWS layanan lain yang digunakan di VPC Amazon yang sama. Selain itu, Amazon DocumentDB dapat diakses oleh instans EC2a atau layanan AWS lain di VPC yang berbeda di Wilayah yang sama atau AWS Wilayah lain melalui peering VPC.

Anda dapat menggunakan SSH tunneling (juga dikenal sebagai port forwarding) untuk mengakses sumber daya Amazon DocumentDB Anda, dari luar VPC cluster. Ini akan menjadi kasus bagi sebagian besar pengguna yang tidak menjalankan aplikasi mereka pada VM di VPC yang sama dengan cluster DocumentDB.

Untuk membuat terowongan SSH, Anda memerlukan instans Amazon EC2 yang berjalan di Amazon VPC yang sama dengan klaster Amazon DocumentDB Anda. Anda dapat menggunakan instans EC2 yang ada di VPC yang sama sebagai klaster Anda atau membuatnya. Anda dapat mengatur terowongan SSH ke cluster Amazon `sample-cluster.node.us-east-1.docdb.amazonaws.com` DocumentDB dengan menjalankan perintah berikut di komputer lokal Anda.

```
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
```

Bendera -L digunakan untuk meneruskan port lokal. Ini adalah prasyarat untuk menghubungkan ke alat BI apa pun yang berjalan pada klien di luar VPC Anda. Setelah Anda menjalankan langkah di atas, Anda dapat melanjutkan ke langkah berikutnya untuk alat BI pilihan Anda.

Untuk informasi lebih lanjut tentang SSH tunneling, lihat dokumentasi tentang Menggunakan terowongan SSH untuk terhubung ke Amazon DocumentDB.

Connect ke Amazon DocumentDB dari Tableau Desktop

Topik

- Menambahkan Driver Amazon DocumentDB JDBC
- Menghubungkan ke Amazon DocumentDB menggunakan Tableau - SSH Tunnel

Menambahkan Driver Amazon DocumentDB JDBC

Untuk terhubung ke Amazon DocumentDB dari Tableau Desktop Anda harus mengunduh dan menginstal driver Amazon DocumentDB JDBC dan konektor DocumentDB Tableau.

1. Unduh file JAR driver Amazon DocumentDB JDBC dari repositori Driver Amazon DocumentDB JDBC dan salin ke salah satu direktori ini sesuai dengan sistem operasi Anda:
   - Jendela - `C:\Program Files\Tableau\Drivers`
   - macOS - `~/Library/Tableau/Drivers`
2. Unduh konektor DocumentDB Tableau (file TACO) dari situs web Tableau Exchange dan salin ke direktori My Tableau Anda. Repository/Connectors
   - Jendela - `C:\Users\[user]\Documents\My Tableau Repository\Connectors`
   - macOS - `/Users/[user]/Documents/My Tableau Repository/Connectors`

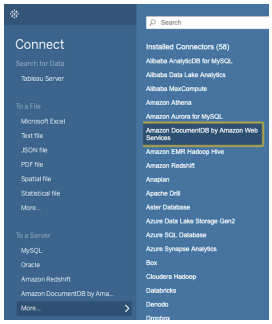
Untuk informasi tambahan, lihat dokumentasi Tableau.

> **Note**
>
> Jika Anda menggunakan sertifikat CA yang lebih baru, pastikan untuk meningkatkan driver JDBC Anda ke v1.4.5 (tersedia di repositori ini AWS GitHub .).

Menghubungkan ke Amazon DocumentDB menggunakan Tableau - SSH Tunnel

Untuk terhubung ke Tableau dari mesin klien di luar VPC cluster DocumentDB Anda, Anda harus menyiapkan terowongan SSH sebelum mengikuti langkah-langkah di bawah ini:

1. Luncurkan aplikasi Tableau Desktop.
2. Arahkan ke Connect > To A Server > More.
3. Pilih Amazon DocumentDB oleh Amazon Web Services di bawah Konektor Terpasang.

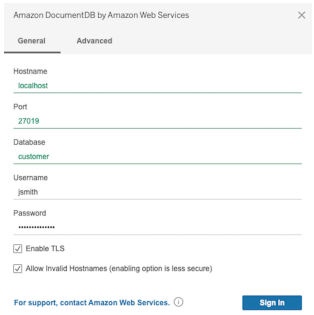


Menghubungkan ke Amazon DocumentDB Menggunakan Tableau - Terowongan SSH Eksternal

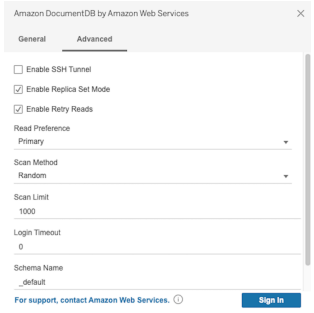
1. Masukkan parameter koneksi yang diperlukan Nama host, Port, Database, Nama Pengguna dan Kata Sandi. Parameter koneksi dalam contoh berikut setara dengan string koneksi JDBC:

   ```
   jdbc:documentdb://localhost:27019/test?
   tls=true&tlsAllowInvalidHostnames=true&scanMethod=random&scanLimit=1000&logir
   ```

   parameter nama pengguna dan kata sandi diteruskan secara terpisah dalam koleksi properti. Untuk informasi selengkapnya tentang parameter string koneksi, lihat dokumentasi github Amazon DocumentDB JDBC Driver.



2. (Opsional) Opsi yang lebih canggih dapat ditemukan di Advanced tab.

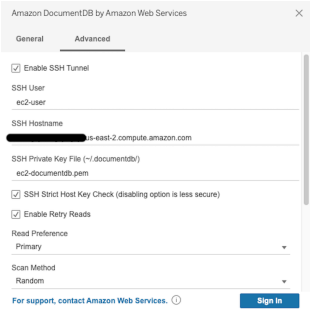


3. Pilih Masuk.

Menghubungkan ke Amazon DocumentDB Menggunakan Tableau - Terowongan SSH Internal

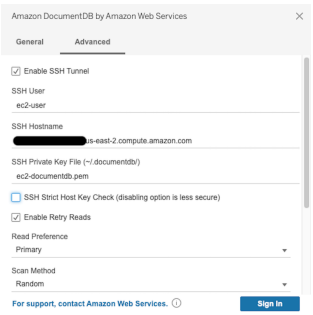
> **Note**
>
> Jika Anda memilih untuk tidak mengatur terowongan SSH menggunakan terminal, Anda dapat menggunakan GUI Tableau untuk menentukan detail instans EC2 Anda yang secara inheren akan digunakan driver JDBC untuk membuat terowongan SSH.

1. Pada tab Advanced, pilih opsi Enable SSH Tunnel untuk meninjau properti lebih lanjut.



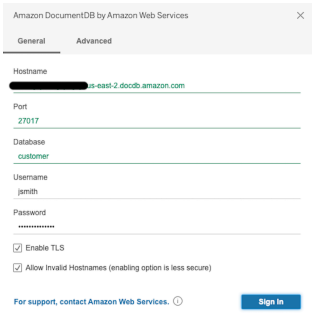
2. Masukkan SSH User, SSH Hostname, dan SSH Private Key File.
3. (Opsional) Anda dapat menonaktifkan opsi Pemeriksaan Kunci Host Ketat SSH yang melewati pemeriksaan kunci host terhadap file host yang dikenal.

> **Note**
>
> Menonaktifkan opsi ini kurang aman karena dapat menyebabkan serangan man-in-the-middle.



4. Masukkan parameter yang diperlukan; Nama host, Port, Database, Nama Pengguna dan Kata Sandi.

> **Note**
>
> Pastikan Anda menggunakan endpoint cluster DocumentDB dan bukan localhost saat menggunakan opsi terowongan SSH internal.



5. Pilih Masuk.

Connect ke Amazon DocumentDB dari DbVisualizer

Topik

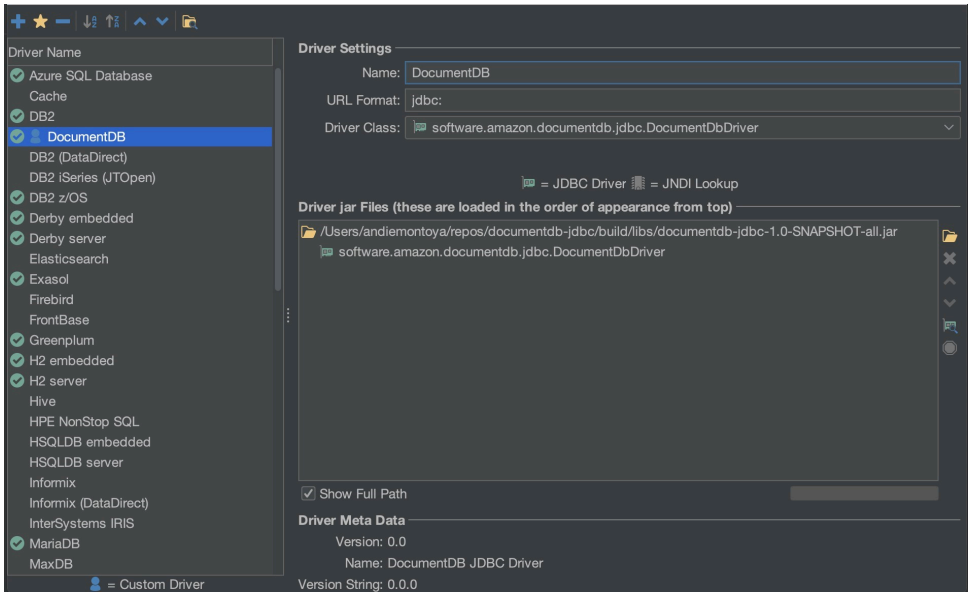
- Menambahkan driver Amazon DocumentDB JDBC
- Menghubungkan ke Amazon DocumentDB menggunakan DbVisualizer

Menambahkan driver Amazon DocumentDB JDBC

Untuk terhubung ke Amazon DocumentDB DbVisualizer dari Anda harus terlebih dahulu mengimpor Driver Amazon DocumentDB JDBC

1. Mulai DbVisualizer aplikasi dan arahkan ke jalur menu: Tools > Driver Manager...
2. Pilih + (atau di menu, pilih Driver > Create Driver).
3. Setel Nama ke`DocumentDB`.
4. Setel Format URL ke `jdbc:documentdb://<host>[:port]/<database>[?option=value[&option=value[...]]]`
5. Pilih tombol folder dan kemudian pilih file JAR driver Amazon DocumentDB JDBC dan pilih tombol Buka.

6. Verifikasi bahwa bidang Kelas Driver diatur `kesoftware.amazon.documentdb.jdbc.DocumentDbDriver`. Pengaturan Driver Manager Anda untuk DocumentDB akan terlihat seperti contoh berikut.



7. Tutup dialog. Driver Amazon DocumentDB JDBC akan disiapkan dan siap digunakan.

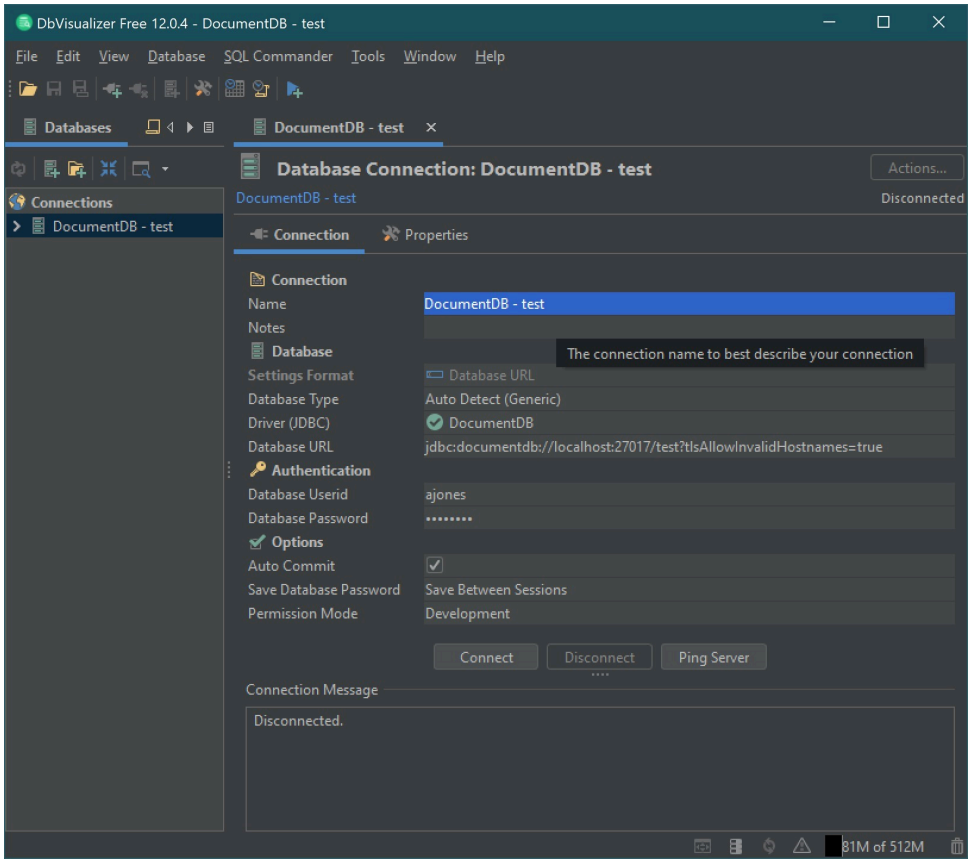
Menghubungkan ke Amazon DocumentDB menggunakan DbVisualizer

Connect ke Amazon DocumentDB Menggunakan DbVisualizer

1. Jika Anda terhubung dari luar VPC cluster Amazon DocumentDB, pastikan Anda telah menyiapkan terowongan SSH.
2. Pilih Database > Buat Koneksi Database dari menu tingkat atas.
3. Masukkan nama deskriptif untuk bidang Nama.
4. Setel Driver (JDBC) ke driver DocumentDB yang Anda buat di bagian sebelumnya.
5. Setel URL Database ke string koneksi JDBC Anda.

   Misalnya: `jdbc:documentdb://localhost:27017/database?tlsAllowInvalidHostnames=true`
6. Setel Userid Database ke ID pengguna Amazon DocumentDB Anda.
7. Atur Kata Sandi Database ke kata sandi yang sesuai untuk ID pengguna.

   Dialog Koneksi Database Anda akan terlihat seperti dialog berikut:



8. Pilih Hubungkan.

Pembuatan skema otomatis JDBC

Amazon DocumentDB adalah database dokumen dan karena itu tidak memiliki konsep tabel dan skema. Namun, alat BI seperti Tableau akan mengharapkan database yang terhubung untuk menyajikan skema. Secara khusus, ketika koneksi driver JDBC perlu mendapatkan skema untuk koleksi dalam database, itu akan polling untuk semua koleksi dalam database. Pengemudi akan menentukan apakah versi skema yang di-cache untuk koleksi itu sudah ada. Jika versi cache tidak ada, itu akan mengambil sampel koleksi untuk dokumen dan membuat skema berdasarkan perilaku berikut.

Topik

- Keterbatasan pembuatan skema
- Opsi metode pemindaian
- Tipe data Amazon DocumentDB
- Memetakan bidang dokumen skalar

- Penanganan tipe data objek dan array

Keterbatasan pembuatan skema

Driver DocumentDB JDBC memberlakukan batas pada panjang pengidentifikasi pada 128 karakter. Generator skema dapat memotong panjang pengidentifikasi yang dihasilkan (nama tabel dan nama kolom) untuk memastikan mereka sesuai dengan batas itu.

Opsi metode pemindaian

Perilaku pengambilan sampel dapat dimodifikasi menggunakan string koneksi atau opsi sumber data.

- ScanMethod= <option>
  - acak - (default) - Dokumen sampel dikembalikan dalam urutan acak.
  - IDForward - Contoh dokumen dikembalikan dalam urutan id.
  - IDReverse - Contoh dokumen dikembalikan dalam urutan terbalik id.
  - semua - Sampel semua dokumen dalam koleksi.
- ScanLimit= <n>- Jumlah dokumen untuk sampel. Nilai harus berupa bilangan bulat positif. Nilai defaultnya adalah 1000. Jika ScanMethod diatur ke semua, opsi ini diabaikan.

Tipe data Amazon DocumentDB

Server Amazon DocumentDB mendukung sejumlah tipe data MongoDB. Di bawah ini adalah tipe data yang didukung, dan tipe data JDBC yang terkait.

Tipe Data MongoDB	Didukung di DocumentDB	Tipe Data JDBC
Data Biner	Ya	VARBINARY
Boolean	Ya	BOOLEAN
Ganda	Ya	DOUBLE
Bilangan Bulat 32-bit	Ya	INTEGER
Integer 64-bit	Ya	BIGINT
Tali	Ya	VARCHAR

Tipe Data MongoDB	Didukung di DocumentDB	Tipe Data JDBC
ObjectId	Ya	VARCHAR
Date	Ya	TIMESTAMP
Null	Ya	VARCHAR
Ekspresi reguler	Ya	VARCHAR
Stempel waktu	Ya	VARCHAR
MinKey	Ya	VARCHAR
MaxKey	Ya	VARCHAR
Objek	Ya	meja virtual
Array	Ya	meja virtual
Decimal128	Tidak	DECIMAL
JavaScript	Tidak	VARCHAR
JavaScript (dengan ruang lingkup)	Tidak	VARCHAR
Tidak terdefinisi	Tidak	VARCHAR
Simbol	Tidak	VARCHAR
DBPointer (4.0+)	Tidak	VARCHAR

Memetakan bidang dokumen skalar

Saat memindai sampel dokumen dari koleksi, driver JDBC akan membuat satu atau lebih skema untuk mewakili sampel dalam koleksi. Secara umum, bidang skalar dalam dokumen memetakan ke kolom dalam skema tabel. Misalnya, dalam kumpulan bernama tim, dan satu dokumen`{ "_id" : "112233", "name" : "Alastair", "age": 25 }`, ini akan dipetakan ke skema:

Nama Tabel	Nama kolom	Tipe data	Key
team	id tim	VARCHAR	PK
team	name	VARCHAR	
team	usia	INTEGER	

Promosi konflik tipe data

Saat memindai dokumen sampel, ada kemungkinan bahwa tipe data untuk bidang tidak konsisten dari dokumen ke dokumen. Dalam hal ini, driver JDBC akan mempromosikan tipe data JDBC ke tipe data umum yang akan sesuai dengan semua tipe data dari dokumen sampel.

Sebagai Contoh:

```
{
"_id" : "112233",
"name" : "Alastair", "age" : 25
}

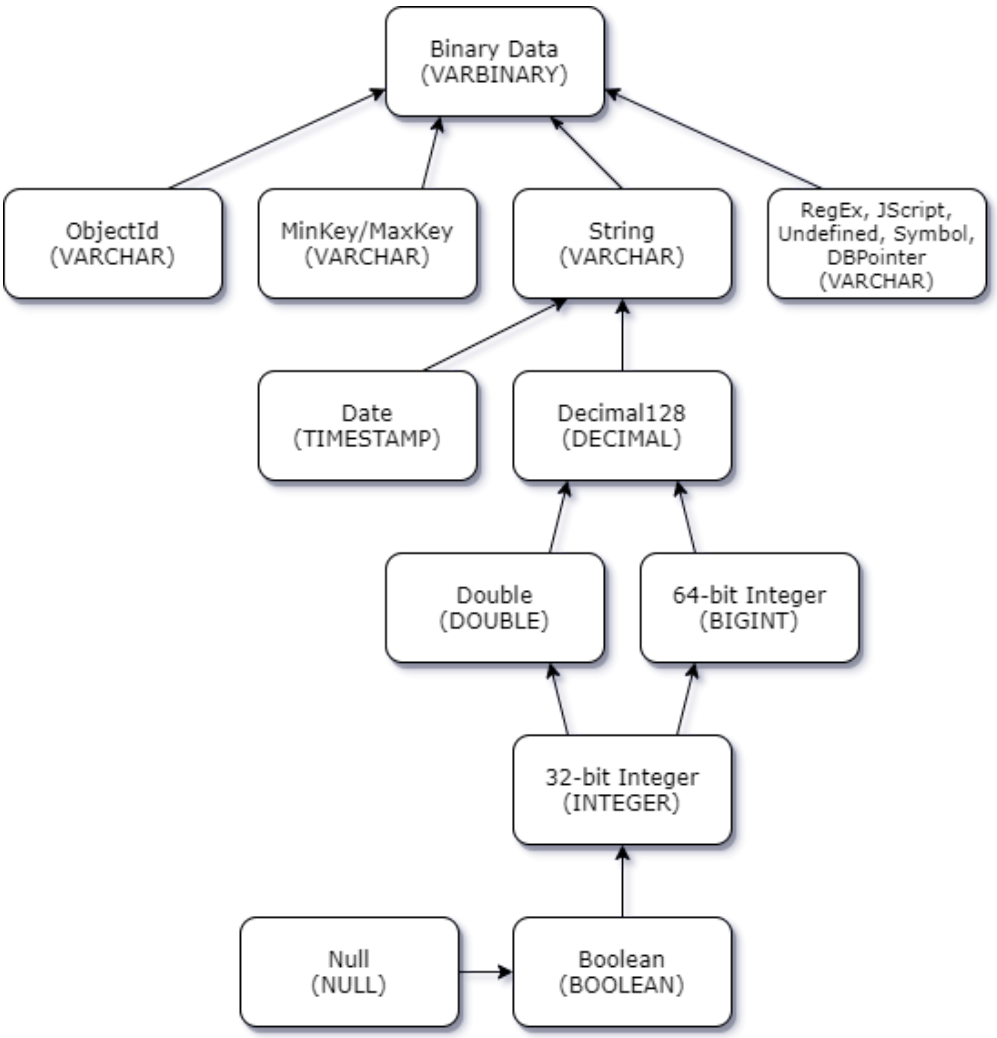
{
"_id" : "112244",
"name" : "Benjamin",
"age" : "32"
}
```

Bidang usia adalah tipe 32-bit integer dalam dokumen pertama tetapi string dalam dokumen kedua. Di sini driver JDBC akan mempromosikan tipe data JDBC ke VARCHAR untuk menangani salah satu tipe data saat ditemui.

Nama Tabel	Nama kolom	Tipe data	Key
team	id tim	VARCHAR	PK
team	name	VARCHAR	
team	usia	VARCHAR	

Scalar-scalar promosi konflik

Diagram berikut menunjukkan cara konflik tipe data skalar-skalar diselesaikan.



Scalar-complex jenis promosi konflik

Seperti konflik tipe skalar-skalar, bidang yang sama dalam dokumen yang berbeda dapat memiliki tipe data yang bertentangan antara kompleks (array dan objek) dan skalar (integer, boolean, dll.). Semua konflik ini diselesaikan (dipromosikan) ke VARCHAR untuk bidang tersebut. Dalam hal ini, array dan data objek dikembalikan sebagai representasi JSON.

Array Tertanam - Contoh Konflik Bidang String:

```
{
   "_id":"112233",
   "name":"George Jackson",
```

```
   "subscriptions":[
      "Vogue",
      "People",
      "USA Today"
   ]
}
{
   "_id":"112244",
   "name":"Joan Starr",
   "subscriptions":1
}
```

Contoh sebelumnya memetakan skema untuk tabel customer2:

Nama Tabel	Nama kolom	Tipe data	Key
pelanggan2	id pelanggan2	VARCHAR	PK
pelanggan2	name	VARCHAR	
pelanggan2	langganan	VARCHAR	

dan tabel virtual customer1_subscriptions:

Nama Tabel	Nama kolom	Tipe data	Key
customer1_subscriptions	id pelanggan1	VARCHAR	PK/FK
customer1_subscriptions	berlangganan_index_lvl0	BIGINT	PK
customer1_subscriptions	nilai	VARCHAR	
customer_address	kota	VARCHAR	
customer_address	region	VARCHAR	
customer_address	negeri	VARCHAR	

Nama Tabel	Nama kolom	Tipe data	Key
customer_address	code	VARCHAR	

Penanganan tipe data objek dan array

Sejauh ini, kami hanya menjelaskan bagaimana tipe data skalar dipetakan. Jenis data Object dan Array (saat ini) dipetakan ke tabel virtual. Driver JDBC akan membuat tabel virtual untuk mewakili bidang objek atau array dalam dokumen. Nama tabel virtual yang dipetakan akan menggabungkan nama koleksi asli diikuti dengan nama bidang yang dipisahkan oleh karakter garis bawah (“_”).

Kunci utama tabel dasar (“_id”) mengambil nama baru di tabel virtual baru dan disediakan sebagai kunci asing ke tabel dasar terkait.

Untuk bidang tipe array tertanam, kolom indeks dihasilkan untuk mewakili indeks ke dalam array di setiap tingkat array.

Contoh bidang objek tertanam

Untuk bidang objek dalam dokumen, pemetaan ke tabel virtual dibuat oleh 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"
   }
}
```

Contoh sebelumnya memetakan skema untuk tabel pelanggan:

Nama Tabel	Nama kolom	Tipe data	Key
pelanggan	id pelanggan	VARCHAR	PK

Nama Tabel	Nama kolom	Tipe data	Key
pelanggan	name	VARCHAR	

dan tabel virtual customer_address:

Nama Tabel	Nama kolom	Tipe data	Key
customer_address	id pelanggan	VARCHAR	PK/FK
customer_address	alamat1	VARCHAR	
customer_address	alamat2	VARCHAR	
customer_address	kota	VARCHAR	
customer_address	region	VARCHAR	
customer_address	negeri	VARCHAR	
customer_address	code	VARCHAR	

Contoh bidang array tertanam

Untuk bidang array dalam dokumen, pemetaan ke tabel virtual juga dibuat oleh driver JDBC.

```
{
   "Collection: customer1",
   "_id":"112233",
   "name":"George Jackson",
   "subscriptions":[
      "Vogue",
      "People",
      "USA Today"
   ]
}
```

Contoh sebelumnya memetakan skema untuk tabel pelanggan1:

Nama Tabel	Nama kolom	Tipe data	Key
pelanggan1	id pelanggan1	VARCHAR	PK
pelanggan1	name	VARCHAR	

dan tabel virtual customer1_subscriptions:

Nama Tabel	Nama kolom	Tipe data	Key
customer1_subscriptions	id pelanggan1	VARCHAR	PK/FK
customer1_subscriptions	berlangganan_index_lvl0	BIGINT	PK
customer1_subscriptions	nilai	VARCHAR	
customer_address	kota	VARCHAR	
customer_address	region	VARCHAR	
customer_address	negeri	VARCHAR	
customer_address	code	VARCHAR	

Dukungan dan keterbatasan SQL

Driver Amazon DocumentDB JDBC adalah driver read-only yang mendukung subset dan beberapa ekstensi umum. SQL-92 Lihat dokumentasi batasan SQL dan dokumentasi batasan JDBC untuk informasi lebih lanjut.

Pemecahan masalah

Jika Anda mengalami masalah saat menggunakan driver Amazon DocumentDB JDBC, lihat Panduan Pemecahan Masalah.

Connect menggunakan driver Amazon DocumentDB ODBC

Driver ODBC untuk Amazon DocumentDB SQL-relational menyediakan antarmuka untuk pengembang dan memungkinkan konektivitas dari alat BI seperti Power BI Desktop dan Microsoft Excel.

Untuk informasi lebih rinci, lihat dokumentasi Driver Amazon DocumentDB ODBC di. GitHub

Topik

- Memulai
- Menyiapkan driver Amazon DocumentDB ODBC di Windows
- Connect ke Amazon DocumentDB dari Microsoft Excel
- Connect ke Amazon DocumentDB dari Microsoft Power BI Desktop
- Pembuatan skema otomatis
- Dukungan dan keterbatasan SQL
- Pemecahan masalah

Memulai

Langkah 1. Buat Cluster Amazon DocumentDB

Jika Anda belum memiliki cluster Amazon DocumentDB, ada beberapa cara untuk memulai.

> **Note**
>
> Amazon DocumentDB adalah layanan Virtual Private Cloud (VPC) -only. Jika Anda terhubung dari mesin lokal di luar VPC cluster, Anda harus membuat koneksi SSH ke instans Amazon EC2. Dalam hal ini, luncurkan cluster Anda menggunakan instruksi di Connect with EC2. Lihat Menggunakan Terowongan SSH untuk Menyambung ke Amazon DocumentDB untuk informasi selengkapnya tentang tunneling SSH dan kapan Anda mungkin membutuhkannya.

Langkah 2. Instalasi JRE atau JDK

Tergantung pada aplikasi BI Anda, Anda mungkin perlu memastikan instalasi JRE atau JDK 64-bit versi 8 atau yang lebih baru diinstal pada komputer Anda. Anda dapat mengunduh Java SE Runtime Environment 8 di sini.

Langkah 3. Unduh Driver Amazon DocumentDB ODBC

Unduh driver Amazon DocumentDB ODBC di sini. Pilih installer yang tepat (misalnya, documentdb-odbc-1.0.0.msi). Ikuti panduan instalasi.

Langkah 4. Menggunakan Terowongan SSH untuk Connect ke Amazon DocumentDB

Cluster Amazon DocumentDB digunakan dalam Amazon Virtual Private Cloud (Amazon VPC). Mereka dapat diakses langsung oleh instans Amazon EC2 atau AWS layanan lain yang digunakan di VPC Amazon yang sama. Selain itu, Amazon DocumentDB dapat diakses oleh instans Amazon EC2 atau layanan AWS lain di berbagai VPC di wilayah yang sama atau AWS wilayah lain melalui peering VPC.

Namun, anggaplah kasus penggunaan Anda mengharuskan Anda (atau aplikasi Anda) mengakses sumber daya Amazon DocumentDB Anda dari luar VPC klaster. Ini akan menjadi kasus bagi sebagian besar pengguna yang tidak menjalankan aplikasi mereka pada VM di VPC yang sama dengan cluster Amazon DocumentDB. Saat menghubungkan dari luar VPC, Anda dapat menggunakan tunneling SSH (juga dikenal sebagai penerusan port) untuk mengakses sumber daya Amazon DocumentDB Anda.

Untuk membuat terowongan SSH, Anda memerlukan instans Amazon EC2 yang berjalan di Amazon VPC yang sama dengan klaster Amazon DocumentDB Anda. Anda dapat menggunakan instans EC2 yang ada di VPC yang sama sebagai klaster Anda atau membuatnya. Anda dapat mengatur terowongan SSH ke cluster Amazon `sample-cluster.node.us-east-1.docdb.amazonaws.com` DocumentDB dengan menjalankan perintah berikut di komputer lokal Anda:

```
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
```

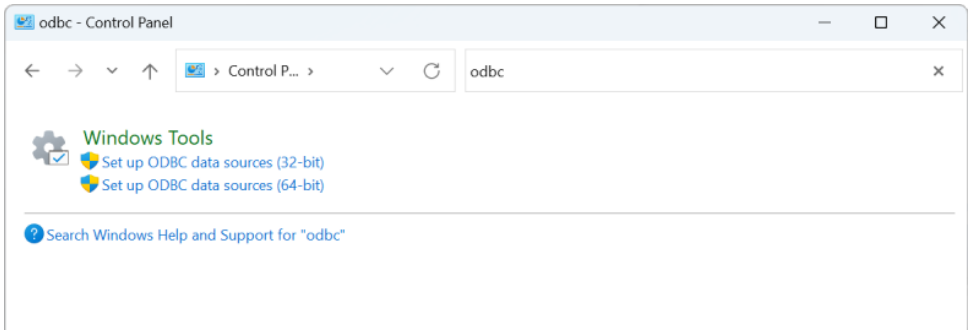
Bendera `-L` digunakan untuk meneruskan port lokal. Ini adalah prasyarat untuk menghubungkan ke alat BI apa pun yang berjalan pada klien di luar VPC Anda. Setelah Anda menjalankan langkah di atas, Anda dapat melanjutkan ke langkah berikutnya untuk alat BI pilihan Anda.

Untuk informasi lebih lanjut tentang SSH tunneling, lihat dokumentasi tentang Menggunakan Terowongan SSH untuk Terhubung ke Amazon DocumentDB.

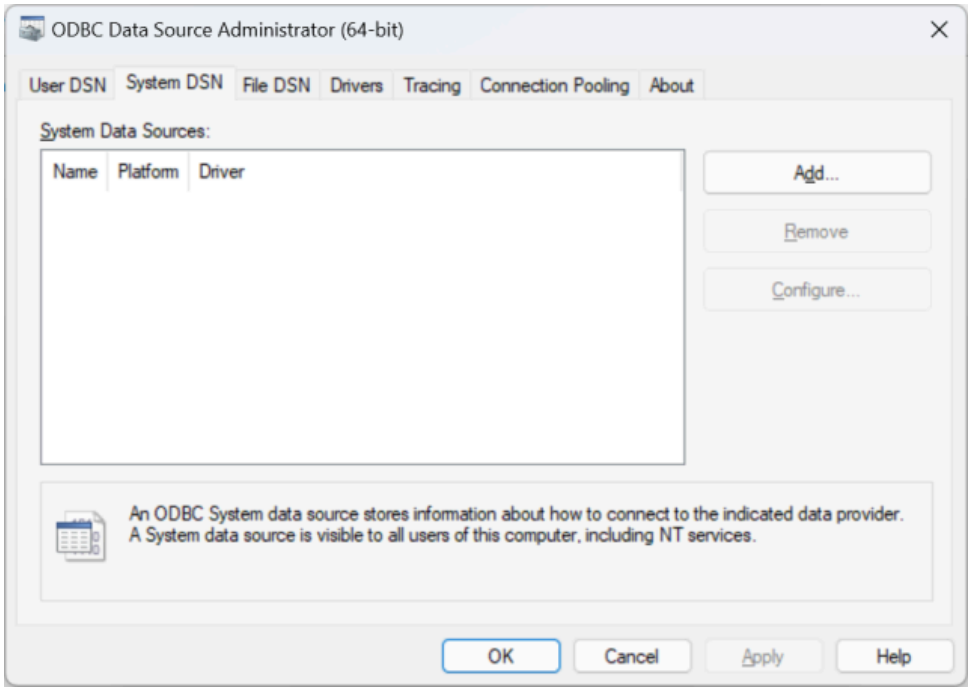
Menyiapkan driver Amazon DocumentDB ODBC di Windows

Gunakan prosedur berikut untuk mengatur driver Amazon DocumentDB ODBC di Windows:

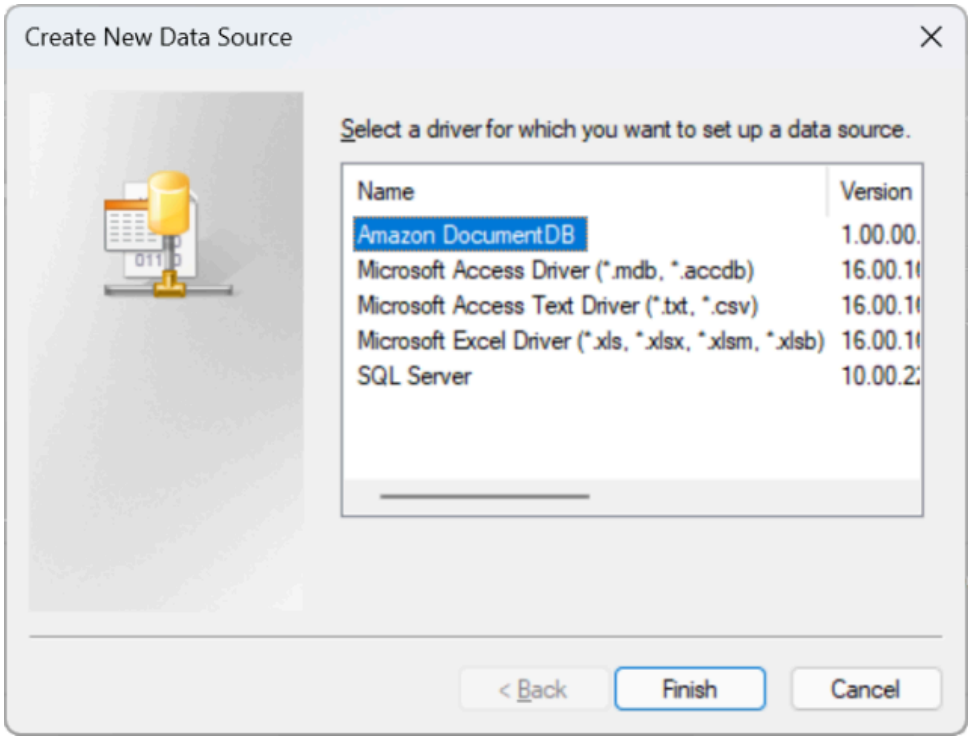
1. Buka Panel Kontrol di Windows dan cari ODBC (atau di menu, pilih Alat Windows > Sumber Data ODBC (32-bit) atau Sumber Data ODBC (64-bit)):



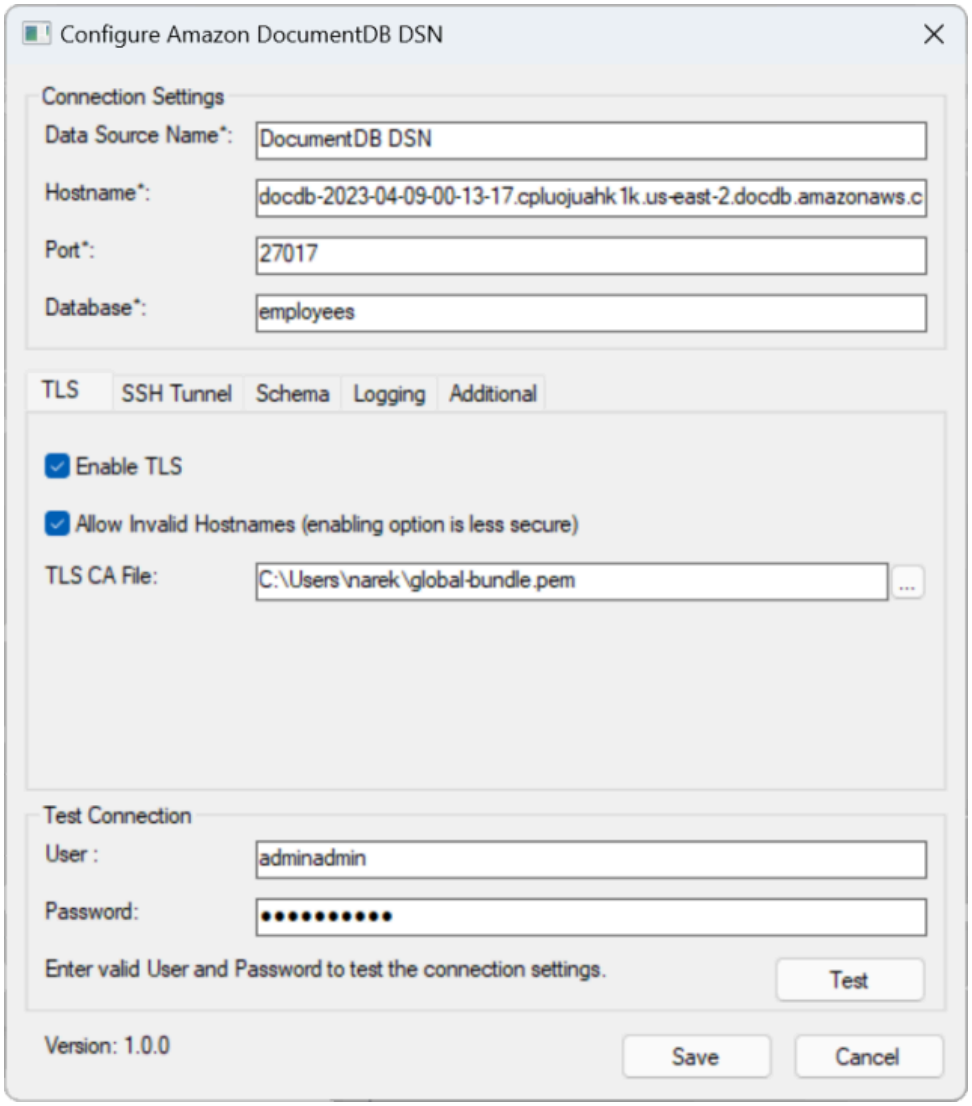
2. Pilih Administrator Sumber Data Driver ODBC yang sesuai: pilih versi 32-bit jika diinstal, jika tidak, pilih versi 64-bit.
3. Pilih tab Sytem DSN dan kemudian pilih Add... untuk menambahkan DSN baru:



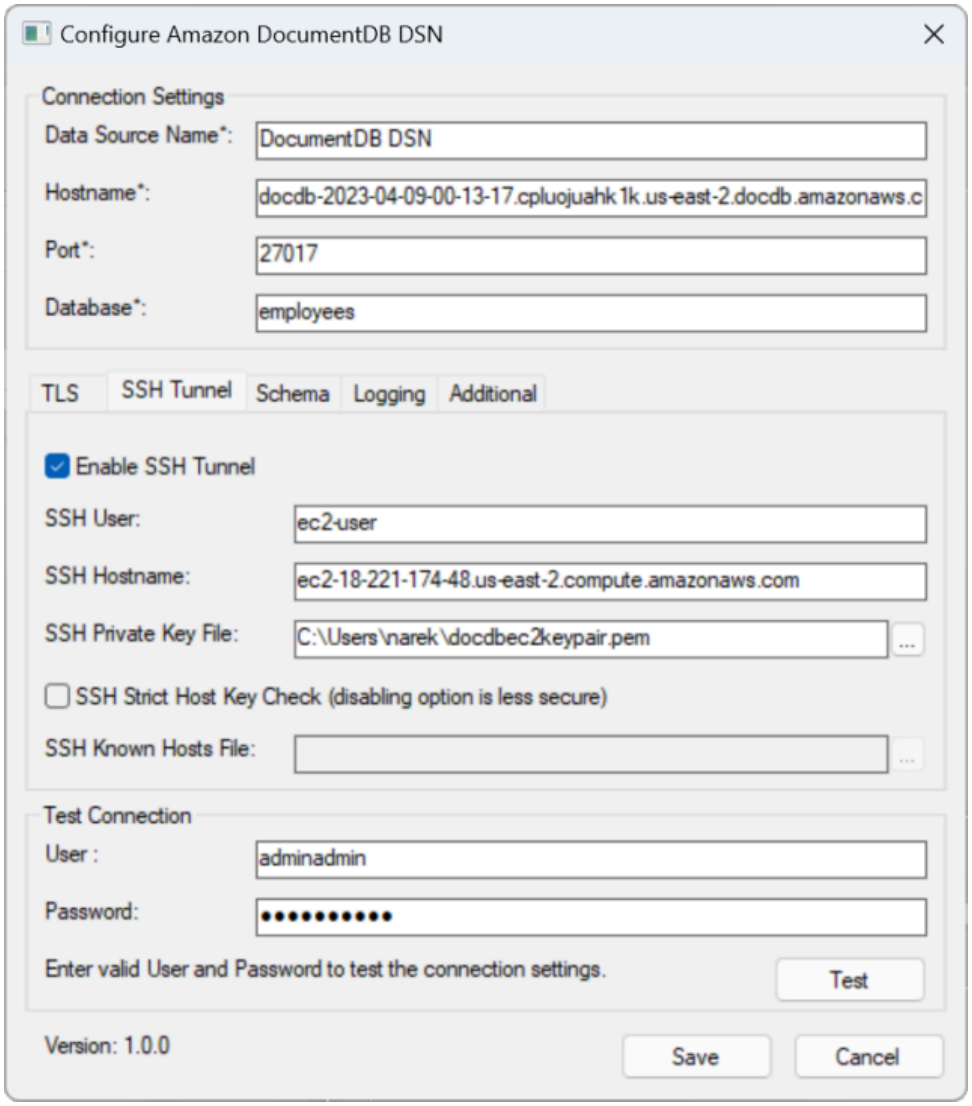
4. Pilih Amazon DocumentDB dari daftar driver sumber data:



5. Dalam dialog Configure Amazon DocumentDB DSN, lengkapi bidang Pengaturan Koneksi, TLS, dan Test Connection, lalu pilih Simpan:



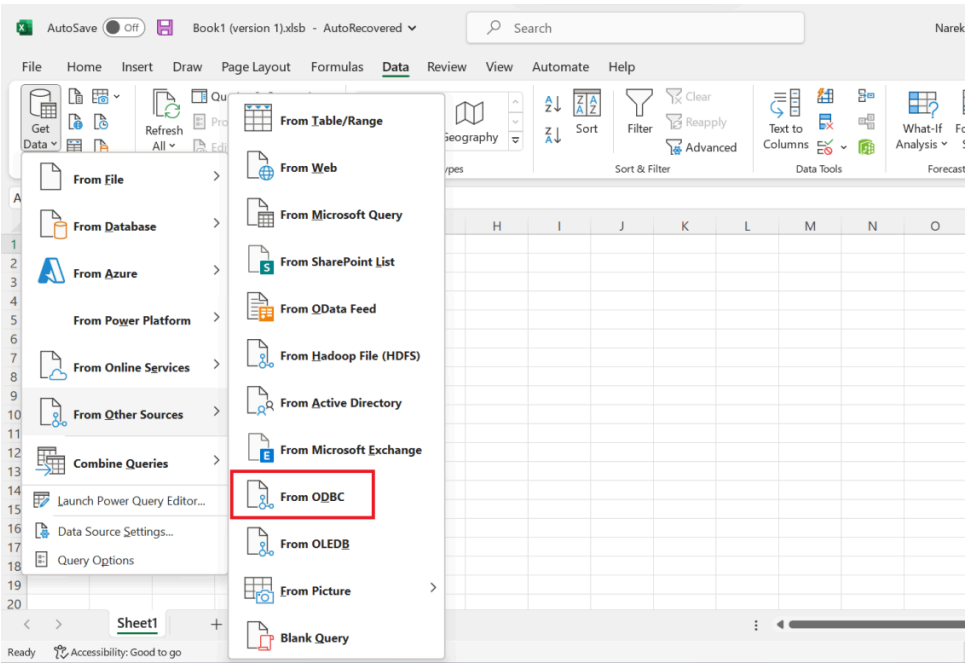
6. Pastikan Anda melengkapi formulir Windows secara akurat, karena detail koneksi akan berbeda tergantung pada metode tunneling SSH yang Anda pilih ke instans EC2. Lihat metode tunneling SSH di sini. Lihat Sintaks String Koneksi dan Opsi untuk informasi selengkapnya tentang setiap properti.



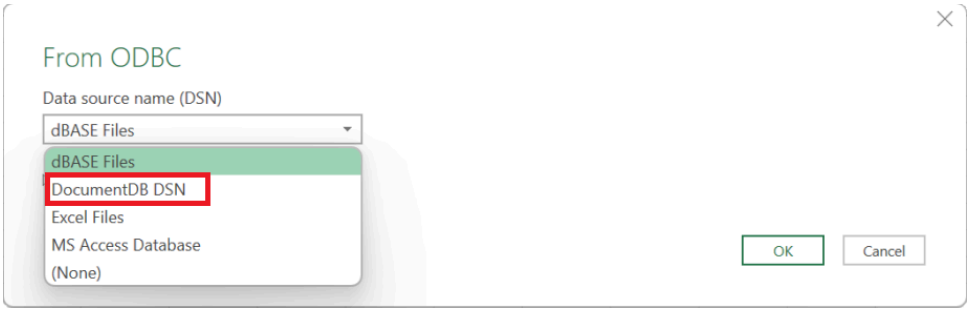
Untuk informasi selengkapnya tentang mengonfigurasi Driver Amazon DocumentDB ODBC di Windows, lihat di sini.

Connect ke Amazon DocumentDB dari Microsoft Excel

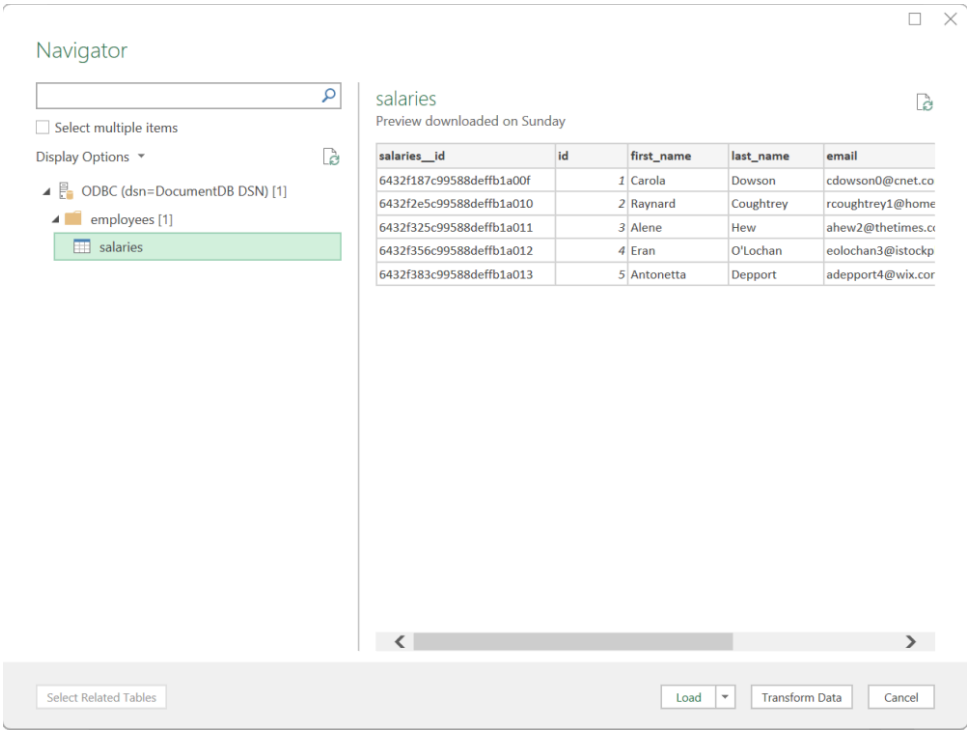
1. Pastikan driver Amazon DocumentDB telah diinstal dan dikonfigurasi dengan benar. Untuk informasi tambahan, lihat Menyiapkan driver ODBC di Windows.
2. Luncurkan Microsoft Excel.
3. Arahkan ke Data > Dapatkan Data > Dari Sumber Lain.
4. Pilih Dari ODBC:



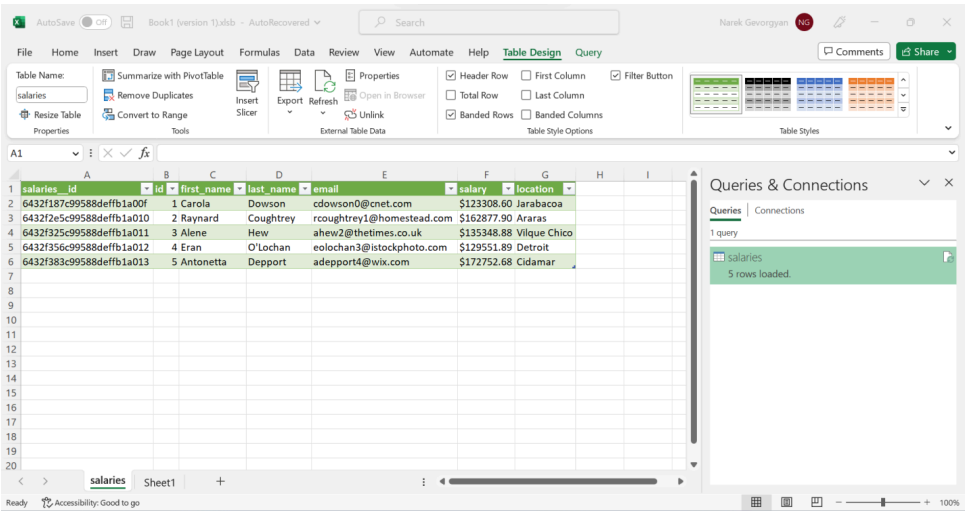
5. Pilih sumber data dari menu tarik-turun nama sumber data (DSN) yang terkait dengan Amazon DocumentDB:



6. Pilih koleksi dari mana Anda ingin memuat data ke Excel:



7. Muat data ke Excel:



Connect ke Amazon DocumentDB dari Microsoft Power BI Desktop

Topik

- Prasyarat
- Menambahkan konektor kustom Microsoft Power BI Desktop
- Menghubungkan menggunakan konektor kustom Amazon DocumentDB
- Mengkonfigurasi Microsoft Power BI Gateway

Prasyarat

Sebelum memulai, pastikan driver Amazon DocumentDB ODBC diinstal dengan benar.

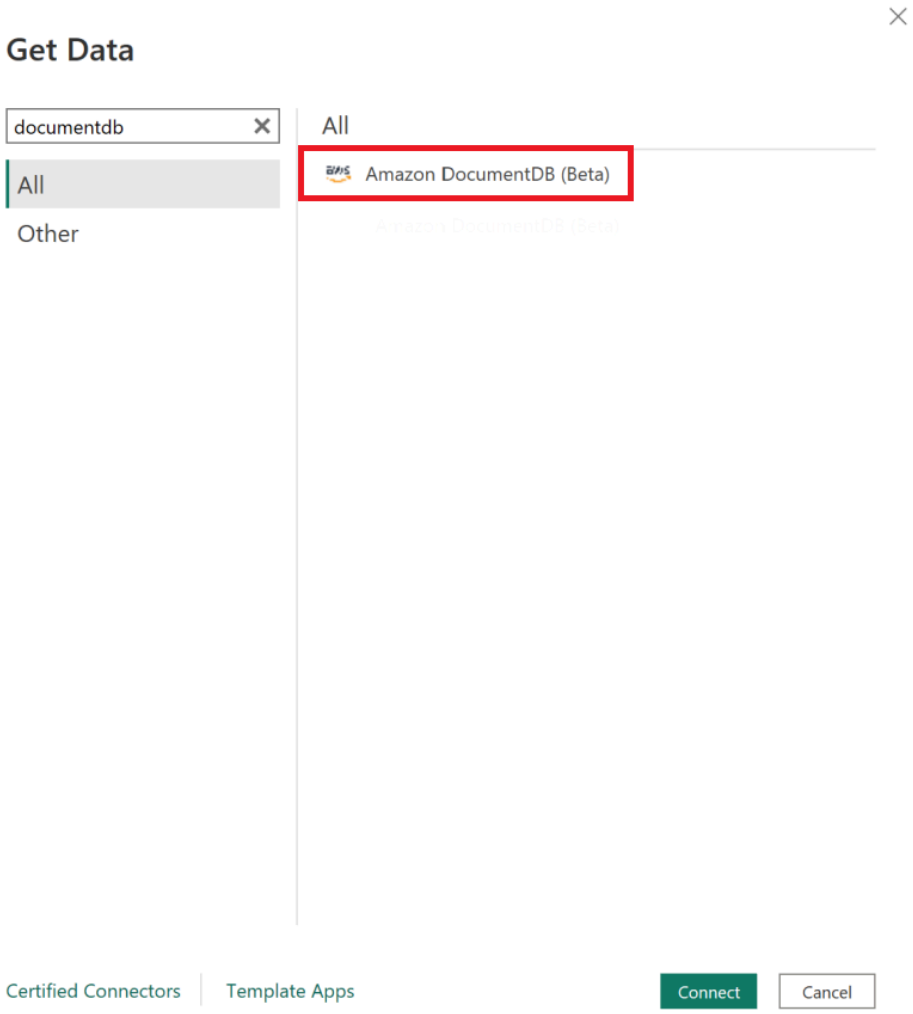
Menambahkan konektor kustom Microsoft Power BI Desktop

Salin `AmazonDocumentDBConnector.mez` file ke `<User>\Documents\Power BI Desktop \Custom Connectors\` folder (atau `<User>\OneDrive\Documents\Power BI Desktop \Custom Connectors` jika menggunakan OneDrive). Ini akan memungkinkan Power BI untuk mengakses konektor khusus. Anda bisa mendapatkan konektor ke Power BI Desktop di sini. Mulai ulang Power BI Desktop untuk memastikan konektor dimuat.

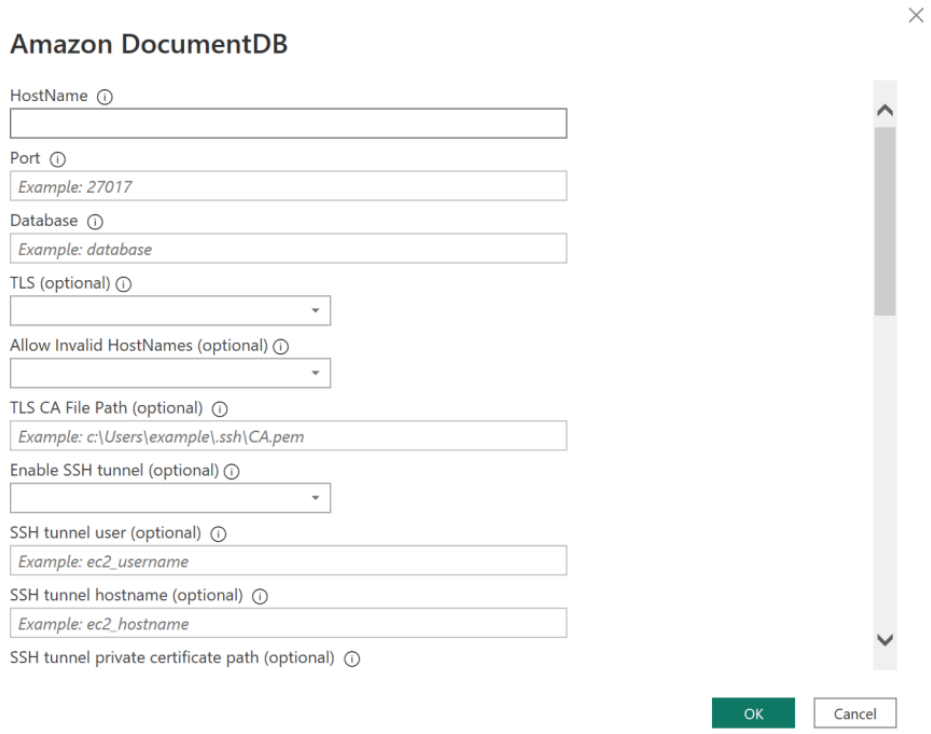
> **Note**
>
> Konektor khusus hanya mendukung nama pengguna dan kata sandi Amazon DocumentDB untuk otentikasi.

Menghubungkan menggunakan konektor kustom Amazon DocumentDB

1. Pilih Amazon DocumentDB (Beta) dari Dapatkan Data dan pilih Connect. Jika Anda mendapatkan peringatan untuk menggunakan layanan pihak ketiga, pilih Lanjutkan.



2. Masukkan semua informasi yang diperlukan untuk terhubung ke cluster Amazon DocumentDB Anda, lalu pilih OK:



> **Note**
>
> Bergantung pada konfigurasi Nama Sumber Data (DSN) driver ODBC Anda, layar detail koneksi SSH mungkin tidak ditampilkan jika Anda telah memberikan informasi yang diperlukan dalam pengaturan DSN.

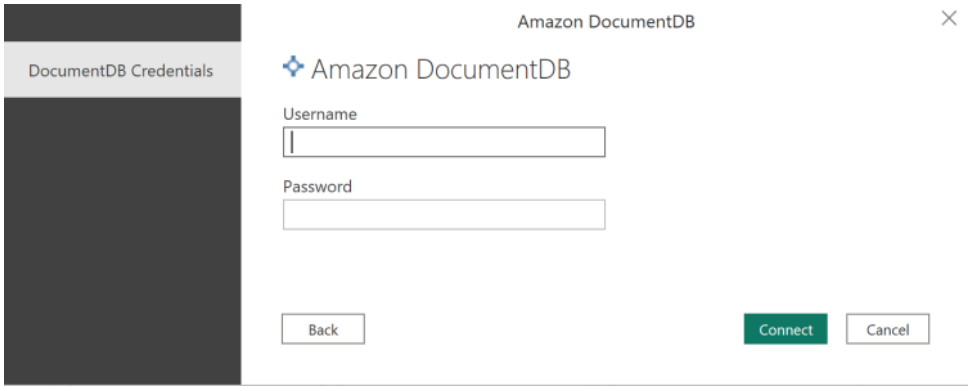
3. Pilih mode konektivitas data:

   - Impor - memuat semua data dan menyimpan informasi pada disk. Data harus disegarkan dan dimuat ulang untuk menampilkan pembaruan data.
   - Direct Query - tidak memuat data, tetapi melakukan kueri langsung pada data. Ini berarti bahwa data tidak perlu di-refresh dan dimuat ulang untuk menampilkan pembaruan data.

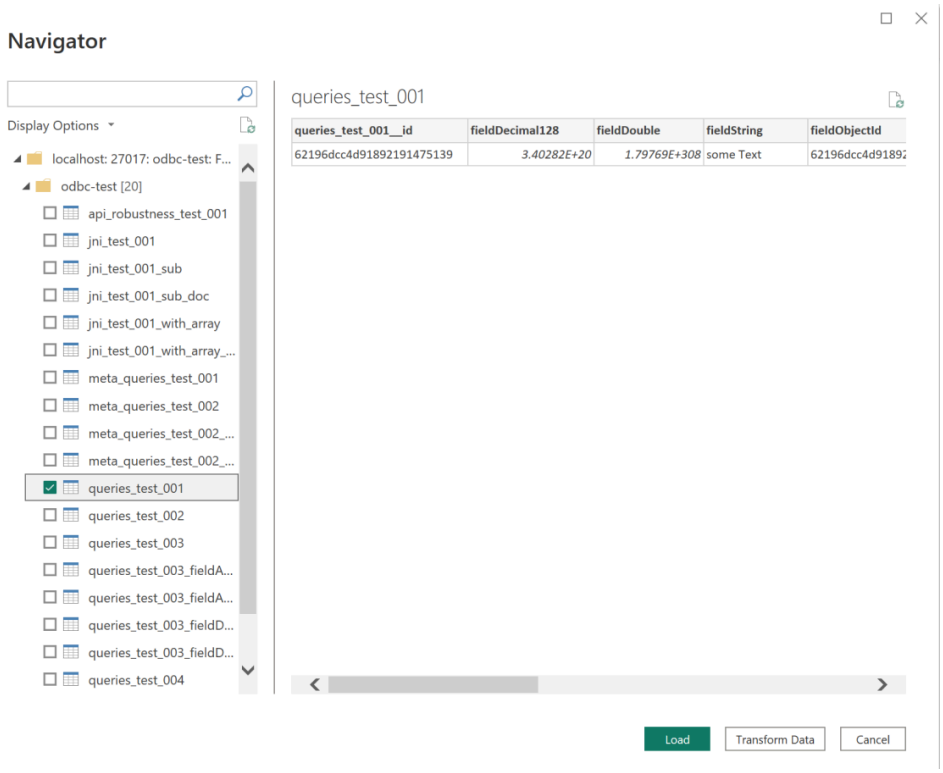


> **Note**
>
> Jika Anda menggunakan kumpulan data yang sangat besar, mengimpor semua data mungkin membutuhkan waktu yang lebih lama.

4. Jika ini adalah pertama kalinya menghubungkan ke sumber data ini, pilih jenis otentikasi dan masukkan kredensyal Anda saat diminta. Kemudian pilih Connect:



5. Dalam dialog Navigator, pilih tabel database yang Anda inginkan, lalu pilih Load untuk memuat data atau Transform Data untuk terus mengubah data.



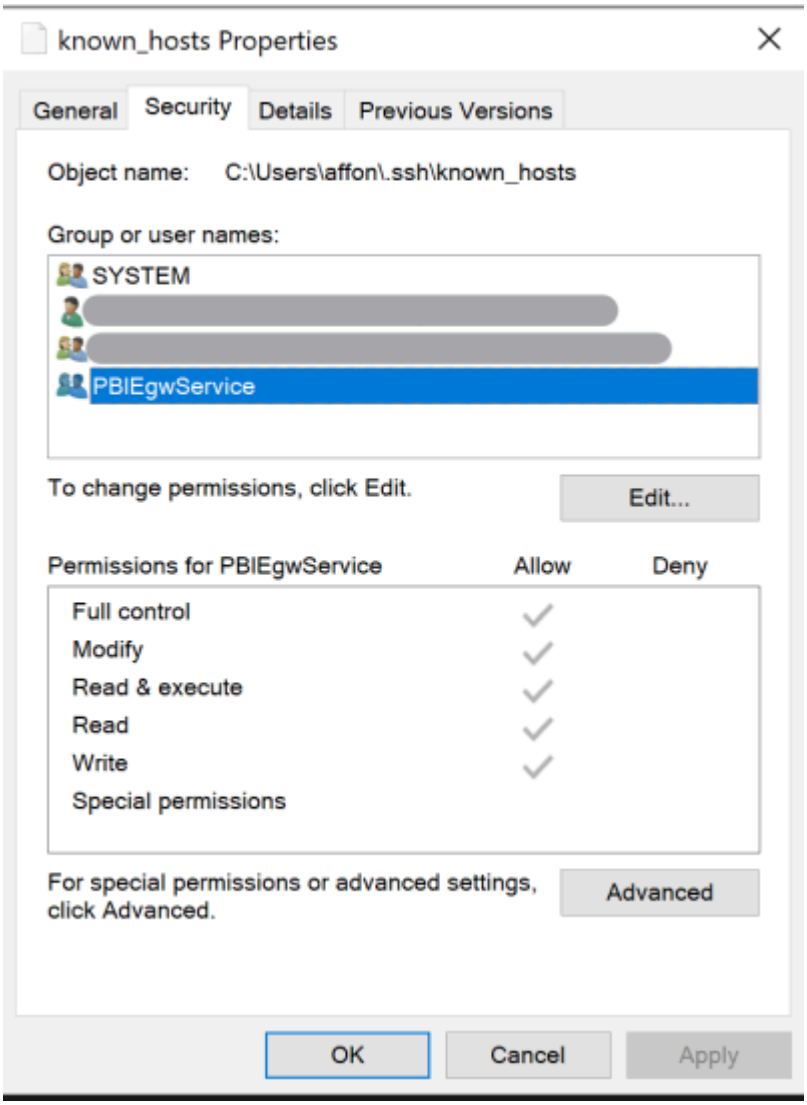
> **Note**
>
> Pengaturan sumber data Anda disimpan setelah Anda terhubung. Untuk memodifikasinya, pilih Transform Data > Pengaturan Sumber Data.

Mengkonfigurasi Microsoft Power BI Gateway

Prasyarat:

- Pastikan konektor khusus akan bekerja dengan Power BI Gateway.
- Pastikan ODBC DSN dibuat di sumber data ODBC di tab Sistem pada mesin tempat Power BI Gateway diinstal.

Jika Anda menggunakan fitur terowongan SSH internal, file `known_hosts` harus ditempatkan di mana akun layanan Power BI memiliki akses ke sana.



> **Note**
>
> Ini juga berlaku untuk file apa pun yang mungkin Anda perlukan untuk dapat membuat sambungan ke klaster Amazon DocumentDB Anda, seperti file sertifikat otoritas sertifikat (CA) (file pem).

Pembuatan skema otomatis

Driver ODBC memanfaatkan driver Amazon DocumentDB JDBC melalui JNI (Java Native Interface) - membuat fitur pembuatan skema otomatis bekerja sama di driver JDBC. Untuk informasi selengkapnya tentang pembuatan skema otomatis, lihat Pembuatan skema otomatis JDBC. Selain itu, untuk mempelajari lebih lanjut tentang arsitektur driver ODBC, lihat di sini.

Dukungan dan keterbatasan SQL

Driver Amazon DocumentDB ODBC adalah driver read-only yang mendukung subset dan beberapa ekstensi umum. SQL-92 Lihat dokumentasi dukungan dan batasan ODBC untuk informasi lebih lanjut.

Pemecahan masalah

Jika Anda mengalami masalah dalam menggunakan driver Amazon DocumentDB ODBC, lihat Panduan Pemecahan Masalah.

Program dengan Amazon DocumentDB

Layanan ini mendukung validasi skema JSON.

Topik

- Panduan pemrograman Amazon DocumentDB Java
- Menggunakan validasi skema JSON

Panduan pemrograman Amazon DocumentDB Java

Panduan komprehensif ini memberikan panduan terperinci untuk bekerja dengan Amazon DocumentDB menggunakan driver Java MongoDB, yang mencakup aspek-aspek penting dari operasi dan manajemen basis data.

Topik

- Pengantar
- Prasyarat
- Model data untuk panduan ini
- Menghubungkan ke Amazon DocumentDB dengan driver MongoDB Java
- Melakukan operasi CRUD di Amazon DocumentDB dengan Java
- Manajemen indeks di Amazon DocumentDB dengan Java
- Event-driven pemrograman dengan Amazon DocumentDB dan Java

Pengantar

Panduan ini dimulai dengan konektivitas, menjelaskan cara membuat koneksi aman ke cluster DocumentDB menggunakan driver MongoDB Java. Ini merinci komponen string koneksi, SSL/TLS implementasi, dan berbagai opsi koneksi termasuk otentikasi IAM dan penyatuan koneksi, bersama dengan strategi penanganan kesalahan yang kuat.

Di bagian operasi CRUD (buat, baca, perbarui, hapus), panduan ini secara menyeluruh mencakup manipulasi dokumen, menunjukkan cara membuat, membaca, memperbarui, dan menghapus dokumen menggunakan operasi tunggal dan massal. Ini menjelaskan penggunaan filter, kueri, dan berbagai opsi operasi, sambil menekankan praktik terbaik untuk penanganan kesalahan dan menerapkan logika coba lagi untuk meningkatkan keandalan. Panduan ini juga secara ekstensif mencakup manajemen indeks, merinci pembuatan dan pemeliharaan berbagai jenis indeks termasuk indeks bidang tunggal, senyawa, jarang, dan teks. Ini menjelaskan cara mengoptimalkan kinerja kueri melalui pemilihan indeks yang tepat dan penggunaan `explain()` fungsi untuk menganalisis rencana eksekusi kueri.

Bagian terakhir berfokus pada pemrograman berbasis peristiwa menggunakan aliran perubahan Amazon DocumentDB, menunjukkan bagaimana menerapkan pemantauan perubahan data waktu nyata dalam aplikasi Java. Ini mencakup implementasi kursor aliran perubahan, penanganan token resume untuk operasi berkelanjutan, dan operasi berbasis waktu untuk pemrosesan data historis. Sepanjang panduan ini, contoh kode praktis dan praktik terbaik disediakan, menjadikannya sumber daya yang tak ternilai bagi Anda saat membangun aplikasi Java yang kuat dengan Amazon DocumentDB.

Prasyarat

Sebelum Anda mulai, pastikan Anda memiliki yang berikut:

- AWS Akun dengan cluster DocumentDB yang dikonfigurasi. Lihat posting blog memulai ini untuk pengaturan cluster DocumentDB.
- Java Development Kit (JDK) diinstal (kami akan menggunakan Amazon Corretto 21 untuk panduan ini).
- Maven untuk manajemen ketergantungan.

Model data untuk panduan ini

Semua kode contoh dalam panduan ini mengasumsikan koneksi ke database pengujian “ProgGuideData” yang memiliki koleksi “Restoran”. Semua kode sampel dalam panduan ini berfungsi pada sistem daftar restoran dan di bawah ini adalah contoh seperti apa dokumen dalam sistem ini:

```
{
    "_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
```

```
    }
}
```

Semua contoh kode yang menunjukkan CRUD, manajemen indeks, dan pemrograman berbasis peristiwa mengasumsikan bahwa Anda memiliki `MongoClient`objek`dbClient`, objek, dan `MongoDatabase`objek`connectionDB`. `MongoCollection`collection

> **Note**
>
> Semua contoh kode dalam panduan ini telah diuji dengan driver MongoDB Java versi 5.3.0.

Menghubungkan ke Amazon DocumentDB dengan driver MongoDB Java

Bagian ini menyediakan panduan langkah demi langkah untuk menghubungkan ke Amazon DocumentDB menggunakan driver Java. Ini akan membantu Anda memulai dengan mengintegrasikan DocumentDB ke dalam aplikasi Java Anda.

Topik

- Langkah 1: Siapkan proyek Anda
- Langkah 2: Buat string koneksi
- Langkah 3: Tulis kode koneksi
- Langkah 4: Tangani pengecualian koneksi
- Langkah 5: Menjalankan kode
- Praktik terbaik koneksi

Langkah 1: Siapkan proyek Anda

1. Menggunakan Maven, buat proyek java:

   ```
   mvn archetype:generate -DgroupId=com.docdb.guide -DartifactId=my-docdb-project -
   DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
   ```

2. Tambahkan driver MongoDB Java sebagai dependensi untuk proyek di file 'pom.xml' Anda:

   ```
   <dependency>
       <groupId>org.mongodb</groupId>
       <artifactId>mongodb-driver-sync</artifactId>
   ```

```
    <version>5.3.0</version>
</dependency>
```

Langkah 2: Buat string koneksi

String koneksi Amazon DocumentDB sangat penting untuk membuat koneksi antara aplikasi Anda dan cluster DocumentDB Anda. String ini merangkum informasi penting seperti titik akhir cluster, port, detail otentikasi, dan berbagai opsi koneksi. Untuk membangun string koneksi DocumentDB, Anda biasanya mulai dengan format dasar:

```
"mongodb://username:password@cluster-endpoint:port/?[connection options]"
```

Anda harus mengganti “nama pengguna” dan “kata sandi” dengan kredensil Anda yang sebenarnya. Anda dapat menemukan titik akhir dan nomor port cluster Anda di Konsol Manajemen AWS serta melalui file. AWS CLI Lihat Menemukan titik akhir klaster untuk menemukan titik akhir cluster untuk cluster Anda. Port default untuk DocumentDB adalah 27017.

Contoh string koneksi

- Membuat koneksi ke DocumentDB menggunakan enkripsi dalam perjalanan dan memastikan permintaan baca masuk ke replika baca dan tulis ke primer:

```
"mongodb://username:password@cluster-endpoint:27017/?tls=true&
   tlsCAFile=global-bundle.pem&
   readPreference=secondaryPreferred&
   retryWrites=false"
```

- Membuat koneksi ke DocumentDB menggunakan otentikasi IAM:

```
"mongodb://cluster-endpoint:27017/?tls=true&
   tlsCAFile=global-bundle.pem&
   readPreference=secondaryPreferred&
   retryWrites=false&
   authSource=%24external&
   authMechanism=MONGODB-AWS"
```

Opsi berbeda yang tersedia untuk string koneksi adalah sebagai berikut:

- Sertifikat TLS

- Membaca dari replika baca
- Tulis kepedulian dan jurnal
- RetryWrites
- Autentikasi IAM
- Kolam koneksi
- Parameter batas waktu koneksi

Sertifikat TLS

`tls=true|false`— Opsi ini mengaktifkan atau menonaktifkan Transport Layer Security (TLS). Secara default, enkripsi dalam perjalanan diaktifkan di Amazon DocumentDB cluster dan oleh karena itu, kecuali TLS dinonaktifkan pada tingkat cluster, nilai untuk opsi ini seharusnya. `true`

Saat menggunakan TLS, kode perlu menyediakan sertifikat SSL saat membuat koneksi ke cluster DocumentDB. Unduh sertifikat yang diperlukan untuk membuat koneksi aman ke cluster: `global-bundle.pem`. Ada dua cara untuk menggunakan `global-bundle.pem` file tersebut.

- Opsi 1 — Ekstrak semua sertifikat dari `global-bundle.pem` file dan gunakan keytool Java untuk menyimpannya dalam `.jks` file yang nantinya dapat digunakan dalam kode. Lihat tab Java Menghubungkan dengan TLS diaktifkan untuk skrip yang menunjukkan cara melakukan ini.
- Opsi 2 - Tambahkan `global-bundle.pem` file secara dinamis dalam kode, buat keystore dalam memori dan gunakan `SSLContext` untuk memberikan sertifikat sebagai bagian dari membuat koneksi.

Membaca dari replika baca

`replicaSet=rs0&readPreference=secondaryPreferred`— Menentukan dua opsi ini merutekan semua permintaan baca ke replika baca, dan menulis permintaan ke instance utama. Menggunakan `replicaSet=rs0` dalam string koneksi memungkinkan driver MongoDB untuk mempertahankan tampilan topologi cluster yang diperbarui secara otomatis, memungkinkan aplikasi untuk mempertahankan visibilitas konfigurasi node saat ini saat instance ditambahkan atau dihapus. Tidak menyediakan opsi ini atau menentukan `readPreference=primary` mengirim semua pembacaan dan penulisan ke instance utama. Untuk opsi lainnya`readPreference`, lihatBaca opsi preferensi.

Tulis kepedulian dan jurnal

Kepedulian menulis menentukan tingkat pengakuan yang diminta dari database untuk operasi penulisan. Driver MongoDB menyediakan opsi untuk menyetel masalah penulisan dan file jurnal. Amazon DocumentDB tidak mengharapkan Anda untuk menetapkan masalah penulisan dan jurnal, dan mengabaikan nilai yang dikirim `w` untuk `j` dan (dan). `writeConcern` `journal` DocumentDB selalu menulis data `writeConcern` dengan`majority`: `journal` dan`true`: sehingga penulisan direkam dengan tahan lama pada sebagian besar node sebelum mengirim pengakuan ke klien.

RetryWrites

`retryWrites=false`- DocumentDB tidak mendukung penulisan yang dapat dicoba ulang dan oleh karena itu atribut ini harus selalu disetel ke. `false`

Autentikasi IAM

`authSource=%24external`dan **`authMechanism=MONGODB-AWS`** — Kedua parameter ini digunakan untuk mengautentikasi menggunakan AWS Identity and Access Management. Autentikasi IAM saat ini hanya tersedia di cluster berbasis instance versi 5.0. Untuk informasi selengkapnya, lihat Otentikasi menggunakan identitas IAM.

Kolam koneksi

Opsi ini tersedia untuk penyatuan koneksi:

- **`maxPoolSize`**— Mengatur jumlah maksimum koneksi yang dapat dibuat di kolam renang. Ketika semua koneksi sedang digunakan dan permintaan baru masuk, ia menunggu koneksi tersedia. Default untuk driver MongoDB Java adalah 100.
- **`minPoolSize`**— Menunjukkan jumlah minimum koneksi yang harus dipertahankan di kolam setiap saat. Default untuk driver MongoDB Java adalah 0.
- **`maxIdleTimeMS`**— Menentukan berapa lama koneksi dapat tetap diam di kolam sebelum ditutup dan dilepas. Default untuk driver MongoDB Java adalah 100 milidetik.
- **`waitQueueTimeoutMS`**— Mengkonfigurasi berapa lama utas harus menunggu koneksi tersedia saat kolam berada pada ukuran maksimumnya. Jika koneksi tidak tersedia dalam waktu ini, pengecualian dilemparkan. Nilai default untuk driver MongoDB Java adalah 120.000 milidetik (2 menit).

Parameter batas waktu koneksi

Timeout adalah mekanisme untuk membatasi jumlah waktu operasi atau upaya koneksi dapat dilakukan sebelum dianggap gagal. Parameter batas waktu berikut tersedia untuk mencegah penantian yang tidak terbatas dan mengelola alokasi sumber daya:

- **`connectTimeoutMS`**— Mengkonfigurasi berapa lama driver akan menunggu untuk membuat koneksi ke cluster. Defaultnya adalah 10.000 milidetik (10 detik).
- **`socketTimeoutMS`**— Menentukan berapa lama driver akan menunggu respons dari server untuk operasi non-tulis. Defaultnya adalah 0, (tidak ada batas waktu atau tak terbatas).
- **`serverSelectionTimeoutMS`**— Spesies berapa lama pengemudi akan menunggu untuk menemukan server yang tersedia di cluster. Nilai default untuk pengaturan ini adalah 30 detik dan cukup untuk instance primer baru yang akan dipilih selama failover.

Langkah 3: Tulis kode koneksi

Contoh kode berikut menunjukkan cara membuat koneksi TLS ke Amazon DocumentDB:

- Ini menciptakan Java `KeyStoreSSLContextdan>` objek.
- Ini juga menciptakan `MongoClientSettings`objek dengan meneruskannya ke `ConnectionString`objek. Untuk membuat koneksi TLS, Anda harus menggunakan `MongoClientSettings` objek untuk mengikat dan. `connectionstring` `sslcontext`
- Menggunakan `MongoClients`mendapat `MongoClient`objek.

```
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);
    }
}
```

Langkah 4: Tangani pengecualian koneksi

Saat bekerja dengan DocumentDB dalam aplikasi Java, penanganan pengecualian koneksi sangat penting untuk mempertahankan operasi database yang kuat dan andal. Mengelola dengan benar, pengecualian ini tidak hanya membantu dalam mendiagnosis masalah dengan cepat, tetapi juga memastikan bahwa aplikasi Anda dapat dengan anggun menangani gangguan jaringan sementara atau tidak tersedianya server, yang mengarah pada peningkatan stabilitas dan pengalaman pengguna. Beberapa pengecualian penting yang terkait dengan membangun koneksi meliputi:

- **`MongoException`**— Pengecualian umum dan dapat dikeluarkan dalam berbagai skenario yang tidak tercakup oleh pengecualian yang lebih spesifik. Pastikan pengecualian ini ditangani setelah semua pengecualian spesifik lainnya karena ini adalah catch umum semua pengecualian MongoDB.
- **`MongoTimeoutException`**— Dikeluarkan saat waktu operasi habis. Misalnya, menanyakan titik akhir cluster yang tidak ada.
- **`MongoSocketException`**— Dikeluarkan untuk masalah terkait jaringan. Misalnya, pemutusan jaringan tiba-tiba selama operasi.

- **`MongoSecurityException`**— Diterbitkan saat otentikasi gagal. Misalnya, menghubungkan dengan kredensyal yang salah.
- **`MongoConfigurationException`**— Diterbitkan ketika ada kesalahan dalam konfigurasi klien. Misalnya, menggunakan string koneksi yang tidak valid.

Langkah 5: Menjalankan kode

Contoh kode berikut membuat koneksi Amazon DocumentDB dan mencetak semua database:

```
public static void TestConnection() {
    try (MongoClient mongoClient = makeDbConnection(DATABASE_NAME, DB_USER_NAME,
 DB_PASSWORD, DB_CLUSTER_ENDPOINT, KEYSTORE_PASSWORD)) {
        List < String > databases = mongoClient.listDatabaseNames().into(new ArrayList
 < > ());
        System.out.println("Databases: " + databases);
    } catch (MongoException e) {
        System.err.println("MongoDB error: " + e.getMessage());
        throw new RuntimeException(e);
    }
}
```

Praktik terbaik koneksi

Berikut ini adalah praktik terbaik yang perlu dipertimbangkan saat menghubungkan ke Amazon DocumentDB dengan driver MongoDB Java:

- Selalu tutup Anda `MongoClient`ketika Anda tidak lagi membutuhkan klien untuk melepaskan sumber daya.
- Tangani pengecualian dengan tepat dan terapkan logging kesalahan yang tepat.
- Gunakan variabel lingkungan atau AWS Secrets Manager untuk menyimpan informasi sensitif seperti nama pengguna dan kata sandi.

Melakukan operasi CRUD di Amazon DocumentDB dengan Java

Bagian ini membahas melakukan operasi CRUD (membuat, membaca, memperbarui, menghapus) di Amazon DocumentDB menggunakan driver MongoDB Java.

Topik

- Membuat dan menyisipkan dokumen dalam koleksi DocumentDB
- Membaca dan mengambil data dari koleksi DocumentDB
- Memperbarui dokumen yang ada dalam koleksi DocumentDB
- Menghapus dokumen dari koleksi DocumentDB
- Penanganan kesalahan dengan logika coba lagi

Membuat dan menyisipkan dokumen dalam koleksi DocumentDB

Memasukkan dokumen ke Amazon DocumentDB memungkinkan Anda menambahkan data baru ke koleksi Anda. Ada beberapa cara untuk melakukan penyisipan, tergantung pada kebutuhan Anda dan volume data yang Anda kerjakan. Metode paling dasar untuk memasukkan dokumen individual ke koleksi adalah`insertOne()`. Untuk memasukkan beberapa dokumen sekaligus, Anda dapat menggunakan `insertMany()` metode ini, yang memungkinkan Anda menambahkan array dokumen dalam satu operasi. Metode lain untuk memasukkan banyak dokumen dalam koleksi DocumentDB adalah. `bulkWrite()` Dalam panduan ini, kami membahas semua metode ini untuk membuat dokumen dalam koleksi DocumentDB.

`insertOne()`

Mari kita mulai dengan memeriksa cara memasukkan dokumen individual ke dalam koleksi Amazon DocumentDBB. Menyisipkan satu dokumen dilakukan dengan menggunakan metode ini`insertOne()`. Metode ini mengambil BsonDocumentuntuk penyisipan dan mengembalikan `InsertOneResult`objek yang dapat digunakan untuk mendapatkan id objek dari dokumen baru dimasukkan. Contoh kode di bawah ini menunjukkan memasukkan satu dokumen restoran ke dalam koleksi:

```
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;
}
```

Saat menggunakan`insertOne()`, pastikan untuk menyertakan penanganan kesalahan yang sesuai. Misalnya, dalam kode di atas, "`restaurantId`" memiliki indeks unik dan oleh karena itu menjalankan kode ini lagi akan memunculkan yang berikut`MongoWriteException`:

```
Failed to insert document: Write operation error on server
 docdbCluster.docdb.amazonaws.com:27017.
Write error: WriteError{code=11000, message='E11000 duplicate key error collection:
 Restaurants index: restaurantId_1', details={}}.
```

InsertMany ()

Metode utama yang digunakan untuk memasukkan banyak dokumen ke dalam koleksi adalah insertMany () dan. `bulkWrite()`

`insertMany()`Metode ini adalah cara paling sederhana untuk memasukkan beberapa dokumen dalam satu operasi. Ini menerima daftar dokumen dan memasukkannya ke dalam koleksi. Metode ini sangat ideal ketika Anda memasukkan sekumpulan dokumen baru yang independen satu sama lain dan tidak memerlukan pemrosesan khusus atau operasi campuran. Kode berikut menunjukkan membaca dokumen JSON dari file dan memasukkannya ke dalam koleksi. `insertMany()`Fungsi mengembalikan [`InsertManyResult`](#)`InsertManyResult`objek yang dapat digunakan untuk mendapatkan ID dari semua dokumen yang dimasukkan.

```
// 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()

`bulkWrite()`Metode ini memungkinkan untuk melakukan beberapa operasi tulis (menyisipkan, memperbarui, menghapus) dalam satu batch. Anda dapat menggunakan `bulkWrite()` ketika Anda perlu melakukan berbagai jenis operasi dalam satu batch, seperti memasukkan beberapa dokumen sambil memperbarui yang lain. `bulkWrite()`mendukung dua jenis batch write, ordered dan unordered:

- Operasi yang dipesan — (default) Amazon DocumentDB memproses operasi penulisan secara berurutan, dan berhenti pada kesalahan pertama yang ditemuinya. Ini berguna ketika urutan operasi penting, seperti ketika operasi nanti bergantung pada yang sebelumnya. Namun, operasi yang diperintahkan umumnya lebih lambat daripada operasi yang tidak teratur. Dengan operasi yang dipesan, Anda harus mengatasi kasus di mana batch berhenti pada kesalahan pertama, yang berpotensi membuat beberapa operasi tidak diproses.
- Operasi tidak teratur - Memungkinkan Amazon DocumentDB memproses sisipan sebagai eksekusi tunggal dalam database. Jika terjadi kesalahan dengan satu dokumen, operasi berlanjut dengan dokumen yang tersisa. Ini sangat berguna ketika Anda memasukkan data dalam jumlah besar dan dapat mentolerir beberapa kegagalan, seperti selama migrasi data atau impor massal di mana beberapa dokumen mungkin gagal karena kunci duplikat. Dengan operasi yang tidak teratur, Anda harus mengatasi skenario keberhasilan sebagian di mana beberapa operasi berhasil sementara yang lain gagal.

Saat bekerja dengan `bulkWrite()` metode ini, ada beberapa kelas penting yang diperlukan. Pertama, `WriteModel`kelas berfungsi sebagai kelas dasar untuk semua operasi tulis dan dengan implementasi spesifik seperti `InsertOneModel`,, `UpdateOneModelUpdateManyModel`, `DeleteOneModel`, dan `DeleteManyModel`menangani berbagai jenis operasi.

`BulkWriteOptions`Kelas diperlukan untuk mengkonfigurasi perilaku operasi massal, seperti pengaturan ordered/unordered eksekusi atau melewati validasi dokumen. `BulkWriteResult`Kelas memberikan informasi rinci tentang hasil eksekusi, termasuk jumlah dokumen yang dimasukkan, diperbarui, dan dihapus.

Untuk penanganan kesalahan, MongoBulkWriteExceptionkelas sangat penting karena berisi informasi tentang kegagalan selama operasi massal, sementara BulkWriteErrorkelas memberikan rincian spesifik tentang kegagalan operasi individu. Kode berikut menunjukkan contoh memasukkan daftar dokumen, serta memperbarui dan menghapus satu dokumen, semua dalam pelaksanaan panggilan `bulkWrite()` metode tunggal. Kode ini juga menunjukkan cara bekerja dengan BulkWriteOptionsdan BulkWriteResult, serta penanganan kesalahan yang tepat dari `bulkWrite()` operasi.

```
List < WriteModel < Document >> bulkOperations = new ArrayList < > ();
// get list of 10 documents representing 10 restaurants
List < Document > restaurantsToInsert = getSampleData();

for (Document doc: restaurantsToInsert) {
    bulkOperations.add(new InsertOneModel < > (doc));
}
// Update operation
bulkOperations.add(new UpdateOneModel < > (
    new Document("restaurantId", "REST-Y2E9H5"),
    new Document("", new Document("stats.likes", 20))
    .append("", new Document("rating.average", 4.5))));
// Delete operation
bulkOperations.add(new DeleteOneModel < > (new Document("restaurantId", "REST-
D2L431")));

// Perform bulkWrite operation
try {
    BulkWriteOptions options = new BulkWriteOptions()
        .ordered(false); // Allow unordered inserts

    BulkWriteResult result = collection.bulkWrite(bulkOperations, options);

    System.out.println("Inserted: " + result.getInsertedCount());
    System.out.println("Updated: " + result.getModifiedCount());
    System.out.println("Deleted: " + result.getDeletedCount());
} catch (MongoBulkWriteException e) {
    System.err.println("Bulk write error occurred: " + e.getMessage());
    // Log individual write errors
    for (BulkWriteError error: e.getWriteErrors()) {
        System.err.printf("Error at index %d: %s (Code: %d)%n", error.getIndex(),
 error.getMessage(),
            error.getCode());
```

```
        // Log the problematic document
        Document errorDoc = new Document(error.getDetails());
        if (errorDoc != null) {
            System.err.println("Problematic document: " + errorDoc);
        }
    }
} catch (Exception e) {
    System.err.println("Error during bulkWrite: " + e.getMessage());
}
```

Retryable menulis

Tidak seperti MongoDB, Amazon DocumentDB tidak mendukung penulisan yang dapat dicoba ulang. Akibatnya, Anda harus menerapkan logika coba ulang khusus dalam aplikasi mereka, terutama untuk menangani masalah jaringan atau tidak tersedianya layanan sementara. Strategi coba ulang yang diimplementasikan dengan baik, biasanya, melibatkan peningkatan penundaan antara upaya coba lagi dan membatasi jumlah total percobaan ulang. Lihat Penanganan kesalahan dengan logika coba lagi di bawah untuk contoh kode membangun logika coba lagi dengan penanganan kesalahan.

Membaca dan mengambil data dari koleksi DocumentDB

Kueri dokumen di Amazon DocumentDB berkisar pada beberapa komponen utama yang memungkinkan Anda mengambil dan memanipulasi data secara tepat. `find()`Metode ini adalah API kueri mendasar di driver MongoDB Java. Ini memungkinkan pengambilan data yang kompleks dengan banyak opsi untuk memfilter, menyortir, dan memproyeksikan hasil. Selain `find()` metode, `Filters`dan `FindIterable`dua komponen fundamental lainnya yang menyediakan blok bangunan untuk operasi query di driver MongoDB Java.

`Filters`Kelas ini adalah kelas utilitas dalam driver MongoDB Java yang menyediakan API lancar untuk membangun filter query. Kelas ini menawarkan metode pabrik statis yang membuat instance `Bson` objek yang mewakili berbagai kondisi kueri. Metode yang paling umum digunakan termasuk `eq()` untuk perbandingan kesetaraan,,,, `gt()` `lt()`, dan untuk perbandingan numerik`gte()`, dan `lte()` untuk menggabungkan beberapa kondisi, `and()` dan `or()` untuk tes keanggotaan array, `in()` dan `nin()` untuk pencocokan pola. `regex()` Kelas ini dirancang untuk menjadi type-safe dan menyediakan pemeriksaan waktu kompilasi yang lebih baik dibandingkan dengan query berbasis dokumen mentah, menjadikannya pendekatan yang disukai untuk membangun kueri DocumentDB dalam aplikasi Java. Penanganan kesalahan kuat, dengan pengecualian yang jelas dilemparkan untuk konstruksi filter yang tidak valid.

`FindIterable`adalah antarmuka khusus yang dirancang untuk menangani hasil `find()` metode. Ini menyediakan serangkaian metode yang kaya untuk menyempurnakan dan mengendalikan eksekusi kueri, menawarkan API yang lancar untuk rantai metode. Antarmuka mencakup metode modifikasi kueri penting seperti `limit()` untuk membatasi jumlah dokumen yang dikembalikan, `skip()` untuk pagination, `sort()` untuk memesan hasil, `projection()` untuk memilih bidang tertentu, dan `hint()` untuk pemilihan indeks. Operasi batch, skip, dan limit di `FindIterable` adalah pagination penting dan alat manajemen data yang membantu mengontrol bagaimana dokumen diambil dan diproses dari database.

Batching (`batchSize`) mengontrol berapa banyak dokumen yang dikembalikan DocumentDB ke klien dalam satu perjalanan pulang-pergi jaringan. Saat Anda menetapkan ukuran batch, DocumentDB tidak mengembalikan semua dokumen yang cocok sekaligus, melainkan mengembalikannya dalam grup dengan ukuran batch yang ditentukan.

Skip memungkinkan Anda untuk mengimbangi titik awal hasil Anda, pada dasarnya memberi tahu DocumentDB untuk melewati sejumlah dokumen tertentu sebelum mulai mengembalikan kecocokan. Misalnya, `skip(20)` akan mem-bypass 20 dokumen pertama yang cocok. Ini biasanya digunakan dalam skenario pagination di mana Anda ingin mengambil halaman hasil berikutnya.

Batas membatasi jumlah total dokumen yang dapat dikembalikan dari kueri. Saat Anda menentukan`limit(n)`, DocumentDB akan berhenti mengembalikan dokumen setelah mengembalikan dokumen 'n', bahkan jika ada lebih banyak kecocokan dalam database.

`FindIterable`mendukung pola iterator dan kursor saat mengambil dokumen dari Amazon DocumentDB. Manfaat menggunakan `FindIterable` sebagai iterator adalah memungkinkan pemuatan dokumen yang lambat dan hanya mengambil dokumen saat diminta oleh aplikasi. Manfaat lain menggunakan iterator adalah Anda tidak bertanggung jawab untuk menjaga koneksi ke cluster dan dengan demikian tidak diperlukan penutupan koneksi secara eksplisit.

`FindIterable`juga menyediakan dukungan `MongoCursor`yang memungkinkan pola kursor digunakan saat bekerja dengan kueri Amazon DocumentDB. `MongoCursor`adalah implementasi khusus driver MongoDB Java yang menyediakan kontrol atas operasi database dan manajemen sumber daya. Ini mengimplementasikan `AutoCloseable` antarmuka, memungkinkan manajemen sumber daya eksplisit melalui blok coba-dengan-sumber daya, yang sangat penting untuk menutup koneksi database dengan benar dan membebaskan sumber daya server. Secara default, waktu kursor habis dalam 10 menit dan DocumentDB tidak memberi Anda opsi untuk mengubah perilaku time out ini. Saat bekerja dengan data batch, pastikan untuk mengambil kumpulan data berikutnya sebelum waktu kursor habis. Salah satu pertimbangan utama saat menggunakan `MongoCursor` adalah bahwa hal itu memerlukan penutupan eksplisit untuk mencegah kebocoran sumber daya.

Pada bagian ini, beberapa contoh disajikan untuk`find()`, `Filters` dan`FindIterable`.

Contoh kode berikut menunjukkan cara menggunakan `find()` untuk mengambil satu dokumen menggunakan bidang “Restaurantid” -nya:

```
Document filter = new Document("restaurantId", "REST-21G145");
Document result = collection.find(filter).first();
```

Meskipun menggunakan `Filters` memungkinkan untuk pemeriksaan kesalahan waktu kompilasi yang lebih baik, driver java juga memungkinkan Anda untuk menentukan `Bson` filter secara langsung dalam `find()` metode. Kode contoh berikut meneruskan `Bson` dokumen ke`find()`:

```
result = collection.find(new Document("$and", Arrays.asList(
    new Document("rating.totalReviews", new Document("$gt", 1000)),
    new Document("priceRange", "$$"))))
```

Contoh kode berikutnya menunjukkan beberapa contoh menggunakan `Filters` kelas dengan`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));
```

Contoh berikut menunjukkan bagaimana untuk rantai operasi`sort()`,`skip()`,`limit()`, dan `batchSize()` pada `FindIterable` objek. Urutan bagaimana operasi ini disediakan akan mempengaruhi kinerja kueri Anda. Sebagai praktik terbaik, urutan operasi ini harus`sort()`,, `projection()``skip()`, `limit()` dan`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);
```

Kode contoh berikut menunjukkan membuat iterator pada`FindIterable`. Ini menggunakan `forEach` konstruksi Java untuk melintasi set hasil.

```
collection.find(Filters.eq("cuisine", "American")).forEach(doc ->
 System.out.println(doc.toJson()));
```

Dalam contoh `find()` kode terakhir, ini menunjukkan bagaimana menggunakan `cursor()` untuk pengambilan dokumen. Ini menciptakan kursor di blok try yang memastikan bahwa kursor akan ditutup ketika kode keluar dari blok 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
```

Memperbarui dokumen yang ada dalam koleksi DocumentDB

Amazon DocumentDB menyediakan mekanisme yang fleksibel dan kuat untuk memodifikasi dokumen yang ada dan memasukkan yang baru ketika tidak ada. Driver MongoDB Java menawarkan beberapa metode untuk pembaruan`updateOne()`: untuk pembaruan dokumen tunggal, untuk beberapa pembaruan dokumen`updateMany()`, `replaceOne()` dan untuk penggantian dokumen lengkap. Selain ketiga metode ini,, `UpdatesUpdateOptions`, dan `UpdateResult`merupakan komponen fundamental lainnya yang menyediakan blok bangunan untuk operasi pembaruan di driver MongoDB Java.

`Updates`Kelas dalam driver MongoDB Java adalah kelas utilitas yang menyediakan metode pabrik statis untuk membuat operator pembaruan. Ini berfungsi sebagai pembangun utama untuk membangun operasi pembaruan dengan cara yang aman dan mudah dibaca. Metode dasar seperti`set()`,`unset()`, dan `inc()` memungkinkan modifikasi langsung dari dokumen. Kekuatan kelas ini menjadi jelas ketika menggabungkan beberapa operasi menggunakan `Updates.combine()` metode yang memungkinkan beberapa operasi pembaruan dijalankan secara atom, memastikan konsistensi data.

`UpdateOptions`adalah kelas konfigurasi yang kuat di driver Java MongoDB yang menyediakan kemampuan kustomisasi penting untuk operasi pembaruan dokumen. Dua aspek penting dari kelas

ini adalah menyediakan dukungan filter upsert dan array untuk operasi pembaruan. Fitur upsert, diaktifkan melaluiupsert(true), memungkinkan pembuatan dokumen baru ketika tidak ada dokumen yang cocok ditemukan selama operasi pembaruan. MelaluiarrayFilters(), operasi pembaruan dapat secara tepat memperbarui elemen array yang memenuhi kriteria tertentu.

UpdateResultdi driver Java MongoDB menyediakan mekanisme umpan balik yang merinci hasil operasi pembaruan. Kelas ini merangkum tiga metrik utama: jumlah dokumen yang cocok dengan kriteria pembaruan (), jumlah dokumen yang benar-benar dimodifikasi (matchedCount), dan informasi tentang dokumen yang diperbarui (modifiedCount). upsertedId Memahami metrik ini sangat penting untuk penanganan kesalahan yang tepat, verifikasi operasi pembaruan, dan menjaga konsistensi data dalam aplikasi.

Perbarui dan ganti satu dokumen

Di DocumentDB, memperbarui satu dokumen dapat dilakukan dengan menggunakan metode updateOne (). Metode ini mengambil parameter filter, biasanya disediakan oleh Filters kelas, untuk mengidentifikasi dokumen yang akan diperbarui, parameter Updat e yang menentukan bidang mana yang akan diperbarui, dan UpdateOptions parameter opsional untuk mengatur opsi yang berbeda untuk pembaruan. Menggunakan updateOne() metode ini hanya akan memperbarui dokumen pertama yang cocok dengan kriteria pemilihan. Kode contoh berikut memperbarui satu bidang dari satu dokumen:

```
collection.updateOne(Filters.eq("restaurantId", "REST-Y2E9H5"),
    Updates.set("name", "Amazing Japanese sushi"));
```

Untuk memperbarui beberapa bidang dalam satu dokumen, gunakan updateOne() dengan Update.combine() seperti yang ditunjukkan pada contoh berikut. Contoh ini juga menunjukkan cara menambahkan item ke array dalam dokumen.

```
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);
```

Contoh kode berikut menunjukkan cara memperbarui dokumen dalam database. Jika dokumen yang ditentukan tidak ada, operasi akan secara otomatis menyisipkannya sebagai dokumen baru. Kode ini juga menunjukkan cara menggunakan metrik yang tersedia melalui `UpdateResult` objek.

```
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)");
}
```

Contoh kode berikut menunjukkan bagaimana untuk sepenuhnya mengganti dokumen yang ada dengan dokumen baru menggunakan `replaceOne()` metode, daripada memperbarui bidang individual. `replaceOne()`Metode ini menimpa seluruh dokumen, hanya mempertahankan `_id` bidang aslinya. Jika beberapa dokumen cocok dengan kriteria filter, hanya dokumen yang ditemui pertama yang diganti.

```
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());
```

Perbarui beberapa dokumen

Ada dua cara untuk memperbarui beberapa dokumen dalam koleksi secara bersamaan. Anda dapat menggunakan `updateMany()` metode ini atau menggunakan `bulkWrite()` metode `UpdateManyModel`dengan. `updateMany()`Metode ini menggunakan parameter filter untuk memilih

dokumen untuk pembaruan, `Update` parameter untuk mengidentifikasi bidang yang akan diperbarui, dan `UpdateOptions` parameter opsional untuk menentukan opsi pembaruan.

Kode contoh berikut menunjukkan penggunaan `updateMany()` metode:

```
Bson filter = Filters.and(
    Filters.in("features", Arrays.asList("Private Dining")),
    Filters.eq("cuisine", "Thai"));
UpdateResult result1 = collection.updateMany(filter, Updates.set("priceRange", "$$$"));
```

Contoh kode berikut menunjukkan `bulkWrite()` metode menggunakan update yang sama:

```
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());
```

Menghapus dokumen dari koleksi DocumentDB

Driver MongoDB Java `deleteOne()` menawarkan untuk menghapus satu dokumen `deleteMany()` dan untuk menghapus beberapa dokumen yang sesuai dengan kriteria tertentu. Sama seperti pembaruan, operasi hapus juga dapat digunakan dengan `bulkWrite()` metode ini. Keduanya `deleteOne()` dan `deleteMany()` mengembalikan `DeleteResult`objek yang memberikan informasi tentang hasil operasi, termasuk jumlah dokumen yang dihapus. Berikut ini adalah contoh penggunaan `deleteMany()` untuk menghapus beberapa dokumen:

```
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());
```

Penanganan kesalahan dengan logika coba lagi

Strategi penanganan kesalahan yang kuat untuk Amazon DocumentDB harus menerapkan kategorisasi kesalahan menjadi dapat dicoba ulang (seperti batas waktu jaringan, masalah koneksi)

dan tidak dapat dicoba ulang (seperti kegagalan otentikasi, kueri tidak valid). Untuk kegagalan operasi karena kesalahan yang harus dicoba lagi, itu harus menerapkan penundaan waktu antara setiap percobaan ulang serta upaya percobaan ulang maksimal. Operasi CRUD harus berada di blok try-catch yang menangkap dan `MongoException`subkelasnya. Selain itu, harus mencakup pemantauan dan pencatatan kesalahan untuk visibilitas operasional. Berikut ini adalah contoh kode yang menunjukkan bagaimana menerapkan penanganan kesalahan coba lagi:

```
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);
    }
}
```

Manajemen indeks di Amazon DocumentDB dengan Java

Indeks memungkinkan pengambilan data yang efisien dari koleksi Amazon DocumentDB. Tanpa indeks, DocumentDB harus memindai setiap dokumen dalam koleksi untuk mengembalikan hasil yang memenuhi kueri yang diberikan. Topik ini memberikan informasi tentang cara membuat, menjatuhkan, dan daftar indeks menggunakan driver MongoDB Java. Ini juga membahas bagaimana menentukan apakah indeks tertentu digunakan dalam kueri dan bagaimana memberikan petunjuk ke Amazon DocumentDB untuk menggunakan indeks tertentu.

Topik

- Membuat indeks dengan Java
- Menjatuhkan indeks
- Menentukan pemilihan indeks dan memberikan petunjuk indeks

Amazon DocumentDB mendukung banyak jenis indeks. Untuk ikhtisar komprehensif dari semua indeks yang didukung, lihat posting blog ini.

Membuat indeks dengan Java

Ada dua mekanisme untuk membuat indeks di Amazon DocumentDB menggunakan `runCommand()` driver Java MongoDB: melalui, dan `createIndex()` dengan metode untuk indeks tunggal atau metode untuk beberapa indeks. `createIndexes()` Salah satu alasan untuk menggunakan `createIndexes()` metode `createIndex()` dan adalah Anda dapat membangun penanganan kesalahan yang lebih baik dengan menangkap kesalahan tertentu yang terkait dengan pembuatan indeks. Alasan lain untuk menggunakan metode ini `runCommand()` adalah bahwa driver MongDB Java menyediakan serangkaian kelas pendukung yang kaya untuk pembuatan dan manipulasi indeks. Perhatikan bahwa kelas-kelas pendukung ini hanya dapat digunakan ketika Anda menggunakan `createIndexes()` metode `createIndex()` atau. Ada tiga kelas pendukung:

- **`Indexes`**— Kelas ini berfungsi sebagai kelas utilitas yang menawarkan metode pabrik statis untuk membuat berbagai jenis indeks. Ini menyederhanakan proses pembuatan definisi indeks yang kompleks dan umumnya digunakan bersama dengan kelas terkait indeks lainnya.
- **`IndexModel`**— Ini adalah kelas fundamental yang merangkum definisi kunci indeks dan opsinya. Ini merupakan spesifikasi indeks lengkap, menggabungkan apa yang harus diindeks (kunci) dengan cara mengindeks (opsi). Kelas ini sangat berguna saat membuat beberapa indeks secara bersamaan, karena memungkinkan Anda untuk menentukan kumpulan spesifikasi indeks yang dapat diteruskan ke `createIndexes()` metode.
- **`IndexOptions`**— Ini adalah kelas konfigurasi komprehensif yang menyediakan serangkaian metode yang kaya untuk menyesuaikan perilaku indeks. Ini termasuk pengaturan untuk indeks unik, indeks jarang, waktu kedaluwarsa (TTL), dan ekspresi filter parsi. Melalui rantai metode, Anda dapat mengonfigurasi beberapa opsi seperti pembuatan indeks latar belakang dan kendala unik.

Buat indeks tunggal

Contoh ini menunjukkan cara membuat indeks tunggal menggunakan metode`createIndex()` di latar belakang. Untuk memahami latar belakang dan pembuatan indeks latar depan, lihat. Jenis build indeks Contoh kode berikut digunakan `IndexOptions`untuk membuat indeks unik dengan nama “unique_restaurantId_idx” di latar belakang. `IndexOptions`Objek ini kemudian diteruskan ke `createIndex()` metode.

```
collection.createIndex(
    Indexes.ascending("restaurantId"),
    new IndexOptions()
        .unique(true)
        .name("unique_restaurantId_idx")
        .background(true));
```

Buat beberapa indeks

Contoh ini membuat beberapa indeks menggunakan `createIndexes()` metode ini. Ini pertama membangun opsi untuk setiap indeks dengan menggunakan `IndexModel`objek dan kemudian melewati daftar `IndexModel` objek ke `createIndexes()` metode. Contoh kode berikut menunjukkan cara membuat indeks komposit dengan menggunakan kelas `Indexes`utilitas. Kelas ini juga digunakan untuk menentukan apakah Anda ingin membuat indeks menggunakan urutan urutan naik atau turun. Setelah membuat beberapa indeks, itu memverifikasi pembuatan indeks dengan memanggil metode. `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()));
```

Buat indeks yang jarang dan sebagian

Contoh ini menunjukkan menciptakan indeks jarang dan sebagian dengan membuat `IndexModel`untuk setiap jenis indeks.

```
// 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))));
```

Buat indeks teks

Contoh ini menunjukkan cara membuat indeks teks. Hanya satu indeks teks yang diizinkan pada koleksi tetapi satu indeks teks dapat menjadi indeks gabungan yang mencakup beberapa bidang. Saat menggunakan beberapa bidang dalam indeks teks, Anda juga dapat menetapkan bobot ke masing-masing bidang dalam indeks. Indeks teks pada bidang array tidak didukung oleh Amazon DocumentDB dan meskipun Anda dapat menggunakan hingga 30 bidang dalam indeks teks majemuk, hanya tiga bidang yang dapat diberi bobot.

```
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());
```

Buat indeks menggunakan **`runCommand()`**

Amazon DocumentDB mendukung pembuatan indeks paralel untuk mengurangi waktu yang dibutuhkan untuk membuat indeks. Pengindeksan paralel menggunakan beberapa pekerja bersamaan. Pekerja default yang digunakan untuk pembuatan indeks adalah dua. Posting blog ini memberikan diskusi mendalam tentang pengindeksan paralel. Saat ini, driver MongDB Java tidak mendukung menentukan opsi pekerja saat Anda menggunakan `createIndex()` atau `createIndexes()` dan oleh karena itu satu-satunya cara untuk menentukan pekerja adalah melalui. `runCommand` Contoh kode berikut menunjukkan cara menggunakan `runCommand` untuk membuat indeks yang meningkatkan pekerja menjadi empat:

```
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);
```

Menjatuhkan indeks

Driver MongoDB Java menyediakan beberapa metode untuk menjatuhkan indeks, melayani berbagai skenario dan preferensi Anda. Anda dapat menjatuhkan indeks berdasarkan nama, spesifikasi kunci, atau menjatuhkan semua indeks sekaligus. Metode `dropIndex()` dan `dropIndexes()` dapat dipanggil pada objek koleksi untuk menjatuhkan indeks. Saat menjatuhkan indeks berdasarkan nama, Anda harus memastikan mereka menggunakan nama indeks yang benar yang mungkin tidak selalu intuitif, terutama untuk indeks gabungan atau yang dihasilkan secara otomatis. Mencoba

menjatuhkan indeks yang tidak ada akan menghasilkan a. MongoCommandException `default _id`Indeks tidak dapat dijatuhkan karena memastikan keunikan dokumen dalam koleksi.

Contoh kode berikut menunjukkan menunjukkan bagaimana untuk menjatuhkan indeks dengan memberikan nama bidang di mana indeks dibuat atau dengan menghapus semua indeks:

```
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();
```

Saat menjatuhkan indeks menggunakan beberapa kunci, pastikan ada indeks gabungan yang berisi semua kunci yang ditentukan dan urutan kunci sudah benar. Kode contoh pembuatan indeks di atas menunjukkan kunci majemuk pada “masakan” dan fitur. Jika Anda mencoba menjatuhkan kunci majemuk itu tetapi urutannya bukan yang digunakan untuk pembuatan, maka terjadi MongoCommnadException kesalahan sebagai berikut:

```
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);
}
```

Kesalahan berikut ditampilkan:

```
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
```

Menentukan pemilihan indeks dan memberikan petunjuk indeks

Bekerja dengan fungsionalitas jelaskan di Amazon DocumentDB sangat penting bagi Anda untuk memahami kinerja kueri dan penggunaan indeks. Saat menjalankan kueri, Anda dapat menambahkan `explain()` metode untuk mendapatkan informasi rinci tentang rencana kueri, termasuk indeks mana, jika ada, yang sedang digunakan. `explain()`Output memberikan wawasan tentang tahapan eksekusi kueri, jumlah dokumen yang diperiksa, dan waktu yang dibutuhkan untuk setiap tahap. Informasi ini sangat berharga untuk mengidentifikasi apakah indeks tertentu digunakan secara efektif atau jika kueri dapat memperoleh manfaat dari struktur indeks yang berbeda.

`explain()`Metode ini dapat dirantai dengan `find()` metode ini. `explain()`Metode ini dapat mengambil `ExplainVerbosity`enum opsional yang menentukan tingkat verbositas yang dikembalikan oleh. `explain()` Pada saat ini, hanya `EXECUTION_STATS` dan `QUERY_PLANNER` enumerator yang didukung oleh DocumentDB. Contoh kode berikut menunjukkan cara mendapatkan perencana kueri untuk kueri tertentu:

```
// 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());
```

Dokumen JSON berikut dikembalikan untuk tingkat verbositas perencana kueri:

```
{
  "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
    }
  }
}
```

Anda memiliki beberapa opsi untuk memengaruhi atau memaksa Amazon DocumentDB menggunakan indeks tertentu. `hintString()`Metode `hint()` dan memungkinkan Anda untuk mengganti perilaku pemilihan indeks default pengoptimal kueri dengan secara eksplisit menentukan indeks mana yang harus digunakan untuk kueri. Sementara pengoptimal kueri DocumentDB umumnya membuat pilihan yang baik untuk pemilihan indeks, ada skenario di mana memaksa indeks tertentu melalui `hint()` atau `hintString()` dapat bermanfaat, seperti ketika berhadapan dengan data miring, atau menguji kinerja indeks.

Contoh kode berikut memaksa penggunaan indeks gabungan “cuisine_features_idx” untuk kueri yang sama yang dijalankan dalam kode di atas:

```
// 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 pemrograman dengan Amazon DocumentDB dan Java

Event-driven pemrograman dalam konteks Amazon DocumentDB mewakili pola arsitektur yang kuat di mana perubahan database berfungsi sebagai generator acara utama yang memicu logika dan proses bisnis berikutnya. Ketika catatan dimasukkan, diperbarui, atau dihapus dalam koleksi DocumentDB, perubahan ini bertindak sebagai peristiwa yang secara otomatis memulai berbagai proses hilir, pemberitahuan, atau tugas sinkronisasi data. Pola ini sangat berharga dalam sistem

terdistribusi modern di mana beberapa aplikasi atau layanan perlu bereaksi terhadap perubahan data secara real-time. Mekanisme utama penerapan pemrograman berbasis peristiwa di DocumentDB adalah dengan mengubah aliran.

> **Note**
>
> Panduan ini mengasumsikan Anda telah mengaktifkan aliran perubahan pada koleksi yang sedang Anda kerjakan. Lihat Menggunakan aliran perubahan dengan Amazon DocumentDB untuk mempelajari cara mengaktifkan aliran perubahan pada koleksi.

Bekerja dengan aliran perubahan dari aplikasi Java

`watch()`Metode dalam driver Java MongoDB adalah mekanisme utama untuk memantau perubahan data real-time di Amazon DocumentDB. `watch()`Metode ini dapat dipanggil oleh `MongoClient`, `MongoDatabase`, dan `MongoCollection`objek.

`watch()`Metode ini mengembalikan instance `ChangeStreamIterable`yang mendukung berbagai opsi konfigurasi, termasuk pencarian dokumen lengkap untuk pembaruan, menyediakan token resume dan stempel waktu untuk keandalan, dan tahapan agregasi pipeline untuk perubahan pemfilteran.

`ChangeStreamIterable`mengimplementasikan antarmuka inti Java `Iterable` dan dapat digunakan dengan`forEach()`. Untuk menangkap peristiwa yang digunakan`forEach()`, teruskan fungsi callback ke `forEach()` yang memproses peristiwa yang diubah. Cuplikan kode berikut menunjukkan cara membuka aliran perubahan pada koleksi untuk memulai pemantauan peristiwa perubahan:

```
ChangeStreamIterable < Document > iterator = collection.watch();
iterator.forEach(event - > {
    System.out.println("Received a change: " + event);
});
```

Cara lain untuk melintasi semua peristiwa perubahan adalah dengan membuka kursor yang memelihara koneksi ke cluster dan terus menerima peristiwa perubahan baru saat terjadi. Untuk mendapatkan kursor aliran perubahan, gunakan `cursor()` metode objek. `ChangeStreamIterable` Contoh kode berikut menunjukkan cara memantau peristiwa perubahan menggunakan kursor:

```
try (MongoChangeStreamCursor < ChangeStreamDocument < Document >> cursor =
 collection.watch().cursor()) {
    System.out.println(cursor.tryNext());
}
```

Sebagai praktik terbaik, buat `MongoChangeStreamCursor`dalam pernyataan coba-dengan-sumber daya atau tutup kursor secara manual. Memanggil `cursor()` metode pada `ChangeStreamIterable`mengembalikan `MongoChangeStreamCursor` yang dibuat di atas `ChangeStreamDocument`objek.

`ChangeStreamDocument`Kelas adalah komponen penting yang mewakili peristiwa perubahan individu dalam aliran. Ini berisi informasi rinci tentang setiap modifikasi, termasuk jenis operasi (menyisipkan, memperbarui, menghapus, mengganti), kunci dokumen, informasi namespace, dan konten dokumen lengkap bila tersedia. Kelas menyediakan metode untuk mengakses berbagai aspek dari peristiwa perubahan, seperti `getOperationType()` untuk menentukan jenis perubahan, `getFullDocument()` untuk mengakses status dokumen lengkap, dan `getDocumentKey()` untuk mengidentifikasi dokumen yang dimodifikasi.

`ChangeStreamDocument`Objek menyediakan dua informasi penting, token resume dan waktu peristiwa perubahan.

Lanjutkan token dan operasi berbasis waktu di aliran perubahan DocumentDB menyediakan mekanisme penting untuk menjaga kesinambungan dan mengelola akses data historis. Token resume adalah pengidentifikasi unik yang dihasilkan untuk setiap peristiwa perubahan, berfungsi sebagai bookmark yang memungkinkan aplikasi untuk memulai ulang pemrosesan aliran perubahan dari titik tertentu setelah pemutusan atau kegagalan. Ketika kursor aliran perubahan dibuat, ia dapat menggunakan token resume yang disimpan sebelumnya melalui `resumeAfter()` opsi, memungkinkan aliran untuk melanjutkan dari tempat yang ditinggalkannya daripada memulai dari awal atau kehilangan acara.

Time-based operasi dalam aliran perubahan menawarkan pendekatan yang berbeda untuk mengelola titik awal pemantauan peristiwa perubahan. `startAtOperationTime()`Opsi ini memungkinkan Anda untuk mulai menonton perubahan yang terjadi pada atau setelah stempel waktu tertentu. Fitur berbasis waktu ini sangat berharga dalam skenario yang membutuhkan pemrosesan data historis, pemulihan point-in-time, atau sinkronisasi antar sistem.

Contoh kode berikut mengambil peristiwa yang terkait dengan dokumen sisipan, menangkap token lanjutannya, dan kemudian menyediakan token itu untuk mulai memantau peristiwa setelah peristiwa

penyisipan. Acara ini dikaitkan dengan peristiwa pembaruan, kemudian mendapatkan waktu cluster saat pembaruan terjadi dan menggunakan stempel waktu itu sebagai titik awal untuk diproses lebih lanjut.

```
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());
  }
```

Secara default, peristiwa perubahan pembaruan tidak menyertakan dokumen lengkap dan hanya mencakup perubahan yang dibuat. Jika Anda perlu mengakses dokumen lengkap yang diperbarui, Anda dapat memanggil `fullDocument()` metode pada `ChangeStreamIterable`objek. Perlu diingat bahwa ketika Anda meminta dokumen lengkap dikembalikan untuk acara pembaruan, ia mengembalikan dokumen yang ada pada saat panggilan untuk mengubah aliran dibuat.

Metode ini mengambil `FullDocument`enum sebagai parameter. Saat ini, Amazon DocumentDB hanya mendukung DEFAULT dan nilai. UPDATE_LOOKUP Cuplikan kode berikut menunjukkan cara meminta dokumen lengkap untuk acara pembaruan saat mulai melihat perubahan:

```
try (MongoChangeStreamCursor < ChangeStreamDocument < Document >> cursor =
 collection.watch().fullDocument(FullDocument.UPDATE_LOOKUP).cursor())
```

Menggunakan validasi skema JSON

Menggunakan operator kueri `$jsonSchema` evaluasi, Anda dapat memvalidasi dokumen yang dimasukkan ke dalam koleksi Anda.

Topik

- Membuat dan menggunakan validasi skema JSON
- Kata kunci yang didukung
- memintas DocumentValidation
- Batasan

Membuat dan menggunakan validasi skema JSON

Membuat koleksi dengan validasi skema

Anda dapat membuat koleksi dengan aturan `createCollection` operasi dan validasi. Aturan validasi ini diterapkan selama penyisipan atau pembaruan dokumen Amazon DocumentDB. Contoh kode berikut menunjukkan aturan validasi untuk koleksi karyawan:

```
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"
} )
```

Menyisipkan dokumen yang valid

Contoh berikut menyisipkan dokumen yang sesuai dengan aturan validasi skema di atas:

```
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})
```

Menyisipkan dokumen yang tidak valid

Contoh berikut menyisipkan dokumen yang tidak sesuai dengan aturan validasi skema di atas. Dalam contoh ini, nilai EmployeeId bukan string:

```
db.employees.insert({
    "name" : { "firstName" : "Carol" , "lastName" : "Smith"},
    "employeeId": 720 ,
    "salary": 1000.0
})
```

Contoh ini menunjukkan sintaks yang salah dalam dokumen.

Memodifikasi koleksi

`collMod`Perintah ini digunakan untuk menambah atau memodifikasi aturan validasi koleksi yang ada. Contoh berikut menambahkan bidang gaji ke daftar bidang wajib:

```
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
      }
   }
} )
```

Mengalamatkan dokumen yang ditambahkan sebelum aturan validasi diubah

Untuk menangani dokumen yang ditambahkan ke koleksi Anda sebelum aturan validasi diubah, gunakan `validationLevel` pengubah berikut:

- ketat: Menerapkan aturan validasi pada semua sisipan dan pembaruan.
- moderat: Menerapkan aturan validasi ke dokumen valid yang ada. Selama pembaruan, dokumen tidak valid yang ada tidak diperiksa.

Dalam contoh berikut, setelah memperbarui aturan validasi pada koleksi bernama “karyawan”, bidang gaji diperlukan. Memperbarui dokumen berikut akan gagal:

```
db.runCommand({
    update: "employees",
    updates: [{
        q: { "employeeId": "c721a" },
        u: { age: 25 , salary : 1000},
        upsert: true }]
})
```

Amazon DocumentDB mengembalikan output berikut:

```
{
"n" : 0,
    "nModified" : 0,
    "writeErrors" : [
        {
"index" : 0,
            "code" : 121,
            "errmsg" : "Document failed validation"
        }
    ],
    "ok" : 1,
    "operationTime" : Timestamp(1234567890, 1)
}
```

Memperbarui tingkat validasi `moderate` akan memungkinkan dokumen di atas diperbarui dengan sukses:

```
db.runCommand({
    "collMod" : "employees",
```

```
    validationLevel : "moderate"
})

db.runCommand({
    update: "employees",
    updates: [{
        q: { "employeeId": "c721a" },
        u: { age: 25 , salary : 1000},
        upsert: true }]
})
```

Amazon DocumentDB mengembalikan output berikut:

```
{
"n" : 1,
    "nModified" : 1,
    "ok" : 1,
    "operationTime" : Timestamp(1234567890, 1)
}
```

Mengambil dokumen dengan $JSONSchema

`$jsonSchema`Operator dapat digunakan sebagai filter untuk menanyakan dokumen yang cocok dengan skema JSON. Ini adalah operator tingkat atas yang dapat hadir dalam dokumen filter sebagai bidang tingkat atas atau digunakan dengan operator kueri seperti`$and`,`$or`, dan`$nor`. Contoh berikut menunjukkan penggunaan $JSONSchema sebagai filter individual dan dengan operator filter lainnya:

Dokumen dimasukkan ke dalam koleksi “karyawan”:

```
{ "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 }
```

Koleksi disaring dengan `$jsonSchema` operator saja:

```
db.employees.find({
       $jsonSchema: { required: ["age"] } })
```

Amazon DocumentDB mengembalikan output berikut:

```
{ "_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 }
```

Koleksi disaring dengan `$jsonSchema` operator dan operator lain:

```
db.employees.find({
       $or: [{ $jsonSchema: { required: ["age", "name"]}},
            { salary: { $lte:1000}}]});
```

Amazon DocumentDB mengembalikan output berikut:

```
{ "_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 }
```

Koleksi disaring dengan `$jsonSchema` operator dan dengan `$match` dalam filter agregat:

```
db.employees.aggregate(
    [{ $match: {
        $jsonSchema: {
            required: ["name", "employeeId"],
            properties: {"salary" :{"bsonType": "double"}}
        }
       }
    }]
)
```

Amazon DocumentDB mengembalikan output berikut:

```
{
"_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
}
```

Melihat aturan validasi yang ada

Untuk melihat aturan validasi yang ada pada koleksi, gunakan:

```
db.runCommand({
    listCollections: 1,
    filter: { name: 'employees' }
})
```

Amazon DocumentDB mengembalikan output berikut:

```
{
    "waitedMS" : NumberLong(0),
    "cursor" : {
        "firstBatch" : [
            {
                "name" : "employees",
                "type" : "collection",
                "options" : {
                    "autoIndexId" : true,
                    "capped" : false,
                    "validator" : {
```

```
                    "$jsonSchema" : {
                        "bsonType" : "object",
                        "title" : "employee validation",
                        "required" : [
                            "name",
                            "employeeId",
                            "salary"
                        ],
                        "properties" : {
                            "name" : {
                                "bsonType" : "object",
                                "properties" : {
                                    "firstName" : {
                                        "bsonType" : [
                                            "string"
                                        ]
                                    },
                                    "lastName" : {
                                        "bsonType" : [
                                            "string"
                                        ]
                                    }
                                },
                                "additionalProperties" : false
                            },
                            "employeeId" : {
                                "bsonType" : "string",
                                "description" : "Unique Identifier for employee"
                            },
                            "salary" : {
                                "bsonType" : "double"
                            },
                            "age" : {
                                "bsonType" : "number"
                            }
                        },
                        "additionalProperties" : true
                    }
                },
                "validationLevel" : "moderate",
                "validationAction" : "error"
            },
            "info" : {
                "readOnly" : false
```

```
                },
                "idIndex" : {
                    "v" : 2,
                    "key" : {
                        "_id" : 1
                    },
                    "name" : "_id_",
                    "ns" : "test.employees"
                }
            }
        ],
        "id" : NumberLong(0),
        "ns" : "test.$cmd.listCollections"
    },
    "ok" : 1,
    "operationTime" : Timestamp(1692788937, 1)
}
```

Amazon DocumentDB juga mempertahankan aturan validasi dalam tahap agregasi. `$out`

Kata kunci yang didukung

Bidang berikut didukung dalam `collMod` perintah `create` dan:

- **`Validator`**— `$jsonSchem` Mendukung operator.
- **`ValidationLevel`**— Mendukung`off`,`strict`, dan `moderate` nilai-nilai.
- **`ValidationAction`**— Mendukung `error` nilainya.

Operator $JSONSchema mendukung kata kunci berikut:

- `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

memintas DocumentValidation

Amazon DocumentDB bypassDocumentValidation mendukung perintah dan metode berikut:

- insert
- update
- findAndModify
- $outtahap dalam aggregate perintah dan dalam db.collection.aggregate() metode

Amazon DocumentDB tidak mendukung perintah berikut untuk: bypassDocumentValidation

- `$merge`dalam `aggregate` perintah dan dalam `db.collection.aggregate()` metode
- `mapReduce`perintah dan `db.collection.mapReduce()` metode
- Perintah `applyOps`

Batasan

Batasan berikut berlaku untuk `$jsonSchema` validasi:

- Amazon DocumentDB mengembalikan kesalahan “Dokumen gagal validasi” ketika operasi gagal dalam aturan validasi.
- Cluster elastis Amazon DocumentDB tidak mendukung. `$jsonSchema`

Kuota dan batas Amazon DocumentDB

Topik ini menjelaskan kuota sumber daya, batas, dan batasan penamaan untuk Amazon DocumentDB (dengan kompatibilitas MongoDB).

Untuk fitur pengelolaan tertentu, Amazon DocumentDB menggunakan teknologi operasional yang dibagi dengan Amazon Relational Database Service (Amazon RDS) dan Amazon Neptune.

Topik

- Tipe instans yang didukung
- Wilayah yang didukung
- Kuota regional
- Batas agregasi
- Batas cluster
- Batas instans
- Kendala penamaan
- Kendala TTL
- Batas cluster elastis
- Batas pecahan cluster elastis
- CPU cluster elastis, memori, koneksi, dan batas kursor per pecahan

Tipe instans yang didukung

Amazon DocumentDB mendukung instans sesuai permintaan dan jenis instans berikut:

- NVMe-backed:
  - Jenis contoh R6GD:`db.r6gd.xlarge`,,,, `db.r6gd.2xlargedb.r6gd.4xlarge`,`db.r6gd.8xlarge`. `db.r6gd.12xlarge` `db.r6gd.16xlarge`
- Memori dioptimalkan:
  - Jenis contoh R8G:`db.r8g.large`,,`db.r8g.xlarge`,`db.r8g.2xlarge`,`db.r8g.4xlarge`, `db.r8g.8xlargedb.r8g.12xlarge`,`db.r8g.16xlarge`. `db.r8g.24xlarge`

  - Jenis contoh R6G:`db.r6g.large`,,`db.r6g.xlarge`,,`db.r6g.2xlarge`, `db.r6g.4xlargedb.r6g.8xlarge`,`db.r6g.12xlarge`. `db.r6g.16xlarge`
  - Jenis contoh R5:`db.r5.large`,,`db.r5.xlarge`,`db.r5.2xlarge`,`db.r5.4xlarge`, `db.r5.8xlargedb.r5.12xlarge`,`db.r5.16xlarge`.
  - Tipe instans R4: `db.r4.large`,`db.r4.2xlarge`,`db.r4.4xlarge`,`db.r4.8xlarge`,`db.r4.16xlarge`.
- Kinerja yang bisa meledak:
  - Jenis instans T4G:`db.t4g.medium`.
  - Tipe instans T3: `db.t3.medium`.

Untuk informasi lebih lanjut tentang jenis instans yang didukung dan spesifikasi mereka, lihat Spesifikasi kelas instans.

Wilayah yang didukung

Amazon DocumentDB tersedia di wilayah berikut: AWS

Nama Wilayah	Region	Availability Zone (penghitungan)
AS Timur (Ohio)	`us-east-2`	3
AS Timur (Virginia Utara)	`us-east-1`	6
AS Barat (Oregon)	`us-west-2`	4
Africa (Cape Town)	`af-south-1`	3
Amerika Selatan (Sao Paulo)	`sa-east-1`	3
Asia Pasifik (Hong Kong)	`ap-east-1`	3
Asia Pasifik (Hyderabad)	`ap-south-2`	3
Asia Pasifik (Malaysia)	`ap-southeast-5`	3
Asia Pasifik (Mumbai)	`ap-south-1`	3
Asia Pasifik (Osaka)	`ap-northeast-3`	3
Asia Pasifik (Seoul)	`ap-northeast-2`	4
Asia Pasifik (Singapura)	`ap-southeast-1`	3
Asia Pasifik (Sydney)	`ap-southeast-2`	3
Asia Pasifik (Jakarta)	`ap-southeast-3`	3

Nama Wilayah	Region	Availability Zone (penghitungan)
Asia Pacific (Melbourne)	ap-southeast-4	3
Asia Pasifik (Thailand)	ap-southeast-7	3
Asia Pasifik (Tokyo)	ap-northeast-1	3
Kanada (Pusat)	ca-central-1	3
Kanada Barat (Calgary)	ca-west-1	3
Wilayah Tiongkok (Beijing)	cn-north-1	3
Tiongkok (Ningxia)	cn-northwest-1	3
Eropa (Frankfurt)	eu-central-1	3
Europe (Zurich)	eu-central-2	3
Eropa (Irlandia)	eu-west-1	3
Eropa (London)	eu-west-2	3
Europe (Milan)	eu-south-1	3
Eropa (Paris)	eu-west-3	3
Eropa (Spanyol)	eu-south-2	3
Eropa (Stockholm)	eu-north-1	3
Meksiko (Tengah)	mx-central-1	3
Timur Tengah (UAE)	me-central-1	3
Israel (Tel Aviv)	il-central-1	3

Nama Wilayah	Region	Availability Zone (penghitungan)
AWS GovCloud (US-West)	`us-gov-west-1`	3
AWS GovCloud (US-East)	`us-gov-east-1`	3

Kuota regional

Untuk fitur pengelolaan tertentu, Amazon DocumentDB menggunakan teknologi operasional yang dibagi dengan Amazon Relational Database Service (Amazon RDS). Tabel berikut berisi batas regional yang dibagi antara Amazon DocumentDB dan Amazon RDS.

> **Note**
>
> Teknologi bersama Amazon RDS yang dijelaskan di atas hanya berlaku untuk cluster berbasis instans Amazon DocumentDB. Cluster elastis Amazon DocumentDB tidak berbagi teknologi dengan Amazon RDS.

Batasan berikut berlaku untuk klaster berbasis instans Amazon DocumentDB dan per akun per wilayah. AWS

Sumber daya	AWS batas default
klaster	40
Grup parameter klaster	50
Cluster per kelompok parameter	100
Contoh per grup parameter	300
Langganan kejadian	20
Contoh	40

Sumber daya	AWS batas default
Snapshot klaster manual	100
Replika baca per klaster	15
Grup Subnet	50
Subnet per grup subnet	20
Tag per sumber daya	50
Grup keamanan VPC per instans	5

Batasan berikut berlaku untuk cluster elastis Amazon DocumentDB dan per akun per wilayah. AWS

Sumber daya	AWS batas default
Cluster elastis	20
Cluster elastis vCPU	1024
Cuplikan cluster elastis manual	20

Anda dapat menggunakan Service Quotas untuk meminta kenaikan kuota, jika kuota dapat disesuaikan. Beberapa permintaan diselesaikan secara otomatis, sementara yang lain dikirimkan ke Dukungan. Anda dapat melacak status permintaan peningkatan kuota yang dikirimkan Dukungan. Permintaan untuk meningkatkan service quotas tidak menerima prioritas dukungan. Jika Anda memiliki permintaan mendesak, hubungi Dukungan. Untuk informasi lebih lanjut tentang Service Quotas, lihat Apa Itu Service Quotas?

Untuk meminta peningkatan kuota untuk Amazon DocumentDB:

1. Buka konsol Service Quotas di https://console.aws.amazon.com/servicequotas dan, jika perlu, masuk.
2. Di panel navigasi, pilih Layanan AWS .
3. Pilih Amazon DocumentDB (dengan kompatibilitas MongoDB) atau Amazon DocumentDB Elastic Cluster dari daftar, atau ketik di kolom pencarian.

4. Jika kuota dapat disesuaikan, Anda dapat memilih tombol radio atau namanya, lalu pilih Minta peningkatan kuota dari kanan atas halaman.
5. Untuk Ubah nilai kuota, masukkan nilai baru. Nilai baru lebih besar dari nilai saat ini.
6. Pilih Permintaan. Setelah permintaan diselesaikan, Nilai kuota yang diterapkanuntuk kuota tersebut diatur ke nilai baru.
7. Untuk melihat permintaan yang tertunda atau baru diselesaikan, pilih Dasbor dari panel navigasi. Untuk permintaan yang tertunda, pilih status permintaan untuk membuka tanda terima permintaan. Status awal dari permintaan adalah `Pending`. Setelah status berubah`Quota requested`, Anda akan melihat nomor kasus dengan Dukungan. Pilih nomor kasus untuk membuka tiket untuk permintaan Anda.

Batas agregasi

Tabel berikut menjelaskan batas agregasi di Amazon DocumentDB.

Sumber daya	Kuota
Jumlah maksimum tahapan yang didukung	500

Batas cluster

Tabel berikut menjelaskan batas klaster berbasis instans Amazon DocumentDB.

Sumber daya	Kuota
Ukuran klaster (jumlah dari semua koleksi dan indeks)	128 TiB
Ukuran koleksi (jumlah semua koleksi tidak dapat melebihi batas klaster) — tidak termasuk ukuran indeks	32 TiB
Koleksi per klaster	100.000
Basis Data per klaster	100.000
Ukuran basis data (jumlah dari semua basis data tidak dapat melebihi batas klaster)	128 TiB
Kedalaman nest dokumen	200 tingkat
Ukuran dokumen	16 MiB
Ukuran kunci indeks	2.048 byte
Indeks per koleksi	64
Kunci dalam indeks gabungan	32
Jumlah maksimum penulisan dalam perintah batch tunggal	100.000
Jumlah pengguna per klaster	1000

Batas instans

Tabel berikut menjelaskan batas Amazon DocumentDB per instans.

Amazon DocumentDB 8.0

Tipe instans	Memori instans (GiB)	Koneksi (semua)	Batas kursor	Buka transaksi	Koneksi (aktif)
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

Tipe instans	Memori instans (GiB)	Koneksi (semua)	Batas kursor	Buka transaksi	Koneksi (aktif)
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

* termasuk R6GD

Amazon DocumentDB 5.0

Tipe instans	Memori instans (GiB)	Koneksi (semua)	Batas kursor	Buka transaksi	Koneksi (aktif)
T3.medium	4	1000	30	50	102
T4G.medium	4	1000	30	50	102
R5.large	16	3400	450	200	1100

Tipe instans	Memori instans (GiB)	Koneksi (semua)	Batas kursor	Buka transaksi	Koneksi (aktif)
R5.xlarge	32	7000	450	400	2700
R5.2xlarge	64	14200	450	800	4500
R5.4xlarge	128	28400	760	1600	4500
R5.8xlarge	256	60000	1520	3200	4500
R5.12xlarge	384	60000	2280	4800	4500
R5.16xlarge	512	60000	3040	6400	4500
R6G.large*	16	3400	450	200	1100
R6G.xlarge*	32	7000	450	400	2700
R6G.2xlarge*	64	14200	450	800	4500
R6G.4xlarge*	128	28400	760	1600	4500
R6G.8xlarge*	256	60000	1520	3200	4500
R6G.12xlarge*	384	60000	2280	4800	4500
R6G.16xlarge*	512	60000	3040	6400	4500
R8G.large	16	3400	450	200	1100
R8G.xlarge	32	7000	450	400	2700
R8G.2xlarge	64	14200	450	800	4500
R8G.4xlarge	128	28400	760	1600	4500

Tipe instans	Memori instans (GiB)	Koneksi (semua)	Batas kursor	Buka transaksi	Koneksi (aktif)
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

* termasuk R6GD

Amazon DocumentDB 4.0

Tipe instans	Memori instans (GiB)	Koneksi (semua)	Batas kursor	Buka transaksi	Koneksi (aktif)
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

Tipe instans	Memori instans (GiB)	Koneksi (semua)	Batas kursor	Buka transaksi	Koneksi (aktif)
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

* termasuk R6GD

Amazon DocumentDB 3.6

Tipe instans	Memori instans (GiB)	Koneksi (semua)	Batas kursor	Buka transaksi	Koneksi (aktif)
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

Tipe instans	Memori instans (GiB)	Koneksi (semua)	Batas kursor	Buka transaksi	Koneksi (aktif)
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

Anda dapat memantau dan alarm pada batas per instans menggunakan CloudWatch metrik berikut. Untuk selengkapnya tentang metrik Amazon CloudWatch DocumentDB, lihat. Memantau Amazon DocumentDB dengan CloudWatch

Sumber daya	CloudWatch batas metrik	CloudWatch metrik penggunaan (maks 1 menit)	CloudWatch metrik penggunaan
Memori Instans	-	-	FreeableMemory
Koneksi (semua)	DatabaseConnectionsLimit	DatabaseConnectionsMax	DatabaseConnections
Cursors	DatabaseCursorsLimit	DatabaseCursorsMax	DatabaseCursors
Transaksi	TransactionsOpenLimit	TransactionsOpenMax	TransactionsOpen

Kendala penamaan

Tabel berikut menjelaskan kendala penamaan di Amazon DocumentDB.

Sumber daya	Batas default
Pengidentifikasi klaster	• Panjangnya adalah [1–63] huruf, angka, atau tanda hubung. • Karakter pertama harus berupa huruf. • Tidak dapat diakhiri dengan tanda hubung atau berisi dua tanda hubung berurutan. • Harus unik untuk semua cluster (di Amazon RDS, Amazon Neptune, dan Amazon DocumentDB) per akun, per Wilayah. AWS
Nama koleksi: `<col>`	Panjangnya adalah [1–57] karakter.
Nama Database: `<db>`	Panjangnya adalah [1–63] karakter.
Nama koleksi yang memenuhi syarat: `<db>.<col>`	Panjangnya adalah [3-120] karakter.
Nama indeks yang memenuhi syarat: `<db>.<col> $<index>`	Panjangnya adalah [6—377] karakter.
Nama Indeks: `<col>$<index>`	Panjangnya adalah [3—255] karakter.
Pengidentifikasi instans	• Panjangnya adalah [1–63] huruf, angka, atau tanda hubung • Karakter pertama harus berupa huruf

Sumber daya	Batas default
	• Tidak boleh diakhiri dengan tanda hubung atau berisi dua tanda hubung yang berurutan • Harus unik untuk semua instans (di Amazon RDS, Amazon Neptune, dan Amazon DocumentDB) per akun, per Wilayah. AWS

Kata sandi utama	• Panjangnya adalah [8-100] karakter ASCII yang dapat dicetak.
	• Dapat menggunakan karakter ASCII apa pun yang dapat dicetak kecuali untuk yang berikut ini:
	  • **/** (garis miring ke depan)
	  • **"** (tanda kutip ganda)
	  • **@** (simbol at)

Nama pengguna utama	• Panjangnya adalah [1-63] karakter alfanumerik. • Karakter pertamanya harus berupa huruf. • Tidak dapat berupa kata yang dicadangkan oleh mesin basis data.

Sumber daya	Batas default
Nama grup parameter	• Panjangnya adalah [1–255] karakter alfanumerik. • Karakter pertama harus berupa huruf. • Tidak boleh diakhiri dengan tanda hubung atau berisi dua tanda hubung berurutan.

Kendala TTL

Penghapusan dari indeks TTL tidak dijamin dalam jangka waktu tertentu dan merupakan upaya terbaik. Faktor-faktor seperti pemanfaatan sumber daya instans, ukuran dokumen, dan throughput keseluruhan dapat memengaruhi waktu penghapusan TTL.

Batas cluster elastis

Tabel berikut menjelaskan batas maksimum dalam cluster elastis Amazon DocumentDB.

Sumber daya	Kuota
Cluster elastis per wilayah	20
vCPU dijumlahkan di semua cluster elastis per wilayah	1024
Cuplikan cluster manual per wilayah	20
Pecahan per cluster	32
Penyimpanan per cluster (ketika data didistrib usikan secara merata oleh shard-key)	4 PiB
Koneksi ke cluster	Nilai yang lebih rendah dari 300.000 atau jumlah pecahan x batas koneksi yang terkait dengan vCPU per shard

Sumber daya	Kuota
UnSharded ukuran koleksi	32 TiB
Ukuran koleksi sharded (ketika data didistribusikan secara merata oleh shard-key)	1PB
Basis Data per klaster	10.000
UnSharded koleksi per cluster	100.000
Koleksi sharded per cluster	1000
Pengguna per cluster	100
Menulis dalam satu perintah batch	100.000
Indeks per koleksi	64
Kedalaman nest dokumen	100 tingkat
Ukuran dokumen	16MB
Ukuran kunci indeks	2048 byte
Kunci dalam indeks gabungan	32

Batas pecahan cluster elastis

Tabel berikut menjelaskan batas pecahan maksimum di cluster elastis Amazon DocumentDB.

Sumber daya	Kuota
vCPU per contoh shard	64
Contoh per pecahan	16
Penyimpanan per pecahan	128 TiB
Penyimpanan per koleksi per pecahan	32 TiB

CPU cluster elastis, memori, koneksi, dan batas kursor per pecahan

Tabel berikut menjelaskan batas maksimum CPU, memori, koneksi, dan kursor di pecahan cluster elastis Amazon DocumentDB.

vCPU per pecahan	Memori instans (GiB)	Batas koneksi	Batas kursor
2	16	1700	450
4	32	3500	450
8	64	7100	450
16	128	14200	760
32	256	28400	1520
48	384	30000	2280
64	512	30000	3040

Menanyakan di Amazon DocumentDB

Bagian ini menjelaskan semua aspek dari melakukan kueri dengan Amazon DocumentDB.

Topik

- Menanyakan dokumen
- Rencana kueri
- Jelaskan hasil
- Perencana kueri v2
- Perencana kueri v3
- Menanyakan data geospasial dengan Amazon DocumentDB
- Indeks sebagian
- Melakukan pencarian teks dengan Amazon DocumentDB

Menanyakan dokumen

Terkadang, Anda mungkin perlu mencari inventaris toko online Anda sehingga pelanggan dapat melihat dan membeli apa yang Anda jual. Melakukan kueri sebuah koleksi relatif mudah, apakah Anda ingin semua dokumen dalam koleksi atau hanya dokumen yang memenuhi kriteria tertentu.

Untuk melakukan kueri dokumen, gunakan operasi `find()`. Perintah `find()` memiliki parameter dokumen tunggal yang mendefinisikan kriteria yang akan digunakan dalam memilih dokumen yang akan dikembalikan. Output dari `find()` adalah dokumen yang diformat sebagai satu baris teks tanpa jeda baris. Untuk memformat dokumen output agar lebih mudah dibaca, gunakan `find().pretty()`. Semua contoh dalam topik ini menggunakan `.pretty()` untuk memformat output.

Contoh kode berikut menggunakan empat dokumen yang Anda sisipkan ke dalam koleksi `example` dalam dua latihan sebelumnya — `insertOne()` dan `insertMany()` yang terletak di bagian Menambahkan Dokumen pada Bekerja dengan Dokumen.

Topik

- Mengambil semua dokumen dalam koleksi
- Mengambil dokumen yang cocok dengan nilai bidang
- Mengambil dokumen yang cocok dengan dokumen yang disematkan

- Mengambil dokumen yang cocok dengan nilai bidang dalam dokumen yang disematkan
- Mengambil dokumen yang cocok dengan array
- Mengambil dokumen yang cocok dengan nilai dalam array
- Mengambil dokumen menggunakan operator

Mengambil semua dokumen dalam koleksi

Untuk mengambil semua dokumen dalam koleksi Anda, gunakan operasi `find()` dengan dokumen kueri kosong.

Kueri berikut mengembalikan semua dokumen di dalam koleksi `example`.

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

Mengambil dokumen yang cocok dengan nilai bidang

Untuk mengambil semua dokumen yang cocok dengan bidang dan nilai, gunakan operasi `find()` dengan dokumen kueri yang mengidentifikasi bidang dan nilai yang akan dicocokkan.

Menggunakan dokumen sebelumnya, kueri ini mengembalikan semua dokumen di mana bidang "Item" sama dengan "Pena".

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

Mengambil dokumen yang cocok dengan dokumen yang disematkan

Untuk menemukan semua dokumen yang cocok dengan dokumen yang ditanamkan, gunakan operasi `find()` dengan dokumen kueri yang menentukan nama dokumen yang ditanamkan dan semua bidang dan nilai untuk dokumen yang ditanamkan itu.

Saat mencocokkan dokumen yang ditanamkan, dokumen yang ditanamkan tersebut harus memiliki nama yang sama seperti dalam kueri. Selain itu, bidang dan nilai dalam dokumen yang ditanamkan harus cocok dengan kueri.

Kueri berikut hanya mengembalikan dokumen "Cat Poster". Ini karena "Pena" memiliki nilai yang berbeda untuk "`OnHand`" dan "`MinOnHand`", dan "Cat Semprot" memiliki satu bidang (`OrderQnty`) lebih banyak daripada dokumen kueri.

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

Mengambil dokumen yang cocok dengan nilai bidang dalam dokumen yang disematkan

Untuk menemukan semua dokumen yang cocok dengan dokumen yang ditanamkan, gunakan operasi `find()` dengan dokumen kueri yang menentukan nama dokumen yang ditanamkan dan semua bidang dan nilai untuk dokumen yang ditanamkan itu.

Mengingat dokumen sebelumnya, kueri berikut menggunakan "notasi titik" untuk menentukan dokumen yang ditanamkan dan bidang yang diminati. Setiap dokumen yang cocok dengan ini dikembalikan, terlepas dari apa bidang lain mungkin hadir dalam dokumen tertanam. Kueri mengembalikan "Cat Poster" dan "Cat Semprot" karena keduanya cocok dengan bidang dan nilai yang ditentukan.

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

Mengambil dokumen yang cocok dengan array

Untuk menemukan semua dokumen yang cocok dengan array, gunakan operasi `find()` dengan nama array yang Anda minati dan semua nilai dalam array itu. Kueri mengembalikan semua dokumen yang memiliki array dengan nama itu di mana nilai arraynya identik dengan dan dalam urutan yang sama seperti dalam kueri.

Kueri berikut hanya mengembalikan "Pena" karena "Cat Poster" memiliki warna tambahan (Putih), dan "Cat Semprot" memiliki warna dalam urutan yang berbeda.

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

Mengambil dokumen yang cocok dengan nilai dalam array

Untuk menemukan semua dokumen yang memiliki nilai array tertentu, gunakan operasi `find()` dengan nama array dan nilai yang Anda minati.

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

Operasi sebelumnya mengembalikan ketiga dokumen karena masing-masing memiliki array bernama `Colors` dan nilai "`Red`" di suatu tempat dalam array. Jika Anda menentukan nilai "`White`", kueri hanya akan mengembalikan "Cat Poster."

Mengambil dokumen menggunakan operator

Kueri berikut mengembalikan semua dokumen dengan nilai "`Inventory.OnHand`" kurang dari 50.

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

Untuk daftar operator kueri yang didukung, lihat Operator kueri dan proyeksi.

Rencana kueri

Bagaimana saya bisa melihat **`ExecutionStats`** untuk Query Plan?

Saat menentukan mengapa kueri dijalankan lebih lambat dari yang diharapkan, akan berguna untuk memahami apa `executionStats` untuk rencana kueri. `executionStats` memberikan jumlah dokumen yang dikembalikan dari tahap tertentu (`nReturned`), jumlah waktu eksekusi yang dihabiskan pada setiap tahap (`executionTimeMillisEstimate`), dan jumlah waktu yang diperlukan untuk menghasilkan rencana kueri (`planningTimeMillis`). Anda dapat menentukan tahapan kueri yang paling memakan waktu untuk membantu memfokuskan upaya pengoptimalan Anda dari output`executionStats`, seperti yang ditunjukkan dalam contoh kueri berikut. Parameter `executionStats` saat ini tidak mendukung perintah `update` dan `delete`.

> **Note**
>
> Amazon DocumentDB mengemulasi MongoDB 3.6 API pada mesin basis data yang dibuat khusus yang menggunakan sistem penyimpanan terdistribusi, toleran terhadap kesalahan, dan dapat pulih sendiri. Akibatnya, rencana kueri dan keluaran dari `explain()` mungkin berbeda antara Amazon DocumentDB dan MongoDB. Pelanggan yang ingin kontrol atas rencana kueri mereka dapat menggunakan operator `$hint` untuk memaksa pemilihan indeks yang diutamakan.

Jalankan kueri yang ingin Anda tingkatkan di bawah perintah `explain()` sebagai berikut.

```
db.runCommand({explain: {query document}}).
explain("executionStats").executionStats;
```

Berikut ini adalah contoh operasi.

```
db.fish.find({}).limit(2).explain("executionStats");
```

Output dari operasi ini terlihat seperti berikut.

```
{
    "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
}
```

Jika Anda tertarik hanya melihat `executionStats` dari kueri di atas, Anda dapat menggunakan perintah berikut. Untuk koleksi kecil, prosesor kueri Amazon DocumentDB dapat memilih untuk tidak menggunakan indeks jika peningkatan performa dapat diabaikan.

```
db.fish.find({}).limit(2).explain("executionStats").executionStats;
```

Cache rencana kueri

Untuk mengoptimalkan performa dan mengurangi durasi perencanaan, Amazon DocumentDB melakukan cache rencana kueri secara internal. Ini memungkinkan kueri dengan bentuk yang sama untuk dieksekusi secara langsung menggunakan rencana cache.

Namun, cache ini terkadang menyebabkan penundaan acak untuk kueri yang sama; misalnya, kueri yang biasanya membutuhkan waktu satu detik untuk dijalankan terkadang membutuhkan waktu sepuluh detik. Ini karena seiring waktu, instans pembaca melakukan cache berbagai bentuk kueri, sehingga menghabiskan memori. Jika Anda mengalami kelambatan acak ini, tidak ada tindakan yang perlu Anda lakukan untuk melepaskan memori--sistem akan mengelola penggunaan memori untuk Anda dan begitu memori mencapai ambang batas tertentu, memori akan dilepaskan secara otomatis.

Jelaskan hasil

Jika Anda ingin mengembalikan informasi tentang rencana kueri, Amazon DocumentDB mendukung mode verbositas `queryPlanner`. Hasil `explain` mengembalikan rencana kueri terpilih yang dipilih oleh pengoptimal dalam format yang mirip dengan berikut ini:

```
{
   "queryPlanner" : {
      "plannerVersion" : <int>,
      "namespace" : <string>,
```

```
      "winningPlan" : {
         "stage" : <STAGE1>,
         ...
         "inputStage" : {
            "stage" : <STAGE2>,
            ...
            "inputStage" : {
               ...
            }
         }
      }
   }
}
```

Bagian berikut akan menentukan hasil `explain` umum.

Topik

- Tahap pemindaian dan filter
- Persimpangan indeks
- Serikat indeks
- Beberapa indeks intersection/union
- Indeks majemuk
- Sortir tahap
- Panggung grup

Tahap pemindaian dan filter

Pengoptimal dapat memilih salah satu pemindaian berikut:

COLLSCAN

Tahap ini adalah pemindaian koleksi sekuensial.

```
{
    "stage" : "COLLSCAN"
}
```

IXSCAN

Tahap ini memindai kunci indeks. Pengoptimal dapat mengambil dokumen dalam tahap ini dan ini dapat mengakibatkan tahap FETCH ditambahkan kemudian.

```
db.foo.find({"a": 1})
{
    "stage" : "IXSCAN",
    "direction" : "forward",
    "indexName" : <idx_name>
}
```

FETCH

Jika pengoptimal mengambil dokumen dalam tahap selain IXSCAN, hasilnya akan menyertakan tahap FETCH. Misalnya, kueri IXSCAN di atas dapat menghasilkan kombinasi tahapan FETCH dan IXSCAN:

```
db.foo.find({"a": 1})
{
    "stage" : "FETCH",
    "inputStage" : {
        "stage" : "IXSCAN",
        "indexName" : <idx_name>
    }
}
```

IXONLYSCAN hanya memindai kunci indeks. Buat indeks gabungan tidak akan menghindari FETCH.

Persimpangan indeks

IXAND

Amazon DocumentDB dapat menyertakan tahap IXAND dengan array inputStages IXSCAN jika dapat memanfaatkan persimpangan indeks. Misalnya, kita mungkin melihat output seperti:

```
{
    "stage" : "FETCH",
    "inputStage" : {
        "stage" : "IXAND",
```

```
        "inputStages" : [
            {
                "stage" : "IXSCAN",
                "indexName" : "a_1"
            },
            {
                "stage" : "IXSCAN",
                "indexName" : "b_1"
            }
        ]
    }
}
```

Serikat indeks

IXOR

Mirip dengan persimpangan indeks, Amazon DocumentDB dapat menyertakan tahap `IXOR` dengan array `inputStages` untuk operator `$or`.

```
db.foo.find({"$or": [{"a": {"$gt": 2}}, {"b": {"$lt": 2}}]})
```

Untuk kueri di atas, output yang dijelaskan mungkin terlihat seperti ini:

```
{
    "stage" : "FETCH",
    "inputStage" : {
        "stage" : "IXOR",
        "inputStages" : [
            {
                "stage" : "IXSCAN",
                "indexName" : "a_1"
            },
            {
                "stage" : "IXSCAN",
                "indexName" : "b_1"
            }
        ]
    }
}
```

Beberapa indeks intersection/union

Amazon DocumentDB dapat menggabungkan beberapa tahap persimpangan atau persatuan indeks bersama-sama kemudian mengambil hasilnya. Contoh:

```
{
    "stage" : "FETCH",
    "inputStage" : {
        "stage" : "IXOR",
        "inputStages" : [
            {
                "stage" : "IXSCAN",
                ...
            },
            {
                "stage" : "IXAND",
                "inputStages" : [
                    {
                        "stage" : "IXSCAN",
                        ...
                    },
                    {
                        "stage" : "IXSCAN",
                        ...
                    }
                ]
            }
        ]
    }
}
```

Penggunaan tahap indeks persimpangan atau persatuan tidak dipengaruhi oleh jenis indeks (jarang, majemuk, dll).

Indeks majemuk

Penggunaan indeks gabungan Amazon DocumentDB tidak dibatasi pada subset awal bidang indeks; itu dapat menggunakan indeks dengan bagian akhiran tetapi mungkin tidak terlalu efisien.

Misalnya, indeks gabungan `{ a: 1, b: -1 }` dapat mendukung ketiga kueri di bawah ini:

`db.orders.find( { a: 1 } )`

```
db.orders.find( { b: 1 } )

db.orders.find( { a: 1, b: 1 } )
```

Sortir tahap

Jika ada indeks pada kunci sortir yang diminta, Amazon DocumentDB dapat menggunakan indeks untuk mendapatkan pesanan. Dalam hal ini, hasilnya tidak akan mencakup tahap `SORT`, melainkan tahap `IXSCAN`. Jika pengoptimal menyukai pengurutan biasa, itu akan mencakup tahap seperti ini:

```
{
    "stage" : "SORT",
    "sortPattern" : {
        "a" : 1,
        "b" : -1
    }
}
```

Panggung grup

Amazon DocumentDB mendukung dua strategi grup yang berbeda:

- `SORT_AGGREGATE`: Pada agregat pengurutan disk.
- `HASH_AGGREGATE`: Dalam agregat hash memori.

Perencana kueri v2

Perencana kueri baru untuk Amazon DocumentDB (perencana versi 2.0) memiliki kemampuan pengoptimalan kueri tingkat lanjut dan peningkatan kinerja. Planner versi 2.0 untuk Amazon DocumentDB 5.0 memberikan peningkatan kinerja hingga 10x dibandingkan versi `find` sebelumnya saat menggunakan dan operator dengan indeks. `update` Peningkatan kinerja terutama berasal dari penggunaan rencana indeks yang lebih optimal dan memungkinkan dukungan pemindaian indeks untuk operator seperti operator negasi (`$neq`,`$nin`) dan bersarang. `$elementMatch` Kueri perencana versi 2.0 berjalan lebih cepat melalui teknik estimasi biaya yang lebih baik, algoritme yang dioptimalkan, dan stabilitas yang ditingkatkan. Planner versi 2.0 juga mendukung API filter cache paket, yang meningkatkan stabilitas perencana. Dengan fitur ini, Amazon DocumentDB 5.0 sekarang menawarkan kemampuan untuk memilih dari berbagai versi perencana kueri.

Topik

- Prasyarat
- Memilih perencana versi 2.0 sebagai perencana kueri default
- Praktik terbaik
- Batasan
- Perbaikan untuk Menemukan dan Memperbarui Operator
- Rencanakan API filter cache
- Potensi perbedaan perilaku antara perencana versi 1.0, 2.0, dan MongoDB
- Planner versi 2.0 menjembatani kesenjangan perilaku dengan MongoDB

Prasyarat

Prasyarat berikut berlaku untuk perencana versi 2.0:

- Planner versi 2.0 tersedia di semua wilayah di mana engine versi 5.0 tersedia.
- Untuk memilih menggunakan versi 2.0 sebagai perencana kueri default, klaster Anda harus menggunakan engine patch versi 3.0.15902 atau yang lebih baru dari Amazon DocumentDB versi 5.0. Untuk langkah-langkah memperbarui ke patch versi mesin terbaru, lihatMelakukan pembaruan tambalan ke versi mesin cluster.
- Untuk menetapkan perencana versi 2.0 sebagai perencana kueri default, Anda memerlukan izin IAM untuk memperbarui grup parameter cluster.
- Untuk Amazon DocumentDB 8.0, perencana versi 3.0 adalah perencana kueri default. Lihat Perencana kueri v3 untuk detail.

Memilih perencana versi 2.0 sebagai perencana kueri default

Gunakan langkah-langkah berikut untuk memilih 2.0 sebagai perencana kueri default dari konsol atau CLI:

- Ikuti langkah-langkah Memodifikasi parameter cluster Amazon DocumentDB untuk memodifikasi grup parameter cluster Anda.
- Untuk parameter berjudul 'PlannerVersion', ubah nilainya menjadi 2.0 yang menunjukkan perencana versi 2.0.

- Pilih Terapkan segera (memilih Terapkan saat reboot akan membuat pilihan tidak efektif hingga reboot cluster berikutnya).

Praktik terbaik

Untuk hasil yang diharapkan, gunakan praktik terbaik berikut saat menerapkan perencana versi 2.0:

- Di cluster global, pilih `plannerVersion` nilai yang sama (1.0 atau 2.0) di grup parameter cluster untuk kedua wilayah. Perhatikan bahwa memilih versi perencana yang berbeda di wilayah primer dan sekunder dapat menyebabkan perilaku dan kinerja kueri yang tidak konsisten.
- Memperbarui ke perencana versi 2.0 selama jendela pemeliharaan terjadwal atau selama periode lalu lintas yang dikurangi akan menjadi yang paling tidak mengganggu, karena mungkin ada peningkatan tingkat kesalahan jika versi perencana diubah saat beban kerja aktif berjalan.
- Planner versi 2.0 bekerja paling optimal dengan MongoDB shell versi 5.0.

Batasan

Batasan berikut berlaku untuk perencana versi 2.0:

- Planner versi 2.0 tidak didukung dalam cluster elastis, yang akan kembali ke versi perencana 1.0.
- Perencana versi 2.0 tidak didukung untuk agregasi dan perintah yang berbeda, yang akan kembali ke perencana versi 1.0.
- Kueri yang berisi regex, pencarian teks, geospasial, jsonschema atau `$expr` dalam filter tidak didukung dengan filter cache paket di perencana versi 2.0.

Perbaikan untuk **`Menemukan`** dan **`Memperbarui`** Operator

Planner versi 2.0 mengoptimalkan operasi fundamental termasuk`find`,, `updatedelete`, dan `find-and-modify` perintah. Bagian tab berikut menunjukkan kemampuan yang ditingkatkan untuk indeks, serta peningkatan kinerja kueri dengan perencana versi 2.0:

Enhanced index support

- Planner versi 2.0 menambahkan dukungan indeks untuk operator negasi termasuk`$nin`,`$ne`,`$not {eq}`, dan`$not {in}`, serta `$type` dan. `$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 }}}}})
```

- Perencana versi 2.0 menggunakan indeks jarang atau sebagian bahkan ketika tidak `$exists` ada dalam ekspresi kueri.

```
Sample Document: {"name": "Bob", "email": "example@fake.com" }

Using Planner Version 1.0, you can specify the command as shown below:
db.foo.find({email: "example@fake.com", email: {$exists: true}})

Using Planner Version 2.0, you can specify command without $exists:
db.foo.find({ email: "example@fake.com" })
```

- Planner versi 2.0 akan menggunakan indeks sebagian bahkan ketika kondisi kueri tidak sama persis dengan ekspresi filter indeks paral.

```
Sample Document: {"name": "Bob", "age": 34}
db.foo.createIndex({"age":1},{partialFilterExpression:{"age":{$lt:50}}})

With Planner Version 1.0, index is used only when the query condition meets the
 partial
 index filter criterion:
     db.foo.find({"age":{$lt:50}})

With Planner Version 2.0, index is used even when the query condition doesn't meet
 the index
criterion:
db.foo.find({"age":{$lt:30}})
```

- Planner versi 2.0 menggunakan pemindaian indeks sebagian dengan kueri $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}}})
```

- Planner versi 2.0 menyertakan dukungan pemindaian indeks untuk`$regex`, tanpa perlu menyediakan `$hint` kode aplikasi Anda. `$regex`mendukung indeks pada pencarian awalan saja.

```
Sample Document: { "x": [1, 2, 3], "y": "apple" }
db.foo.createIndex({ "x": 1, "y": 1 })
db.foo.find({"y":{ $regex: "^a" }})
```

- Perencana versi 2.0 meningkatkan kinerja kueri yang melibatkan indeks multi-kunci, dengan kondisi kesetaraan pada bidang multi-kunci.

```
Sample Document: {"x": [1, 2, 3],
"y": 5}
db.foo.createIndex({"x": 1, "y":1})
db.foo.find({"x": 2,
"y": {$gt: 1}}).limit(1)
```

- Planner versi 2.0 meningkatkan kinerja kueri yang melibatkan beberapa filter, terutama pada koleksi dengan dokumen lebih besar dari 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}}]})
```

- Planner versi 2.0 meningkatkan kinerja kueri saat menggunakan `$in` operator dengan indeks gabungan dengan menghilangkan tahap pengurutan.

```
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})
```

  Ini juga meningkatkan kinerja kueri yang menggunakan indeks multi-kunci dengan elemen. `$in`

```
Sample Document: {"x": [1, 2, 3]}
```

```
db.foo.createIndex({"x": 1})
db.foo.find("x":{$in:[>100 elements]})
```

Query performance improvements

- Planner versi 2.0 meningkatkan kinerja kueri yang melibatkan beberapa filter, terutama pada koleksi dengan dokumen lebih besar dari 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}}]})
```

- Planner versi 2.0 meningkatkan kinerja kueri saat menggunakan `$in` operator dengan indeks gabungan dengan menghilangkan tahap pengurutan.

```
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})
```

  Ini juga meningkatkan kinerja kueri yang menggunakan indeks multikey dengan elemen. `$in`

```
Sample Document: {"x": [1, 2, 3]}
db.foo.createIndex({"x": 1})
db.foo.find("x":{$in:[>100 elements]})
```

Rencanakan API filter cache

> **Note**
>
> Indeks teks tidak didukung dengan filter cache paket.

- Perencana versi 2.0 menambahkan dukungan untuk fitur filter indeks yang memungkinkan Anda menentukan daftar indeks yang dapat digunakan oleh bentuk kueri tertentu. Fitur ini dapat diakses melalui API dan dapat dikontrol dari sisi server. Jika Anda mengalami regresi kueri, fitur ini memberi Anda opsi yang lebih cepat dan lebih fleksibel untuk mengurangi masalah tanpa harus mengubah kode aplikasi Anda.

```
db.runCommand({ planCacheSetFilter: <collection>, query: <query>,
sort: <sort>, // optional,
indexes: [ <index1>, <index2>, ...],
comment: <any> // optional})
```

  Untuk membuat daftar semua filter pada koleksi, gunakan perintah berikut:

```
db.runCommand(
{
planCacheListFilters: <collection>
}
)
```

  Perintah ini menunjukkan semua filter indeks pada koleksi. Contoh output:

```
{
"filters" : [
{
"query" : {a: "@", b: "@"},
"sort" : {a: 1},
"indexes" : [
<index1>,
...
]
},
...
],
"ok": 1
}
```

- Anda dapat menggunakan dua bidang baru dari output `explain` perintah untuk menganalisis pemfilteran indeks perencana versi 2.0: `indexFilterSet` dan. `indexFilterApplied` `indexFilterSet`disetel ke “true” jika ada filter indeks yang disetel pada koleksi yang cocok

dengan bentuk kueri. `indexFilterApplied`diatur ke “true” jika, dan hanya jika kueri diterapkan indeks filter dan memilih rencana menggunakan indeks dalam daftar filter.

Anda dapat menghapus filter indeks dengan perintah berikut:

```
db.runCommand(
{
planCacheClearFilters: <collection>>
query: <query pattern>, // optional
sort: <sort specification>, // optional
comment: <any>. //optional
}
)
```

Untuk menghapus semua filter pada koleksi “foo”, gunakan perintah berikut:

```
db.runCommand({planCacheClearFilters: "foo"})
```

Untuk menghapus bentuk kueri tertentu dengan jenis apa pun, Anda dapat menyalin dan menempelkan bentuk kueri dari output`planCacheListFilters`:

```
db.runCommand({planCacheClearFilters: "foo", query: {a: @}})
```

Untuk menghapus bentuk kueri tertentu dengan bidang tertentu untuk diurutkan berdasarkan, Anda dapat menyalin dan menempelkan bentuk kueri dari output`planCacheListFilters`:

```
db.runCommand({planCacheClearFilters: "foo", query: {a: @},sort: {a: 1}})
```

Potensi perbedaan perilaku antara perencana versi 1.0, 2.0, dan MongoDB

Dalam beberapa kasus tepi, ada kemungkinan bahwa perencana versi 2.0 dapat menghasilkan hasil yang sedikit berbeda dari hasil di MongoDB. Bagian ini membahas beberapa contoh kemungkinan ini.

$(update) and $(projection)

- Dalam beberapa kasus, `$(update)` dan `$(projection)` operator di MongoDB mungkin berperilaku berbeda dari perencana Amazon DocumentDB versi 1.0. Di bawah ini adalah beberapa contoh:

```
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 } }
```

  - Perencana versi 1.0 - Bidang Pembaruan 2022
  - MongoDB - Bidang Pembaruan 2022
  - Perencana versi 2.0 - Bidang Pembaruan 2021

- ```
  db.col.insert({x:[1,2,3]})
  db.col.update({$and:[{x:1},{x:3}]},{$set:{"x.$":500}})
  ```

  - Perencana versi 1.0 - Secara acak memperbarui elemen pertama cocok
  - MongoDB - Secara acak memperbarui elemen pertama yang cocok
  - Perencana versi 2.0 - Tidak membuat pembaruan

- ```
  db.col.insert({x:[1,2,3]})
  db.col.find()
  ```

  - Perencana versi 1.0 - Secara acak memilih elemen yang cocok
  - MongoDB - Secara acak memilih elemen yang cocok
  - Perencana versi 2.0 - Tidak membuat pilihan

- ```
  db.col.insert({x:100})
  db.col.update({x:100},{x:100})
  ```

  - Perencana versi 1.0 - nPerubahan hitungan yang dimodifikasi
  - MongoDB - NPerubahan hitungan yang dimodifikasi
  - Perencana versi 2.0 - NModified count tidak berubah ketika diperbarui dengan nilai yang sama.
  ```

- Ketika `$(update)` operator digunakan dengan`$setOnInsert`, perencana versi 1.0 dan MongoDB melempar kesalahan, tetapi perencana versi 2.0 tidak.
- Mengganti nama bidang yang tidak ada untuk `$field` memunculkan kesalahan di perencana versi 2.0, sedangkan tidak menghasilkan pembaruan di perencana versi 1.0 dan MongoDB.

Index behavior

- Planner versi 2.0 memunculkan kesalahan saat `$hint` diterapkan dengan indeks yang tidak sesuai, sedangkan perencana versi 1.0 dan MongoDB tidak.

```
// 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"
}
```

- $neartidak dapat digunakan $hint({“$natural”:1}) dalam perencana versi 2.0.

```
// 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" }
```

- Sementara MongoDB mendukung pemindaian indeks regex lengkap, perencana versi 2.0 mendukung pemindaian indeks regex pada bidang awalan saja.

```
// 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)
```

- Ada beberapa perbedaan yang melekat dalam menggunakan filter cache paket dengan perencana versi 2.0 dibandingkan dengan MongoDB. Sementara perencana versi 2.0 tidak mendukung menentukan “proyeksi” dan “pemeriksaan” dengan filter cache paket, MongoDB melakukannya. Namun, filter indeks MongoDB hanya dalam memori dan hilang setelah restart. Perencana versi 2.0 mempertahankan filter indeks melalui restart dan tambalan.

Others

- Format log audit DHTML saat menggunakan perencana versi 2.0 sedikit berbeda dari versi perencana 1.0.

```
command -  db.col.find({x:1})

************** Audit logs generated ******************

// v1 format for dml audit logs
{"atype":"authCheck","ts":1746473479983,"timestamp_utc":"2025-05-05
 19:31:19.983","remote_ip":"127.0.0.1:47022","users":
[{"user":"serviceadmin","db":"test"}],"param":
{"command":"find","ns":"test.col","args":{"batchSize":101,"filter":
{"x":1},"find":"col","limit":18446744073709551615,"lsid":
```

```
{"id":{"$binary":"P6RCGz9ZS4iWBSSHWXW15A==","$type":"4"},"uid":
{"$binary":"6Jo8PisnEi3dte03+pJFjdCyn/5cGQL8V2KqaoWsnk8=","$type":"0"}},"maxScan":18446744

// v2 formal for dml audit logs
{"atype":"authCheck","ts":1746473583711,"timestamp_utc":"2025-05-05
 19:33:03.711","remote_ip":"127.0.0.1:37754","users":
[{"user":"serviceadmin","db":"test"}],"param":
{"command":"find","ns":"test.col","args":{"find":"col","filter":{"x":1},"lsid":
{"id":{"$binary":"nJ88TGCSSd
+BeD2+ZtrhQg==","$type":"4"}},"$db":"test"},"result":0}}
```

- Kondisi indeks sebagai bagian dari rencana penjelasan:

```
rs0:PRIMARY> db.col.createIndex({index1:1})
{
 "createdCollectionAutomatically" : false,
 "numIndexesBefore" : 1,
 "numIndexesAfter" : 2,
 "ok" : 1,
 "operationTime" : Timestamp(1761149251, 1)
}
```

Perencana versi 2.0 menjelaskan output rencana yang menampilkan kondisi indeks dan filter:

```
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)
}
```

Perencana versi 1.0 menjelaskan keluaran rencana:

```
rs0:PRIMARY> db.col.find({$and:[{price:{$eq:300}},{item:
{$eq:"apples"}}]}).explain()
{
 "queryPlanner" : {
  "plannerVersion" : 1,
  "namespace" : "test.col",
  "winningPlan" : {
   "stage" : "IXSCAN",
   "indexName" : "price_1",
   "direction" : "forward"
  }
 },
 "ok" : 1,
 "operationTime" : Timestamp(1761149533, 1)
}
```

Planner versi 2.0 menjembatani kesenjangan perilaku dengan MongoDB

Ada beberapa area di mana perencana versi 2.0 menutup kesenjangan perilaku dari MongoDB:

- Planner versi 2.0 memungkinkan pencarian indeks numerik pada array yang diratakan untuk: `$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}}})
```

- Sementara perencana versi 1.0 mengecualikan string dalam proyeksi, perilaku perencana versi 2.0 sejajar dengan MongoDB dan memperlakukannya sebagai nilai literal”

```
// Planner V2/ MongoDB
> db.col.find()
{ "_id" : ObjectId("681537738aa101903ed2fe05"), "x" : 1, "y" : 1 }
> db.col.find({},{x:"string"})
{ "_id" : ObjectId("681537738aa101903ed2fe05"), "x" : "string" }

// Planner V1 treats strings as exclude in projection
rs0:PRIMARY> db.col.find()
{ "_id" : ObjectId("68153744d42969f11d5cca72"), "x" : 1, "y" : 1 }
rs0:PRIMARY> db.col.find({},{x:"string"})
{ "_id" : ObjectId("68153744d42969f11d5cca72"), "y" : 1 }
```

- Planner versi 2.0, seperti MongoDB, tidak mengizinkan proyeksi pada bidang yang sama “x” dan “x.a”:

```
// Planner version 2/MongoDB will error out
> db.col.find()
{ "_id" : ObjectId("68153da2012265816bc9ba23"), "x" : [ { "a" : 1 }, 3 ] }
 db.col.find({},{"x.a":1,"x":1}) // error

// Planner Version 1 does not error out
db.col.find()
{ "_id" : ObjectId("68153da2012265816bc9ba23"), "x" : [ { "a" : 1 }, 3 ] }

db.col.find({},{"x.a":1,"x":1})
{ "_id" : ObjectId("68153d60143af947c720d099"), "x" : [ { "a" : 1 }, 3 ] }
```

- Planner versi 2.0, seperti MongoDB, memungkinkan proyeksi pada subdokumen:

```
// 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"
}
```

- Dengan Planner versi 2.0, seperti MongoDB, proyeksi tidak mendukung bidang setelah operator: $

```
// Mongo and Planner Version 2 will error out
  db.col.find()
{ "_id" : ObjectId("68155fa812f843439b593f3f"), "x" : [ { "a" : 100 } ] }
 db.col.find({"x.a":100},{"x.$.a":1}) - // error

// v1 will not error out
 db.col.find()
{ "_id" : ObjectId("68155fa812f843439b593f3f"), "x" : [ { "a" : 100 } ] }
db.col.find({"x.a":100},{"x.$.a":1})
{ "_id" : ObjectId("68155dee13b051d58239cd0a"), "x" : [ { "a" : 100 } ] }
```

- Planner versi 2.0, seperti MongoDB, memungkinkan penggunaan: `$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 }
```

Perencana kueri v3

Planner Versi 3 di Amazon DocumentDB 8.0 mendukung 21 tahap agregasi, termasuk 6 tahap baru. Planner V3 hadir dengan dukungan bawaan untuk perintah yang berbeda. Ini menawarkan peningkatan kinerja hingga 2x secara keseluruhan dibandingkan Planner v2 di Amazon DocumentDB 5.0. Semua fitur dan operator baru di Amazon DocumentDB 8.0 kompatibel dengan Planner v3. Tahapan agregasi baru di Amazon DocumentDB 8.0 yang didukung oleh Planner v3 termasuk $replaceWith, $vectorSearch, $merge, $set, $unset, $bucket. Planner v3 juga mendukung fitur dan operator baru di Amazon DocumentDB 8.0 termasuk Collation, Views, $merge, $pow, $rand, $dateTrunc, $date, $date. ToParts FromParts

Topik

- Prasyarat
- Memilih perencana versi 3.0 sebagai perencana kueri default
- Praktik terbaik
- Batasan
- Perbaikan pada Operator agregat dan berbeda
- Potensi perbedaan perilaku antara perencana versi 1.0, 3.0, dan MongoDB

Prasyarat

Prasyarat berikut berlaku untuk perencana versi 3.0:

- Planner versi 3.0 tersedia di semua wilayah di mana engine versi 8.0 tersedia.
- Perencana versi 3.0 adalah perencana kueri default ketika engine versi 8.0 dipilih.

Memilih perencana versi 3.0 sebagai perencana kueri default

Jika Anda mengubah perencana kueri default di Amazon DocumentDB 8.0, dan perlu kembali ke planner v3, Anda dapat melakukannya dari konsol atau CLI:

- Ikuti langkah-langkah Memodifikasi parameter cluster Amazon DocumentDB untuk memodifikasi grup parameter cluster Anda.
- Untuk parameter berjudul 'PlannerVersion', ubah nilainya menjadi 3.0 yang menunjukkan perencana versi 3.0.
- Pilih Terapkan segera (memilih Terapkan saat reboot akan membuat pilihan tidak efektif hingga reboot cluster berikutnya).

Praktik terbaik

Untuk hasil yang diharapkan, gunakan praktik terbaik berikut saat menerapkan perencana versi 3.0:

- Di cluster global, pilih `plannerVersion` nilai yang sama (1.0 atau 2.0 atau 3.0) di grup parameter cluster untuk kedua wilayah. Perhatikan bahwa memilih versi perencana yang berbeda di wilayah primer dan sekunder dapat menyebabkan perilaku dan kinerja kueri yang tidak konsisten.
- Memperbarui ke perencana versi 3.0 selama jendela pemeliharaan terjadwal atau selama periode lalu lintas yang dikurangi akan menjadi yang paling tidak mengganggu, karena mungkin ada peningkatan tingkat kesalahan jika versi perencana diubah saat beban kerja aktif berjalan.

Batasan

Batasan berikut berlaku untuk perencana versi 3.0:

- Planner versi 3.0 tidak didukung dalam cluster elastis, yang akan kembali ke versi perencana 1.0.

- Sementara Planner v1 memungkinkan penggunaan 'PlanHint' untuk memastikan bahwa rencana kueri tertentu dipilih oleh pengoptimal kueri, Planner v3 tidak mengizinkan penggunaan 'planHint' dan bergantung pada pengoptimalan internal untuk memilih paket terbaik untuk kueri yang diberikan.

Perbaikan pada **`Operator agregat`** dan **`berbeda`**

Planner versi 3.0 memperkenalkan peningkatan di seluruh tahapan $agregat, dan perintah $distinct. Berikut ini adalah beberapa perbaikan yang paling penting..

- Perencana memindahkan tahap $match lebih awal di pipeline bila memungkinkan, mengurangi jumlah dokumen yang diproses oleh tahap berikutnya.

```
//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 } }
])
```

- Perencana secara otomatis menggabungkan tahap $lookup dan $unwind ketika mereka beroperasi di bidang yang sama, mengurangi pemrosesan data perantara dan meningkatkan kinerja.

```
                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
```

- Planner versi 3.0 memperkenalkan strategi eksekusi Distinct Scan baru yang secara signifikan meningkatkan kinerja untuk operasi yang berbeda pada indeks kardinalitas rendah.

```
                  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"
                        }
                }
        }
}
```

Potensi perbedaan perilaku antara perencana versi 1.0, 3.0, dan MongoDB

Dalam beberapa kasus tepi, ada kemungkinan bahwa perencana versi 3.0 dapat menghasilkan hasil yang sedikit berbeda dari perencana versi 1.0. Bagian ini membahas beberapa contoh kemungkinan ini.

Feature Differences

- Pada koleksi kosong atau ketika tahap sebelumnya seperti match menyaring semua dokumen, Planner v1 memberikan output “field” :0. Planner v3 dan MongoDB tidak akan memberikan output apa pun.
- Dengan Plannerv1, {"$skip” :n} tidak melewatkan jika ada kurang dari n dokumen yang dikembalikan oleh tahap sebelumnya. Plannerv3 dan MongoDB melewatkan dengan benar terlepas dari jumlah dokumen yang dikembalikan.
- Ketika koleksi asing yang direferensikan di $lookup tidak ada, plannerv1 memunculkan kesalahan. Planner v3 dan MongoDB memperlakukan koleksi asing sebagai koleksi kosong dan melakukan $lookup.

```
db.coll.aggregate([
    {$lookup: {from: "does_not_exist", localField: "a", foreignField: "a", as:
 "c"}}
])
```

- Hanya Planner v1 yang akan mengizinkan penggunaan beberapa $search dalam pipeline Planner v2/v3 akan menimbulkan kesalahan dan MongoDB tidak mendukungnya.

```
VectorSearch = { "$search": { "vectorSearch": {
   "vector": [0.2, 0.5, 0.8],
   "path": "vectorEmbedding",
   "similarity": "cosine",
   "k": 2,
   "efSearch": 1
   }}}
db.coll.aggregate([VectorSearch, VectorSearch])
```

- Hanya Planner v3 yang berfungsi ketika tahap VectorSearch bukan tahap pertama. Dalam hal ini, MongoDB akan melempar kesalahan dan Planner v1 tidak mendukung tahap $VectorSearch.

```
VectorSearch = { {"$vectorSearch": {
    "queryVector": [0.2, 0.5, 0.8],
    "path": "vectorEmbedding",
    "similarity": "euclidean",
    "limit": 4,
    "numCandidates": 100} }

db.coll.aggregate([$match:{}, VectorSearch])
```

Menanyakan data geospasial dengan Amazon DocumentDB

Bagian ini mencakup bagaimana Anda dapat menanyakan data Geospasial dengan Amazon DocumentDB. Setelah Anda membaca bagian ini, Anda akan dapat menjawab bagaimana menyimpan, menanyakan, dan mengindeks data Geospasial di Amazon DocumentDB.

Topik

- Ikhtisar
- Mengindeks dan menyimpan data geospasial

- Meminta data geospasial
- Batasan

Ikhtisar

Kasus penggunaan umum untuk Geospasial melibatkan analisis kedekatan dari data Anda. Misalnya, “menemukan semua bandara dalam jarak 50 mil dari Seattle”, atau “temukan restoran terdekat dari lokasi tertentu”. Amazon DocumentDB menggunakan spesifikasi GeoJSON untuk mewakili data geospasial. GeoJSON adalah spesifikasi open-source untuk bentuk JSON-formatting dalam ruang koordinat. Koordinat GeoJSON menangkap bujur dan lintang, mewakili posisi pada bola mirip bumi.

Mengindeks dan menyimpan data geospasial

Amazon DocumentDB menggunakan tipe GeoJSON 'Point' untuk menyimpan data geospasial. Setiap dokumen GeoJSON (atau subdokumen) umumnya terdiri dari dua bidang:

- type - bentuk yang diwakili, yang menginformasikan Amazon DocumentDB bagaimana menafsirkan bidang “koordinat”. Saat ini, Amazon DocumentDB hanya mendukung poin
- koordinat — pasangan lintang dan bujur direpresentasikan sebagai objek dalam array — [bujur, lintang]

Amazon DocumentDB juga menggunakan indeks 2dsphere untuk mengindeks data Geospasial. Amazon DocumentDB mendukung poin pengindeksan. Amazon DocumentDB mendukung query proximity dengan pengindeksan 2dsphere.

Mari pertimbangkan skenario di mana Anda sedang membangun aplikasi untuk layanan pengiriman makanan. Anda ingin menyimpan berbagai garis lintang dan pasangan bujur restoran di Amazon DocumentDB. Untuk melakukannya, pertama-tama kami sarankan Anda membuat indeks pada bidang Geospasial yang memegang pasangan lintang dan bujur.

```
use restaurantsdb
db.usarestaurants.createIndex({location:"2dsphere"})
```

Output dari perintah ini akan terlihat seperti ini:

```
{
 "createdCollectionAutomatically" : true,
```

```
 "numIndexesBefore" : 1,
 "numIndexesAfter" : 2,
 "ok" : 1
}
```

Setelah Anda membuat indeks, Anda dapat mulai memasukkan data ke dalam koleksi Amazon DocumentDB Anda.

```
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
      ]
   }
});
```

Meminta data geospasial

Amazon DocumentDB mendukung kueri kedekatan, inklusi, dan persimpangan data Geospasial. Contoh yang baik dari kueri kedekatan adalah menemukan semua titik (semua bandara) yang kurang dari jarak tertentu dan lebih dari jarak dari titik lain (kota). Contoh yang baik dari kueri inklusi adalah menemukan semua titik (semua bandara) yang terletak di area/polygon (negara bagian New York) tertentu. Contoh yang baik dari kueri persimpangan adalah menemukan poligon (status) yang berpotongan dengan titik (kota). Anda dapat menggunakan operator Geospasial berikut untuk mendapatkan wawasan dari data Anda.

- **`$nearSphere`**- `$nearSphere` adalah operator find yang mendukung pencarian titik dari terdekat ke terjauh dari titik GeoJSON.
- **`$geoNear`**- `$geoNear` adalah operator agregasi yang mendukung penghitungan jarak dalam meter dari titik GeoJSON.
- **`$minDistance`**- `$minDistance` adalah operator find yang digunakan bersama dengan `$nearSphere` atau `$geoNear` untuk memfilter dokumen yang setidaknya pada jarak minimum yang ditentukan dari titik pusat.
- **`$maxDistance`**- `$maxDistance` adalah operator find yang digunakan bersama dengan `$nearSphere` atau `$geoNear` untuk memfilter dokumen yang paling banyak pada jarak maksimum yang ditentukan dari titik pusat.
- **`$geoWithin`**- `$geoWithin` adalah operator find yang mendukung pencarian dokumen dengan data geospasial yang ada seluruhnya dalam bentuk tertentu seperti poligon.
- **`$geoIntersects`**- `$geoIntersects` adalah operator find yang mendukung pencarian dokumen yang data geospasialnya berpotongan dengan objek GeoJSON tertentu.

> **Note**
>
> `$geoNear`dan `$nearSphere` memerlukan indeks 2dsphere pada bidang GeoJSON yang Anda gunakan dalam kueri kedekatan Anda.

Contoh 1

Dalam contoh ini, Anda akan belajar bagaimana menemukan semua restoran (poin) yang diurutkan berdasarkan jarak terdekat dari alamat (titik).

Untuk melakukan kueri seperti itu, Anda dapat menggunakan `$geoNear` untuk menghitung jarak set titik dari titik lain. Anda juga dapat menambahkan `distanceMultiplier` untuk mengukur jarak dalam kilometer.

```
db.usarestaurants.aggregate([
   {
      "$geoNear":{
         "near":{
            "type":"Point",
            "coordinates":[
               -122.3516,
               47.6156
            ]
         },
         "spherical":true,
         "distanceField":"DistanceKilometers",
         "distanceMultiplier":0.001
      }
   }
])
```

Perintah di atas akan mengembalikan restoran yang diurutkan berdasarkan jarak (paling dekat dengan terjauh) dari titik yang ditentukan. Output dari perintah ini akan terlihat seperti ini

```
{ "_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**
>
> numOpsi `limit` dan untuk `$geoNear` tahap, serta batas default 100 dokumen, tidak digunakan lagi di Amazon DocumentDB 8.0 (Engine patch Version 4.0.5748) dan yang lebih baru. Contoh berikut menggunakan `limit` dan `num` berlaku hanya untuk Amazon DocumentDB 8.0 (Engine patch Version 4.0.4448) dan versi sebelumnya.

Untuk membatasi jumlah hasil dalam kueri, gunakan `num` opsi `limit` or.

`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
      }
   }
])
```

Contoh 2

Dalam contoh ini, Anda akan belajar cara menemukan semua restoran (poin) dalam jarak 2 kilometer dari alamat (titik) tertentu. Untuk melakukan kueri seperti itu, Anda dapat menggunakan `$nearSphere` dalam minimum `$minDistance` dan maksimum `$maxDistance` dari GeoJSON Point

```
db.usarestaurants.find({
   "location":{
      "$nearSphere":{
         "$geometry":{
            "type":"Point",
            "coordinates":[
               -122.3516,
               47.6156
            ]
         },
         "$minDistance":1,
         "$maxDistance":2000
      }
   }
},
{
   "name":1
})
```

Perintah di atas akan mengembalikan restoran pada jarak maksimum 2 kilometer dari titik yang ditentukan. Output dari perintah ini akan terlihat seperti ini

```
{ "_id" : ObjectId("611f3da985009a81ad38e74b"), "name" : "Noodle House" }
```

Batasan

Amazon DocumentDB tidak mendukung kueri atau pengindeksan LineString Poligon,,,,, dan. MultiPoint MultiPolygon MultiLineString GeometryCollection

Indeks sebagian

Indeks sebagian mengindeks dokumen dalam koleksi yang memenuhi kriteria filter tertentu. Fitur indeks sebagian didukung di kluster berbasis instans Amazon DocumentDB 5.0.

Topik

- Buat indeks sebagian
- Operator yang didukung
- Kueri menggunakan indeks paral
- Fungsionalitas indeks sebagian
- Keterbatasan indeks sebagian

Buat indeks sebagian

Untuk membuat indeks sebagian, gunakan `createIndex()` metode dengan `partialFilterExpression` opsi. Misalnya, operasi berikut membuat indeks gabungan unik dalam koleksi pesanan yang mengindeks dokumen yang memiliki `OrderID` dan memiliki `isDelivered` bidang sebagai true:

```
db.orders.createIndex(
  {"category": 1, "CustomerId": 1, "OrderId": 1},
  {"unique": true, "partialFilterExpression":
    {"$and": [
      {"OrderId": {"$exists": true}},
      {"isDelivered": {"$eq": false}}
    ]}
  }
)
```

Operator yang didukung

- $ eq
- $ ada
- $ dan (hanya di tingkat atas)
- $gt/$gte/$lt/$lte (pemindaian indeks hanya digunakan ketika filter, diprediksikan dalam kueri, cocok dengan ekspresi filter sebagian persis) (Lihat Batasan)

Kueri menggunakan indeks paral

Pola kueri berikut dimungkinkan menggunakan indeks sebagian:

- Predikat kueri sama persis dengan ekspresi filter indeks sebagian:

```
db.orders.find({"$and": [
    {"OrderId": {"$exists": true}},
    {"isDelivered": {"$eq": false}}
  ]}).explain()
```

- Hasil yang diharapkan filter kueri adalah subset logis dari filter paral:

```
db.orders.find({"$and": [
    {"OrderId": {"$exists": true}},
    {"isDelivered": {"$eq": false}},
    {"OrderAmount": {"$eq": "5"}}
  ]}).explain()
```

- Sub-predikat kueri dapat digunakan bersama dengan indeks lain:

```
db.orders.createIndex({"anotherIndex":1})
db.orders.find({ "$or": [
      {"$and": [
        {"OrderId": {"$exists": true}},
        {"isDelivered": {"$eq": false}}
      ]},
      {"anotherIndex": {"$eq": 5}}
    ]
  }).explain()
```

> **Note**
>
> Perencana kueri dapat memilih untuk menggunakan pemindaian koleksi daripada pemindaian indeks jika efisien untuk melakukannya. Ini biasanya terlihat untuk koleksi atau kueri yang sangat kecil yang akan mengembalikan sebagian besar koleksi.

Fungsionalitas indeks sebagian

Daftar indeks sebagian

Buat daftar indeks sebagian dengan sebagian FilterExpression menggunakan `getIndex` operasi. Misalnya, `getIndex` operasi yang dikeluarkan dalam mencantumkan indeks sebagian dengan bidang kunci, nama, dan partialFilterExpressions:

```
db.orders.getIndexes()
```

Contoh ini mengembalikan output sebagai berikut:

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

Beberapa ekspresi filter paral pada kunci yang sama: urutan

Indeks paral yang berbeda dapat dibuat untuk kombinasi bidang yang sama (key:order). Indeks ini harus memiliki nama yang berbeda.

```
db.orders.createIndex(
  {"OrderId":1},
  {
    name:"firstPartialIndex",
    partialFilterExpression:{"OrderId":{"$exists": true}}
  }
)
```

```
db.orders.createIndex(
  {"OrderId":1},
  {
    name:"secondPartialIndex",
    partialFilterExpression:{"OrderId":{"$gt": 1000}}
  }
)
```

Jalankan `getIndexes` operasi untuk mencantumkan semua indeks dalam koleksi:

```
db.orders.getIndexes()
```

Contoh-contoh ini mengembalikan output berikut:

```
[
  {
    "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**
>
> Nama indeks harus berbeda dan harus dihapus dengan nama saja.

Indeks dengan properti parsi dan TTL

Anda juga dapat membuat indeks yang memiliki properti parsi dan TTL dengan menentukan keduanya `partialFilterExpression` dan `expireAfterSeconds` opsi selama pembuatan indeks. Ini memungkinkan Anda untuk memberikan kontrol lebih besar atas dokumen mana yang sekarang dihapus dari koleksi.

Misalnya, Anda mungkin memiliki indeks TTL yang mengidentifikasi dokumen yang akan dihapus setelah jangka waktu tertentu. Anda sekarang dapat memberikan ketentuan tambahan tentang kapan harus menghapus dokumen menggunakan opsi indeks sebagian:

```
db.orders.createIndex(
    { "OrderTimestamp": 1 },
    {
        expireAfterSeconds: 3600 ,
        partialFilterExpression: { "isDelivered": { $eq: true } }
```

```
    }
)
```

Contoh ini mengembalikan output sebagai berikut:

```
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1,
        "operationTime" : Timestamp(1234567890, 1)
}
```

Jalankan `getIndexes` operasi untuk mencantumkan indeks yang ada dalam koleksi:

```
db.orders.getIndexes()
[
    {
        "v" : 4,
        "key" : {
            "_id" : 1
        },
        "name" : "_id_",
        "ns" : "test.orders"
    }
```

Contoh ini mengembalikan output sebagai berikut:

```
[
    {
        "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
    }
]
```

Keterbatasan indeks sebagian

Batasan berikut berlaku untuk fitur indeks sebagian:

- Kueri ketidaksetaraan di Amazon DocumentDB hanya akan menggunakan indeks sebagian ketika predikat filter kueri sama persis dengan dan memiliki tipe data `partialFilterExpression` yang sama.

  > **Note**
  >
  > Bahkan `$hint` tidak dapat digunakan untuk memaksa IXSCAN untuk kasus di atas.

  Dalam contoh berikut, hanya `partialFilterExpression` diterapkan pada `field1` tetapi tidak`field2`:

  ```
  db.orders.createIndex(
    {"OrderAmount": 1},
    {"partialFilterExpression": { OrderAmount : {"$gt" : 5}}}
  )

  db.orders.find({OrderAmount : {"$gt" : 5}}) // Will use partial index
  db.orders.find({OrderAmount : {"$gt" : 6}}) // Will not use partial index
  db.orders.find({OrderAmount : {"$gt" : Decimal128(5.00)}}) // Will not use partial
   index
  ```

- A `partialFilterExpression` dengan operator array tidak didukung. Operasi berikut akan menghasilkan kesalahan:

  ```
  db.orders.createIndex(
  ```

```
  {"CustomerId":1},
  {'partialFilterExpression': {'OrderId': {'$eq': [1000, 1001, 1002]}}}
)
```

- Operator berikut tidak didukung di FilterExpression bidang paral:
  - `$all`(operator array)
  - `$mod`(operator array)
  - `$or`
  - `$xor`
  - `$not`
  - `$nor`
- Tipe data dari ekspresi filter dan filter harus sama.

Melakukan pencarian teks dengan Amazon DocumentDB

Fitur pencarian teks lengkap asli Amazon DocumentDB (Indeks Teks v1) memungkinkan Anda melakukan pencarian teks pada kumpulan data tekstual besar menggunakan indeks teks tujuan khusus. Bagian ini menjelaskan fungsionalitas fitur indeks teks dan memberikan langkah-langkah tentang cara membuat dan menggunakan indeks teks di Amazon DocumentDB. Batasan pencarian teks juga terdaftar.

Topik

- Fungsionalitas yang didukung
- Menggunakan indeks teks Amazon DocumentDB
- Perbedaan dengan MongoDB
- Praktik dan pedoman terbaik
- Indeks Teks V2
- Batasan

Fungsionalitas yang didukung

Pencarian teks Amazon DocumentDB mendukung fungsionalitas yang kompatibel dengan MongoDB API berikut:

- Buat indeks teks pada satu bidang.

- Buat indeks teks majemuk yang menyertakan lebih dari satu bidang teks.
- Lakukan pencarian kata tunggal atau multi-kata.
- Kontrol hasil pencarian menggunakan bobot.
- Urutkan hasil pencarian berdasarkan skor.
- Gunakan indeks teks dalam pipa agregasi.
- Cari frase yang tepat.

Menggunakan indeks teks Amazon DocumentDB

Untuk membuat indeks teks pada bidang yang berisi data string, tentukan string “teks” seperti yang ditunjukkan di bawah ini:

Indeks bidang tunggal:

```
db.test.createIndex({"comments": "text"})
```

Indeks ini mendukung kueri pencarian teks di bidang string “komentar” dalam koleksi yang ditentukan.

Buat indeks teks majemuk pada lebih dari satu bidang string:

```
db.test.createIndex({"comments": "text", "title":"text"})
```

Indeks ini mendukung kueri pencarian teks di bidang string “komentar” dan “judul” dalam koleksi yang ditentukan. Anda dapat menentukan hingga 30 bidang saat membuat indeks teks majemuk. Setelah dibuat, kueri pencarian teks Anda akan menanyakan semua bidang yang diindeks.

> **Note**
>
> Hanya satu indeks teks yang diizinkan pada setiap koleksi.

Daftar indeks teks pada koleksi Amazon DocumentDB

Anda dapat menggunakan `getIndexes()` pada koleksi Anda untuk mengidentifikasi dan mendeskripsikan indeks, termasuk indeks teks, seperti yang ditunjukkan pada contoh berikut:

```
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
   }
]
```

Setelah Anda membuat indeks, mulailah memasukkan data ke dalam koleksi Amazon DocumentDB Anda.

```
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"}])
```

Menjalankan kueri pencarian teks

Jalankan kueri pencarian teks satu kata

Anda perlu menggunakan `$text` dan `$search` operator untuk melakukan pencarian teks. Contoh berikut mengembalikan semua dokumen di mana bidang teks diindeks berisi string “apel” atau “apel” dalam format lain seperti “apel”:

```
db.test.find({$text: {$search: "apple"}})
```

Output:

Output dari perintah ini terlihat seperti ini:

```
{ "_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" }
```

Jalankan pencarian teks multi-kata

Anda juga dapat melakukan pencarian teks multi-kata pada data Amazon DocumentDB Anda. Perintah di bawah ini mengembalikan dokumen dengan bidang teks yang diindeks yang berisi “apple” atau “pie”:

```
db.test.find({$text: {$search: "apple pie"}})
```

Output:

Output dari perintah ini terlihat seperti ini:

```
{ "_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" }
```

Jalankan pencarian teks frase multi-kata

Untuk pencarian frase multi-kata, gunakan contoh ini:

```
db.test.find({$text: {$search: "\"apple pie\""}})
```

Output:

Perintah di atas mengembalikan dokumen dengan bidang teks diindeks yang berisi frase yang tepat “apple pie”. Output dari perintah ini terlihat seperti ini:

```
{ "_id" : 4, "star_rating" : 2, "comments" : "bake the apple pie in the oven" }
```

Jalankan pencarian teks dengan filter

Anda juga dapat menggabungkan pencarian teks dengan operator kueri lain untuk memfilter hasil berdasarkan kriteria tambahan:

```
db.test.find({$and: [{star_rating: 5}, {$text: {$search: "interest"}}]})
```

Output:

Perintah di atas mengembalikan dokumen dengan bidang teks yang diindeks yang berisi segala bentuk “minat” dan “star_rating” sama dengan 5. Output dari perintah ini terlihat seperti ini:

```
{ "_id" : 5, "star_rating" : 5, "comments" : "interesting couch" }
{ "_id" : 6, "star_rating" : 5, "comments" : "interested in couch for sale, year
 2022" }
```

Batasi jumlah dokumen yang dikembalikan dalam pencarian teks

Anda dapat memilih untuk membatasi jumlah dokumen yang dikembalikan dengan menggunakan`limit`:

```
db.test.find({$and: [{star_rating: 5}, {$text: {$search: "couch"}}]}).limit(1)
```

Output:

Perintah di atas mengembalikan satu hasil yang memenuhi filter:

```
{ "_id" : 5, "star_rating" : 5, "comments" : "interesting couch" }
```

Urutkan hasil berdasarkan skor teks

Contoh berikut mengurutkan hasil pencarian teks berdasarkan skor teks:

```
db.test.find({$text: {$search: "apple"}}, {score: {$meta: "textScore"}}).sort({score:
 {$meta: "textScore"}})
```

Output:

Perintah di atas mengembalikan dokumen dengan bidang yang diindeks teks yang berisi “apel”, atau “apel” dalam format lain seperti “apel”, dan mengurutkan hasilnya berdasarkan seberapa relevan dokumen tersebut terkait dengan istilah pencarian. Output dari perintah ini terlihat seperti ini:

```
{ "_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 }
```

`$text`dan `$search` juga didukung untuk`aggregate`,`count`,`findAndModify`,`update`, dan `delete` perintah.

Operator agregasi

Pipa agregasi menggunakan **`$match`**

```
db.test.aggregate(
   [{ $match: { $text: { $search: "apple pie" } } }]
)
```

Output:

Perintah di atas mengembalikan hasil sebagai berikut:

```
{ "_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" }
```

Kombinasi dari operator agregasi lainnya

```
db.test.aggregate(
   [
      { $match: { $text: { $search: "apple pie" } } },
      { $sort: { score: { $meta: "textScore" } } },
      { $project: { score: { $meta: "textScore" } } }
   ]
```

```
)
```

Output:

Perintah di atas mengembalikan hasil sebagai berikut:

```
{ "_id" : 4, "score" : 0.6079270860936958 }
{ "_id" : 1, "score" : 0.3039635430468479 }
{ "_id" : 2, "score" : 0.3039635430468479 }
{ "_id" : 3, "score" : 0.3039635430468479 }
```

Tentukan beberapa bidang saat membuat indeks teks

Anda dapat menetapkan bobot hingga tiga bidang dalam indeks teks majemuk Anda. Bobot default yang ditetapkan ke bidang dalam indeks teks adalah satu (1). Berat adalah parameter opsional dan harus dalam kisaran 1 hingga 100000.

```
db.test.createIndex(
   {
     "firstname": "text",
     "lastname": "text",
     ...
   },
   {
     weights: {
       "firstname": 5,
       "lastname":10,
       ...
     },
     name: "name_text_index"
   }
 )
```

Perbedaan dengan MongoDB

Fitur indeks teks Amazon DocumentDB menggunakan indeks terbalik dengan algoritma frekuensi istilah. Indeks teks jarang secara default. Karena perbedaan dalam logika parsing, pembatas tokenisasi, dan lainnya, set hasil yang sama seperti MongoDB mungkin tidak dikembalikan untuk kumpulan data atau bentuk kueri yang sama.

Perbedaan tambahan berikut antara indeks teks Amazon DocumentDB dan MongoDB ada:

- Indeks majemuk menggunakan indeks non-teks tidak didukung.
- Indeks teks Amazon DocumentDB tidak peka huruf besar/kecil.
- Hanya bahasa Inggris yang didukung dengan indeks teks.
- Pengindeksan teks bidang array (atau multi-kunci) tidak didukung. Misalnya, membuat indeks teks pada “a “dengan dokumen {“a”: [“apple”, “pie”]} akan gagal.
- Pengindeksan teks wildcard tidak didukung.
- Indeks teks unik tidak didukung.
- Tidak termasuk istilah tidak didukung.

Praktik dan pedoman terbaik

- Untuk kinerja optimal pada kueri penelusuran teks yang melibatkan pengurutan berdasarkan skor teks, sebaiknya Anda membuat indeks teks sebelum memuat data.
- Indeks teks memerlukan penyimpanan tambahan untuk salinan internal yang dioptimalkan dari data yang diindeks. Ini memiliki implikasi biaya tambahan.

Indeks Teks V2

Amazon DocumentDB 8.0 memperkenalkan versi baru indeks teks (V2) yang mengubah parser pencarian teks yang mendasarinya untuk menghadirkan lebih banyak kompatibilitas dengan MongoDB.

Selain fungsionalitas yang disediakan oleh indeks teks V1, indeks teks V2 juga memberikan dukungan berikut:

- Perencana memindahkan tahap $match lebih awal di pipeline bila memungkinkan, mengurangi jumlah dokumen yang diproses oleh tahap berikutnya.
- Peningkatan tokenisasi karakter khusus di bidang teks, seperti email, URL, dan jalur file. V2 dapat mengurai dan mencocokkan token individual dalam string ini, sedangkan V1 tidak bisa.

```
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" }
```

Batasan

Pencarian teks memiliki batasan berikut di Amazon DocumentDB:

- Indeks teks menyimpan leksem dan informasi posisinya. Ukuran gabungan semua leksem dan informasi posisinya, dalam satu dokumen, dibatasi hingga 1MB.

Indeks

Topik

- Manajemen Indeks
- Jenis Indeks
- Properti Indeks

Manajemen Indeks

Bagian ini mencakup topik manajemen indeks untuk Amazon DocumentDB.

Topik

- Pembuatan indeks
- Mempertahankan indeks
- Praktik terbaik untuk bekerja dengan indeks

Pembuatan indeks

Membangun indeks di Amazon DocumentDB memerlukan sejumlah keputusan yang harus dibuat:

- Seberapa cepat itu harus diselesaikan?
- Bisakah koleksi tidak dapat diakses saat build sedang berlangsung?
- Berapa banyak daya komputasi instans yang dapat dialokasikan ke build?
- Jenis indeks apa yang harus dibuat?

Bagian ini membantu Anda menjawab pertanyaan-pertanyaan ini dan memberikan perintah dan contoh pemantauan untuk membuat indeks Amazon DocumentDB pada koleksi cluster berbasis instans Anda.

Pedoman

Pedoman berikut mencakup batasan dasar dan pengorbanan konfigurasi saat membuat indeks baru:

- Dukungan versi Amazon DocumentDB - Meskipun pengindeksan pekerja tunggal didukung di semua versi Amazon DocumentDB, beberapa pengindeksan pekerja hanya didukung di Amazon DocumentDB versi 4.0 dan 5.0.
- Trade-off kinerja - Meningkatkan jumlah pekerja dalam proses pembuatan indeks meningkatkan pemanfaatan CPU dan membaca IO pada instance utama database Amazon DocumentDB Anda. Sumber daya yang dibutuhkan untuk membuat indeks baru tidak akan tersedia untuk beban kerja Anda yang sedang berjalan.
- Cluster elastis - Pengindeksan paralel tidak didukung pada kluster elastis Amazon DocumentDB.
- Pekerja maksimum - Jumlah maksimum pekerja yang dapat Anda konfigurasi tergantung pada ukuran instance utama Anda di cluster database Anda. Ini adalah setengah dari jumlah total vCPU pada instance utama cluster database Anda. Misalnya, Anda dapat menjalankan maksimal 32 pekerja pada instance db.r6g.16xlarge yang memiliki 64 vCPU.

> **Note**
>
> Build indeks paralel didukung pada instance 2xlarge ke atas.

- Pekerja minimum - Jumlah minimum pekerja yang dapat Anda konfigurasi adalah satu. Pengaturan default untuk pembuatan indeks pada cluster berbasis instance adalah dua pekerja. Namun, Anda dapat mengurangi jumlah pekerja menjadi satu dengan menggunakan opsi “utas pekerja”. Ini akan menjalankan proses dengan satu pekerja.
- Kompresi indeks - Amazon DocumentDB tidak mendukung kompresi indeks. Ukuran data untuk indeks mungkin lebih besar daripada saat Anda menggunakan opsi lain.
- Mengindeks beberapa koleksi - Setengah dari vCPU pada instance utama cluster database Anda dapat digunakan untuk pekerja yang dikonfigurasi yang melakukan pembuatan indeks pada beberapa koleksi.
- Jenis indeks - Lihat Jenis Indeks daftar jenis indeks yang didukung di Amazon DocumentDB.

Jenis build indeks

Empat jenis build indeks adalah:

- Foreground: Pembuatan indeks latar depan memblokir semua operasi database lain pada koleksi tempat indeks dibuat, hingga pembuatan indeks selesai. Build latar depan Amazon DocumentDB terdiri dari lima tahap.
- Foreground (unique): Build indeks latar depan dokumen tunggal (unik) memblokir operasi database lain seperti build latar depan biasa. Berbeda dengan build latar depan dasar, build unik menggunakan tahap tambahan (kunci pengurutan 2) untuk mencari kunci duplikat. Build latar depan (unik) terdiri dari enam tahap.
- Latar Belakang: Build indeks latar belakang memungkinkan operasi database lainnya berjalan di latar depan saat indeks sedang dibuat. Build latar belakang Amazon DocumentDB terdiri dari delapan tahap.
- Latar belakang (unik): Build indeks latar belakang dokumen tunggal (unik) memungkinkan operasi database lain berjalan di latar depan saat indeks sedang dibuat. Tidak seperti build latar belakang dasar, build unik menggunakan tahap tambahan (kunci pengurutan 2) untuk mencari kunci duplikat. Build latar belakang (unik) terdiri dari sembilan tahap.

Jenis build indeks default adalah latar depan.

Untuk membangun indeks di latar belakang, Anda menambahkan parameter “background”: true saat membuat indeks.

Memulai

Untuk memulai pembuatan indeks pada koleksi, gunakan perintah createIndexes. Secara default, perintah akan menjalankan dua pekerja paralel yang meningkatkan kecepatan proses pembuatan indeks sebanyak dua kali.

Misalnya, proses perintah berikut menunjukkan cara membuat indeks untuk bidang “user_name” dalam dokumen dan meningkatkan kecepatan proses pengindeksan menjadi empat pekerja:

1. Buat indeks di latar belakang menggunakan dua pekerja paralel di cluster:

```
db.runCommand({"createIndexes":"test","indexes":[{"key": {"user_name":1},
 "name":"username_idx", "background":true }]})
```

2. Untuk mengoptimalkan kecepatan proses pembuatan indeks, Anda dapat menentukan jumlah pekerja dengan menggunakan opsi “thread pekerja” (“pekerja”:<number>) dalam perintah db.runCommand createIndexes.

Tingkatkan kecepatan proses menjadi empat pekerja paralel:

```
db.runCommand({"createIndexes":"test","indexes":[{"key": {"user_name":1},
 "name":"username_idx", "workers":4}]})
```

> **Note**
>
> Semakin tinggi jumlah pekerja, semakin cepat pembuatan indeks berlangsung. Namun, semakin tinggi jumlah pekerja meningkat, semakin tinggi beban meningkat pada vCPU dan membaca IO dari instance utama Anda. Pastikan klaster Anda cukup disediakan untuk menangani beban yang meningkat tanpa menurunkan beban kerja lainnya.

Status kemajuan pengindeksan

Proses pembuatan indeks bekerja dengan menginisialisasi, memindai koleksi, menyortir kunci, dan, akhirnya, memasukkan kunci melalui pembuat indeks. Prosesnya memiliki hingga enam tahap saat Anda menjalankannya di latar depan, dan hingga sembilan tahap saat Anda menjalankannya di latar belakang. Anda dapat melihat metrik status seperti penyelesaian persentase, jumlah total blok penyimpanan yang dipindai, kunci yang diurutkan, dan kunci yang disisipkan berdasarkan tahap demi tahap.

Pantau kemajuan pada proses pengindeksan dengan menggunakan perintah db.currentTop () di shell mongo. Penyelesaian 100% dari tahap terakhir menunjukkan bahwa semua indeks telah berhasil dibuat:

```
db.currentOp({"command.createIndexes": { $exists : true } })
```

> **Note**
>
> Melihat status kemajuan pengindeksan hanya didukung di Amazon DocumentDB 5.0.

Tahapan pembuatan indeks

Stage	Latar depan	Latar depan (unik)	Latar Belakang	Latar Belakang (unik)
Inisialisasi	1	1	1	1
indeks bangunan: menginisialisasi	2	2	2	2
indeks bangunan: koleksi pemindaian	3	3	3	3
indeks bangunan: kunci penyortiran 1	4	4	4	4
indeks bangunan: kunci penyortiran 2		5		5
indeks bangunan: memasukkan kunci	5	6	5	6
memvalidasi: indeks pemindaian			6	7
memvalidasi: menyortir tupel			7	8
memvalidasi: koleksi pemindaian			8	9

- inisialisasi - createIndex sedang mempersiapkan pembuat indeks. Fase ini harus sangat singkat.
- indeks bangunan: menginisialisasi - Pembangun indeks sedang mempersiapkan untuk membuat indeks. Fase ini harus sangat singkat.
- indeks bangunan: koleksi pemindaian - Pembuat indeks melakukan pemindaian koleksi untuk mengumpulkan kunci indeks. Satuan ukuran adalah “blok”.

> **Note**
>
> Jika lebih dari satu pekerja dikonfigurasi untuk build indeks, itu akan ditampilkan di tahap ini. Tahap “pengumpulan pemindaian” adalah satu-satunya tahap yang menggunakan banyak pekerja selama proses pembuatan indeks. Semua tahapan lainnya akan menampilkan satu pekerja.

- indeks bangunan: kunci penyortiran 1 - Pembuat indeks menyortir kunci indeks yang dikumpulkan. Satuan ukuran adalah “kunci”.
- indeks bangunan: kunci penyortiran 2 - Pembuat indeks menyortir kunci indeks yang dikumpulkan yang sesuai dengan tupel mati. Fase ini hanya ada untuk bangunan indeks yang unik. Satuan ukuran adalah “kunci”.
- indeks bangunan: memasukkan kunci - Pembuat indeks memasukkan kunci indeks ke dalam indeks baru. Satuan ukuran adalah “kunci”.
- memvalidasi: indeks pemindaian - createIndex memindai indeks untuk menemukan kunci yang perlu divalidasi. Satuan ukuran adalah “blok”.
- memvalidasi: menyortir tupel - createIndex menyortir output dari fase pemindaian indeks.
- memvalidasi: koleksi pemindaian - createIndex memindai koleksi untuk memvalidasi kunci indeks yang ditemukan dalam dua fase sebelumnya. Satuan ukuran adalah “blok”.

Contoh keluaran pembuatan indeks

Dalam contoh keluaran berikut (membangun indeks latar depan), status pembuatan indeks ditampilkan. Bidang “msg” merangkum kemajuan build dengan menunjukkan tahapan dan persentase penyelesaian build. Bidang “pekerja” menunjukkan jumlah pekerja yang digunakan selama tahap pembuatan indeks tersebut. Bidang “kemajuan” menunjukkan angka aktual yang digunakan untuk menghitung persentase penyelesaian.

> **Note**
>
> Bidang IndexBuildName “saat ini”, “msg”, dan “kemajuan” tidak didukung di Amazon DocumentDB versi 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
}
```

Sumber daya pemecahan masalah

Lihat sumber daya berikut tentang pemecahan masalah pembuatan indeks:

- Pembuatan indeks gagal
- Indeks latar belakang membangun masalah latensi dan gagal

Mempertahankan indeks

Bloat indeks

Amazon DocumentDB Multi-Version menggunakan Concurrency Control (MVCC) untuk mengelola transaksi bersamaan. Ketika dokumen dihapus atau diperbarui, versi sebelumnya tetap dalam koleksi dan indeks sebagai versi “mati”. Proses pengumpulan sampah secara otomatis merebut kembali ruang dari versi mati ini untuk operasi masa depan.

Indeks kembung terjadi ketika indeks koleksi menjadi lebih besar karena akumulasi entri indeks mati atau usang atau fragmentasi dalam halaman. Persentase yang dilaporkan mewakili jumlah ruang indeks yang dapat digunakan oleh entri indeks future. Bloat ini menghabiskan ruang di cache buffer dan penyimpanan. Jika Anda ingin menghapus kembung, Anda perlu membangun kembali indeks.

Contoh: Jalankan perintah berikut untuk menentukan penyimpanan yang tidak digunakan untuk indeks Anda:

```
db.coll.aggregate({$indexStats:{}});
```

Hasilnya terlihat mirip dengan ini:

```
{
    "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
    }
}
```

Anda dapat membangun kembali indeks tanpa downtime menggunakan perintah reIndex, yang memerlukan pemindaian seluruh koleksi. Lihat Pemeliharaan indeks menggunakan ReIndex.

Pemeliharaan indeks menggunakan reIndex

ReIndex adalah perintah yang digunakan untuk membangun kembali indeks. Hal ini biasanya digunakan ketika indeks telah menjadi rusak atau tidak efisien. Seiring waktu, indeks dapat mengakumulasi ruang yang tidak terpakai karena banyak pembaruan, penyisipan, atau penghapusan, yang menyebabkan kinerja menurun. Pengindeksan ulang membantu menghilangkan ruang yang tidak terpakai dan mengembalikan efisiensi indeks.

Pedoman ReIndex

- ReIndex hanya didukung di Amazon DocumentDB 5.0 dan di atasnya
- ReIndex selalu berjalan di latar belakang.
- Amazon DocumentDB mendukung reindex indeks tunggal di latar belakang, memungkinkan untuk beberapa pekerja. Indeks lama dapat digunakan oleh kueri saat proses reIndex berjalan.
- Amazon DocumentDB mendukung laporan kemajuan pengindeksan melalui CurrenTop. Anda dapat melihat tahapan pembuatan indeks yang mirip dengan tahapan pembuatan Indeks yang dilihat selama pembuatan indeks. Satu-satunya perbedaan adalah bahwa ReIndex selalu memiliki delapan tahap, terlepas dari apakah itu unik atau tidak. Tidak ada tahap “indeks bangunan: kunci pengurutan 2".
- ReIndex dapat berjalan bersamaan dengan perintah apa pun kecuali perintah terkait indeks pada koleksi yang sama: createIndexes, dropIndex, collMod, dan RenameCollection.
- ReIndex saat ini tidak didukung untuk indeks teks, geospasial, vektor, dan sebagian.

Membangun reIndex

Gunakan perintah berikut untuk membangun kembali indeks Anda:

```
db.runCommand({ reIndex: "collection-name", index: "index-name"})
```

Secara opsional, Anda juga dapat mengontrol jumlah pekerja yang ditugaskan untuk proses pembangunan kembali:

```
db.runCommand({ reIndex: "collection-name", index: "index-name", workers: number })
```

Untuk informasi khusus untuk mengelola indeks dengan Java, lihat Manajemen indeks di Amazon DocumentDB dengan Java.

Praktik terbaik untuk bekerja dengan indeks

Indeks bangunan

Ketika mengimpor data ke Amazon DocumentDB, Anda harus membuat indeks Anda sebelum mengimpor set data besar. Anda dapat menggunakan Alat Indeks Amazon DocumentDB untuk mengekstrak indeks dari instance MongoDB atau direktori mongodump yang sedang berjalan, dan membuat indeks tersebut di cluster Amazon DocumentDB. Untuk panduan lebih lanjut tentang migrasi, lihat Migrasi ke DocumentDB.

Selektivitas indeks

Kami merekomendasikan Anda membatasi pembuatan indeks ke bidang di mana jumlah nilai duplikat kurang dari 1% dari jumlah total dokumen dalam koleksi. Sebagai contoh, jika koleksi Anda berisi 100.000 dokumen, hanya buat indeks pada bidang di mana nilai yang sama terjadi 1.000 kali atau kurang.

Memilih indeks dengan jumlah nilai unik yang tinggi (yaitu, kardinalitas tinggi) memastikan bahwa operasi filter mengembalikan sejumlah kecil dokumen, sehingga menghasilkan kinerja yang baik selama pemindaian indeks. Contoh indeks kardinalitas tinggi adalah indeks unik, yang menjamin bahwa predikat kesetaraan mengembalikan paling banyak satu dokumen. Contoh kardinalitas rendah termasuk indeks di atas bidang Boolean dan indeks di atas hari dalam seminggu. Karena kinerjanya yang buruk, indeks kardinalitas rendah tidak mungkin dipilih oleh pengoptimal kueri database. Pada

saat yang sama, indeks kardinalitas rendah terus mengkonsumsi sumber daya seperti ruang disk dan. I/Os Sebagai aturan praktis, Anda harus menargetkan indeks pada bidang di mana frekuensi nilai khas adalah 1% dari total ukuran koleksi atau kurang.

Selain itu, direkomendasikan untuk hanya membuat indeks pada bidang yang umumnya digunakan sebagai filter dan secara teratur mencari indeks yang tidak terpakai. Untuk informasi selengkapnya, lihat the section called “Menganalisis penggunaan indeks dan mengidentifikasi indeks yang tidak digunakan”.

Dampak indeks pada penulisan data

Meskipun indeks dapat meningkatkan performa kueri dengan menghindari kebutuhan untuk memindai setiap dokumen dalam koleksi, peningkatan ini disertai dengan tradeoff. Untuk setiap indeks pada koleksi, setiap kali dokumen dimasukkan, diperbarui, atau dihapus, basis data harus memperbarui koleksi dan menulis bidang untuk masing-masing indeks untuk koleksi. Sebagai contoh, jika koleksi memiliki sembilan indeks, basis data harus melakukan sepuluh penulisan sebelum mengakui operasi ke klien. Dengan demikian, setiap indeks tambahan menimbulkan latensi tulis tambahan, I/Os, dan peningkatan penyimpanan yang digunakan secara keseluruhan.

Instans klaster harus berukuran tepat untuk menjaga semua memori set kerja. Ini menghindari kebutuhan untuk terus membaca halaman indeks dari volume penyimpanan, yang berdampak negatif pada kinerja dan menghasilkan I/O biaya yang lebih tinggi. Untuk informasi selengkapnya, lihat Ukuran instans.

Untuk performa terbaik, meminimalkan jumlah indeks dalam koleksi Anda, tambahkan hanya indeks yang diperlukan untuk meningkatkan performa untuk kueri umum. Meskipun beban kerja bervariasi, pedoman yang baik adalah untuk menjaga jumlah indeks per koleksi menjadi lima atau lebih sedikit.

Mengidentifikasi indeks yang hilang

Mengidentifikasi indeks yang hilang adalah praktik terbaik yang kami rekomendasikan untuk dilakukan secara teratur. Untuk informasi selengkapnya, lihat the section called “Identifikasi indeks yang terlewat”.

Mengidentifikasi indeks yang tidak digunakan

Mengidentifikasi dan menghapus indeks yang tidak terpakai adalah praktik terbaik yang kami rekomendasikan untuk dilakukan secara teratur. Untuk informasi selengkapnya, lihat the section called “Menganalisis penggunaan indeks dan mengidentifikasi indeks yang tidak digunakan”.

Jenis Indeks

Topik

- Indeks Bidang Tunggal
- Indeks Senyawa
- Multi-key Indeks
- Indeks Teks
- Indeks Geospasial
- Indeks vektor

Indeks Bidang Tunggal

Indeks bidang tunggal menyimpan informasi dari bidang tunggal apa pun dalam kumpulan dokumen. Secara default, semua koleksi memiliki indeks di `_id` lapangan. Anda dapat menambahkan indeks tambahan untuk mempercepat kueri dan operasi penting.

Indeks bidang tunggal bermanfaat ketika:

- Aplikasi Anda sering menanyakan atau memfilter dokumen berdasarkan nilai bidang tertentu.
- Anda perlu mengurutkan dokumen secara efisien berdasarkan bidang tertentu.
- Anda ingin memastikan keunikan untuk bidang tertentu di seluruh dokumen dalam koleksi dengan membuat indeks unik.

Properti indeks yang didukung

Opsi	3.6	4.0	5.0	8.0	Cluster elastis
name	Ya	Ya	Ya	Ya	Ya
unik	Ya	Ya	Ya	Ya	Ya
jarang *	Ya	Ya	Ya	Ya	Ya

Opsi	3.6	4.0	5.0	8.0	Cluster elastis
FilterExpressionparsial*	Tidak	Tidak	Ya	Ya	Tidak
kedaluwarsaAfterSeconds	Ya	Ya	Ya	Ya	Ya

* `partialFilterExpression` Opsi `sparse` dan tidak dapat digunakan bersama dalam definisi indeks yang sama. Jika Anda mencoba membuat indeks dengan opsi ini, itu akan gagal dengan kesalahan berikut:

```
Error in specification: cannot mix partialFilterExpression and sparse options
```

Membuat indeks bidang tunggal

Gunakan `createIndex()` metode ini untuk membuat indeks bidang tunggal. Sintaksnya adalah:`db.collection.createIndex(<key>, <options>)`.

`key`Parameternya adalah dokumen JSON yang menentukan bidang dan urutan pengurutan indeks:

```
{
  "<field>": <1 (ascending)|-1 (descending)>
}
```

`options`Parameternya adalah dokumen JSON yang menentukan opsi untuk indeks:

```
{
  "name": "<name>",
  "unique": <true | false>,
  "sparse": <true | false>,
  "partialFilterExpression": <filter expression>,
  "expireAfterSeconds": <seconds before expiry>
}
```

Lihat Properti Indeks untuk contoh pembuatan indeks bidang tunggal.

Indeks Senyawa

Indeks majemuk menyimpan informasi dari dua atau lebih bidang dalam kumpulan dokumen, memungkinkan kueri pada bidang pertama atau bidang awalan apa pun. Indeks ini mengoptimalkan kinerja untuk kueri yang memfilter pada beberapa bidang secara bersamaan atau menggabungkan pemfilteran dengan operasi penyortiran. Mereka juga efektif untuk kueri kondisi tunggal di bidang terindeks paling kiri. Basis data memanfaatkan entri indeks ini untuk secara efisien menemukan dokumen yang cocok tanpa melakukan pemindaian koleksi lengkap.

Indeks bidang majemuk bermanfaat ketika:

- Anda perlu memfilter pada beberapa bidang secara bersamaan.
- Anda perlu menggabungkan pemfilteran dengan operasi penyortiran.

Properti indeks yang didukung

Opsi	3.6	4.0	5.0	8.0	Cluster elastis
name	Ya	Ya	Ya	Ya	Ya
unik	Ya	Ya	Ya	Ya	Ya
jarang *	Ya	Ya	Ya	Ya	Ya
FilterExpressionparsial*	Tidak	Tidak	Ya	Ya	Tidak

* `partialFilterExpression` Opsi `sparse` dan tidak dapat digunakan bersama dalam definisi indeks yang sama. Jika Anda mencoba membuat indeks dengan opsi ini, itu akan gagal dengan kesalahan berikut:

```
Error in specification: cannot mix partialFilterExpression and sparse options
```

Membuat indeks majemuk

Gunakan `createIndex()` metode ini untuk membuat indeks majemuk. Sintaks metode adalah: `db.collection.createIndex(<keys>, <options>)`

`keys`Parameternya adalah dokumen JSON yang menentukan bidang dan urutan pengurutan indeks:

```
{
  "<field 1>": <1 (ascending)|-1 (descending)>,
  "<field 2>": <1 (ascending)|-1 (descending)>,
  ...
}
```

Perhatikan bahwa hanya satu bidang yang dapat menjadi array dalam indeks majemuk. Jika Anda mencoba membuat indeks gabungan pada dua atau lebih bidang array, itu akan gagal dengan kesalahan berikut:

```
multiple fields of compound index cannot be arrays
```

`options`Parameternya adalah dokumen JSON yang menentukan opsi untuk indeks:

```
{
  "name": "<name>",
  "unique": <true | false>,
  "sparse": <true | false>,
  "partialFilterExpression": <filter expression>
}
```

Lihat Properti Indeks untuk contoh pembuatan indeks majemuk.

Multi-key Indeks

Untuk bidang yang memiliki nilai array, indeks multi-kunci memungkinkan Anda membuat kunci indeks untuk setiap elemen dalam array. Pengindeksan array menciptakan entri indeks untuk setiap elemen array.

Multi-key indeks bermanfaat ketika aplikasi Anda sering menanyakan atau memfilter dokumen berdasarkan nilai dalam array.

Properti indeks yang didukung

Opsi	3.6	4.0	5.0	8.0	Cluster elastis
name	Ya	Ya	Ya	Ya	Ya
unik	Ya	Ya	Ya	Ya	Ya
jarang *	Ya	Ya	Ya	Ya	Ya
FilterExpressionparsial*	Tidak	Tidak	Ya	Ya	Tidak
kedaluwarsaAfterSeconds	Ya	Ya	Ya	Ya	Ya

* `partialFilterExpression` Opsi `sparse` dan tidak dapat digunakan bersama dalam definisi indeks yang sama. Jika Anda mencoba membuat indeks dengan opsi ini, itu akan gagal dengan kesalahan berikut:

```
Error in specification: cannot mix partialFilterExpression and sparse options
```

Membuat indeks multi-kunci

Gunakan `createIndex()` metode ini untuk membuat indeks multi-kunci. Sintaks metode adalah: `db.collection.createIndex(<key>, <options>)`

`key`Parameternya adalah dokumen JSON yang menentukan bidang dan urutan pengurutan indeks:

```
{
  "<field>": <1 (ascending)|-1 (descending)>
}
```

`options`Parameternya adalah dokumen JSON yang menentukan opsi untuk indeks:

```
{
  "name": "<name>",
```

```
  "unique": <true | false>,
  "sparse": <true | false>,
  "partialFilterExpression": <filter expression>,
  "expireAfterSeconds": <seconds before expiry>
}
```

Contoh berikut membuat indeks multi-kunci pada `categories` bidang yang diurutkan dalam urutan menaik dengan nama: `book_categories`

```
db.collection.createIndex(
  {
    "categories": 1
  },
  {
    "name": "book_categories"
  }
)
```

Lihat Properti Indeks untuk contoh pembuatan indeks multi-kunci.

Indeks Teks

Indeks teks berguna dalam mencari kata atau frasa di bidang teks dalam dokumen, memungkinkan Anda untuk melakukan fungsi pencarian teks lengkap. Anda dapat membuat indeks teks bidang tunggal atau indeks teks gabungan dengan lebih dari satu bidang teks. Namun, Anda hanya dapat memiliki satu indeks teks per koleksi. Untuk informasi selengkapnya, lihat Membuka kunci kekuatan penelusuran teks Amazon DocumentDB dengan kasus penggunaan dunia nyata.

Properti indeks yang didukung

Opsi	3.6	4.0	5.0	8.0	Cluster elastis
name	Tidak	Tidak	Ya	Ya	Tidak

Membuat indeks teks

Gunakan `createIndex()` metode ini untuk membuat indeks teks. Sintaks metode adalah: `db.collection.createIndex(<keys>, <options>)`

`keys`Parameter adalah dokumen JSON yang menentukan bidang (s) dan jenis indeks teks:

```
{
  "<field 1>": "text",
  "<field 2>": "text"
}
```

`options`Parameter adalah dokumen JSON yang menentukan nama indeks:

```
{
  "name": "<name>"
}
```

Lihat Index Properties untuk contoh pembuatan indeks teks.

Indeks Geospasial

Indeks geospasial adalah jenis indeks khusus yang dirancang untuk secara efisien menanyakan dan mengelola data geospasial yang disimpan dalam kumpulan dokumen. Amazon DocumentDB mendukung indeks 2dsphere, yang secara khusus dirancang untuk menangani data geospasial pada bola (seperti Bumi). Ini memungkinkan perhitungan dan kueri yang akurat berdasarkan geometri bola.

Indeks geospasial bermanfaat ketika aplikasi Anda perlu melakukan kueri berbasis lokasi, seperti:

- menemukan tempat menarik terdekat,
- menentukan apakah suatu lokasi berada dalam area tertentu
- menghitung jarak antar lokasi

Properti indeks yang didukung

Opsi	3.6	4.0	5.0	8.0	Cluster elastis
name	Ya	Ya	Ya	Ya	Ya

Membuat indeks geospasial

Gunakan `createIndex()` metode ini untuk membuat indeks geospasial. Sintaks metode adalah: `db.collection.createIndex(<key>, <options>)`

`key`Parameter adalah dokumen JSON yang menentukan bidang dan jenis indeks 2dsphere:

```
{
  "<field>": "2dsphere"
}
```

`options`Parameternya adalah dokumen JSON yang menentukan opsi untuk indeks:

```
{
  "name": "<name>"
}
```

Lihat Properti Indeks untuk contoh pembuatan indeks geospasial.

Indeks vektor

Indeks vektor adalah jenis indeks khusus yang dirancang untuk secara efisien menanyakan dan mengelola data vektor yang disimpan dalam kumpulan dokumen. Amazon DocumentDB mendukung indeks Hierarchical Navigable Small World (HNSW) dan Inverted File with Flat Compression (IVFFlat).

Untuk informasi selengkapnya, lihat Pencarian vektor untuk Amazon DocumentDB.

Indeks vektor bermanfaat untuk pembelajaran mesin dan kasus penggunaan AI generatif, seperti:

- pencarian semantik
- rekomendasi produk
- personalisasi
- chatbots
- deteksi penipuan
- deteksi anomali

Properti indeks yang didukung

Opsi	3.6	4.0	5.0	8.0	Cluster elastis
name	Tidak	Tidak	Ya	Ya	Tidak

Membuat indeks vektor

Gunakan perintah createIndex dengan `runCommand()` metode untuk membuat indeks vektor. Sintaksnya adalah:

```
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]
    }
  }]
})
```

Parameter kunci adalah dokumen JSON yang menentukan bidang dan jenis indeks vektor:

```
{
  "<field>": "vector"
}
```

Lihat Properti Indeks untuk contoh pembuatan indeks vektor.

Properti Indeks

Properti indeks menentukan perilaku dan karakteristik indeks di Amazon DocumentDB. Memahami properti ini membantu Anda merancang indeks efisien yang memenuhi persyaratan spesifik aplikasi Anda.

Topik

- Indeks Properti: nama
- Indeks Properti: unik
- Indeks Properti: jarang
- Indeks Properti: sebagian FilterExpression
- Properti Indeks: kedaluwarsa AfterSeconds

Indeks Properti: nama

Jenis indeks yang didukung

Opsi	3.6	4.0	5.0	8.0	Cluster elastis
bidang tunggal	Ya	Ya	Ya	Ya	Ya
senyawa	Ya	Ya	Ya	Ya	Ya
multi-tombol	Ya	Ya	Ya	Ya	Ya
text	Tidak	Tidak	Ya	Ya	Tidak
geospasial	Ya	Ya	Ya	Ya	Ya
vektor	Tidak	Tidak	Ya	Ya	Tidak

Gunakan opsi nama untuk memberikan nama opsional untuk indeks.

Semua contoh menggunakan dokumen contoh berikut:

```
{
  "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')
}
```

Bidang tunggal

```
db.collection.createIndex(
  {
    "productId": 1
  },
  {
    "name": "single_field_index"
  }
)
```

Senyawa

```
db.collection.createIndex(
  {
    "productId": 1,
    "manufacturer": 1
  },
  {
    "name": "compound_index"
  }
)
```

Multi-key

```
db.collection.createIndex(
  {
    "tags": 1
  },
  {
    "name": "multikey_index"
  }
)
```

Teks

```
db.collection.createIndex(
  {
    "name": "text"
  },
  {
    "name": "text_index"
  }
)
```

Geospasial

```
db.collection.createIndex(
  {
    "supplier.location": "2dsphere"
  },
```

```
  {
    "name": "geospatial_index"
  }
)
```

vektor

```
db.runCommand({
  "createIndexes": "collection",
  "indexes": [{
    "key": {
      "productEmbedding": "vector"
    },
    "name": "hnsw_index",
    "vectorOptions": {
      "type": "hnsw",
      "dimensions": 2,
      "similarity": "euclidean",
      "m": 16,
      "efConstruction": 64
    }
  }]
})
```

Indeks Properti: unik

Jenis indeks yang didukung

Opsi	3.6	4.0	5.0	8.0	Cluster elastis
bidang tunggal	Ya	Ya	Ya	Ya	Ya
senyawa	Ya	Ya	Ya	Ya	Ya
multi-tombol	Ya	Ya	Ya	Ya	Ya

Gunakan opsi unik untuk memastikan keunikan bidang di seluruh dokumen dalam koleksi.

Contoh

Contoh berikut menunjukkan cara membuat indeks unik pada dokumen contoh berikut:

```
{
  "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')
}
```

bidang tunggal

Buat indeks bidang tunggal yang unik di productID untuk memastikan bahwa productID yang sama tidak ada di lebih dari 1 dokumen:

```
db.collection.createIndex(
  {
    "productId": 1
  },
  {
```

```
    name: "productId_unique",
    unique: true
  }
)
```

Senyawa

Buat indeks senyawa unik pada sku dan pabrikan untuk memastikan bahwa kombinasi sku dan pabrikan yang sama tidak ada di lebih dari 1 dokumen:

```
db.collection.createIndex(
  {
    "sku": 1,
    "manufacturer": 1
  },
  {
    name: "sku_and_manufacturer_unique",
    unique: true
  }
)
```

multi-tombol

Buat indeks multi-kunci unik pada barcode untuk memastikan bahwa nilai apa pun dalam array barcode tidak ada di lebih dari 1 dokumen:

```
db.collection.createIndex(
  {
    "barcodes": 1
  },
  {
    name: "barcodes_unique",
    unique: true
  }
)
```

Pengindeksan array menciptakan entri indeks untuk setiap elemen array. Misalnya, jika sebuah array memiliki 50 item, ia memiliki 50 entri indeks. Karena itu, indeks multi-kunci yang unik menegakkan keunikan di semua item individual. Misalnya, dokumen berikut akan melanggar batasan unik pada indeks bidang array nilai:

```
{ "values": [ 1, 2, 3] }
```

```
{ "values": [ 3, 2 ] }   --> 3 and 2 already exist
{ "values": [ 1 ] }      --> 1 already exists
```

Perhatikan perilaku berikut dengan indeks unik:

1. Jika Anda membuat indeks unik pada data yang ada di mana dua (atau lebih) dokumen memiliki nilai yang sama untuk bidang yang diindeks, build indeks akan gagal dengan kesalahan berikut: `could not create unique index: <collection> index: <index name>`
2. Jika Anda menyisipkan dokumen di mana nilai bidang yang diindeks cocok dengan nilai bidang itu di dokumen lain, penyisipan akan gagal dengan kesalahan berikut: `E11000 duplicate key error collection: <collection> index: <index name>`
3. Jika Anda memperbarui dokumen yang ada sehingga nilai baru dari bidang yang diindeks cocok dengan nilai bidang tersebut di dokumen lain, pembaruan akan gagal dengan kesalahan berikut: `E11000 duplicate key error collection: <collection> index: <index name>`
4. Jika bidang yang diindeks hilang dari dokumen, nilainya akan diperlakukan sebagai null. Pembuatan indeks, penyisipan, dan pembaruan akan gagal seperti dijelaskan di atas jika bidang yang diindeks hilang dari dua (atau lebih) dokumen.

Indeks Properti: jarang

Jenis indeks yang didukung

Opsi	3.6	4.0	5.0	8.0	Cluster elastis
bidang tunggal	Ya	Ya	Ya	Ya	Ya
senyawa	Ya	Ya	Ya	Ya	Ya
multi-tombol	Ya	Ya	Ya	Ya	Ya

Gunakan opsi sparse untuk melewati dokumen pengindeksan yang kehilangan bidang yang diindeks, mengurangi ukuran indeks dan menghemat ruang dalam memori. Karena ukuran indeks lebih kecil, kueri yang menggunakannya lebih efisien. Agar kueri menggunakan indeks jarang, Anda harus

menggunakan klausa $exists pada bidang yang diindeks. Jika Anda menghilangkan klausa $exists, Amazon DocumentDB tidak akan menggunakan indeks sparse.

Contoh

Contoh berikut menunjukkan cara membuat indeks sparse pada dokumen contoh berikut:

```
{
  "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')
}
```

Perhatikan bahwa ulasan, materi, dan bidang warna tidak ada di semua dokumen.

bidang tunggal

Buat indeks bidang tunggal yang jarang pada bidang material:

```
db.collection.createIndex(
```

```
  {
    "material": 1
  },
  {
    "name": "material_sparse",
    "sparse": true
  }
)
```

Indeks ini akan digunakan saat menemukan semua produk yang memiliki nilai material:

```
db.collection.find({
  "material": {
    $exists: true
  }
})
```

majemuk

Buat indeks senyawa jarang pada bidang material dan warna:

```
db.collection.createIndex(
  {
    "material": 1,
    "color": 1
  },
  {
    "name": "material_and_color_sparse",
    "sparse": true
  }
)
```

Indeks ini akan digunakan saat menemukan semua produk yang memiliki kombinasi nilai material dan warna:

```
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
      }
    }
  ]
})
```

multi-tombol

Buat indeks multi-kunci yang jarang pada array ulasan:

```
db.collection.createIndex(
  {
    "reviews": 1
  },
  {
```

```
    "name": "reviews_sparse",
    "sparse": true
  }
)
```

Indeks ini akan digunakan saat menemukan semua produk yang memiliki array ulasan:

```
db.collection.find({
  "reviews": {
    $exists: true
  }
})
```

Indeks Properti: sebagian FilterExpression

Jenis indeks yang didukung

Opsi	3.6	4.0	5.0	8.0	Cluster elastis
bidang tunggal	Tidak	Tidak	Ya	Ya	Tidak
senyawa	Tidak	Tidak	Ya	Ya	Tidak
multi-tombol	Tidak	Tidak	Ya	Ya	Tidak
geospasial	Tidak	Tidak	Tidak	Tidak	Tidak

Gunakan FilterExpression opsi parsi untuk membuat indeks sebagian yang hanya menyertakan dokumen yang memenuhi kondisi filter tertentu. Ini memungkinkan Anda untuk membuat indeks yang lebih efisien dengan mengindeks hanya sebagian dokumen dalam koleksi, daripada mengindeks semua dokumen, mengurangi ukuran indeks dan menghemat ruang dalam memori. Karena ukuran indeks lebih kecil, kueri yang menggunakannya lebih efisien. Amazon DocumentDB akan menggunakan indeks sebagian dalam skenario berikut:

- Predikat kueri sama persis dengan ekspresi filter indeks sebagian.
- Hasil yang diharapkan filter kueri adalah subset logis dari filter paral.

- Sub-predikat kueri dapat digunakan bersama dengan indeks lainnya.

Untuk informasi selengkapnya, lihat Indeks sebagian.

Contoh

Contoh berikut menunjukkan cara membuat indeks sebagian pada dokumen contoh berikut:

```
{
  "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')
}
```

bidang tunggal

Buat indeks bidang tunggal sebagian pada produsen untuk produk yang memiliki warna Space Grey:

```
db.collection.createIndex(
```

```
  {
    "manufacturer": 1
  },
  {
    "name": "manufacturer_space_gray",
    "partialFilterExpression": {
      "color": {
        $eq: "Space Gray"
      }
    }
  }
)
```

Indeks ini akan digunakan saat menemukan produk warna Space Grey oleh produsen:

```
db.collection.find({
  "manufacturer": "The Manufacturer",
  "color": {
    $eq: "Space Gray"
  }
})
```

majemuk

Buat indeks senyawa sebagian pada produsen dan warna untuk produk yang memiliki bahan Polycarbonate:

```
db.collection.createIndex(
  {
    "manufacturer": 1,
    "color": 1
  },
  {
    "name": "manufacturer_and_color_polycarbonate",
    "partialFilterExpression": {
      "material": {
        $eq: "Polycarbonate"
      }
    }
  }
)
```

Indeks ini akan digunakan ketika menemukan produk bahan Polycarbonate oleh produsen dan warna:

```
db.collection.find({
  "manufacturer": "The Manufacturer",
  "color": "Space Gray",
  "material": {
    $eq: "Polycarbonate"
  }
})
```

multi-tombol

Buat indeks multi-kunci sebagian pada tag untuk produk yang memiliki produsen The Manufacturer:

```
db.collection.createIndex(
  {
    "tags": 1
  },
  {
    "name": "tags_the_manufacturer",
    "partialFilterExpression": {
      "manufacturer": {
        $eq: "The Manufacturer"
      }
    }
  }
)
```

Indeks ini akan digunakan ketika menemukan produk yang ditandai dengan printer yang memiliki produsen The Manufacturer:

```
db.collection.find({
  "tags": "printer",
  "manufacturer": {
    $eq: "The Manufacturer"
  }
})
```

Properti Indeks: kedaluwarsa AfterSeconds

Jenis indeks yang didukung

Jenis indeks	3.6	4.0	5.0	8.0	Cluster elastis
bidang tunggal	Ya	Ya	Ya	Ya	Ya
multi-tombol	Ya	Ya	Ya	Ya	Ya

Gunakan AfterSeconds opsi kedaluwarsa untuk membuat indeks time to live (TTL). Indeks TTL memungkinkan Anda untuk menghapus dokumen berdasarkan usia mereka berdasarkan kondisi batas waktu untuk setiap dokumen. Ketika dokumen mencapai usia TTL yang ditentukan, itu dihapus dari koleksi.

Untuk praktik terbaik dengan penghapusan TTL, lihat. TTL dan beban kerja time series

Contoh

Contoh berikut menunjukkan cara membuat indeks TTL pada dokumen contoh berikut:

```
{
  "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')
}
```

bidang tunggal

Buat indeks TTL di LastUpDated untuk menghapus dokumen yang belum diperbarui dalam 90 hari:

```
db.collection.createIndex(
  {
    "lastUpdated": 1
  },
  {
    "name": "lastUpdated_ttl",
    "expireAfterSeconds": 7776000
  }
)
```

multi-tombol

Buat indeks TTL di reviews.review_date untuk menghapus dokumen yang tidak memiliki ulasan dalam setahun terakhir:

```
db.collection.createIndex(
```

```
  {
    "reviews.review_date": 1
  },
  {
    "name": "reviews_review_date_ttl",
    "expireAfterSeconds": 31536000
  }
)
```

Perhatikan bahwa jika indeks TTL berada di bidang array, semua item dalam array dicentang. Jika ada stempel waktu dalam array yang memenuhi kondisi batas waktu, dokumen akan dihapus.

Optimalisasi Kinerja

Bab ini memberikan panduan tentang mengoptimalkan kinerja Amazon DocumentDB dan memecahkan masalah kinerja umum.

Topik

- Masalah Kinerja Umum
- Anti-patterns
- Tips Peningkatan Kinerja
- Analisis Rencana Kueri

Masalah Kinerja Umum

Bagian ini mencakup masalah kinerja umum yang mungkin Anda temui dengan Amazon DocumentDB dan cara mengatasinya.

Topik

- Pemanfaatan CPU Tinggi
- Masalah Koneksi dengan Amazon DocumentDB
- Kueri Berjalan Lambat

Pemanfaatan CPU Tinggi

Pemanfaatan CPU yang tinggi adalah salah satu masalah yang mungkin Anda temui di beberapa titik saat bekerja dengan Amazon DocumentDB. Bagian ini memberikan wawasan untuk menyelesaikan masalah.

Identifikasi - Temukan masalahnya

Pemanfaatan CPU yang tinggi pada instans Amazon DocumentDB menyebabkan kelambatan aplikasi dengan meningkatkan latensi di semua operasi database. Ketika penggunaan CPU Amazon DocumentDB naik di atas level optimal, waktu eksekusi kueri meningkat, pembuatan koneksi membutuhkan waktu lebih lama, dan respons aplikasi berkurang. Puncak CPU pada instance utama operasi penulisan lambat, sementara puncak pada replika menghasilkan latensi yang lebih tinggi untuk operasi baca.

Beberapa Penyebab Umum Pemanfaatan CPU Tinggi di Amazon DocumentDB dapat berupa:

- Pemindaian Koleksi: Kueri tanpa indeks yang tepat memaksa pemindaian koleksi penuh, menghabiskan CPU berlebihan untuk memeriksa setiap dokumen.
- Lonjakan Koneksi: Lonjakan tiba-tiba dalam koneksi aplikasi membanjiri sumber daya CPU yang diperlukan untuk otentikasi dan manajemen sesi.
- Agregasi Kompleks: Multi-stage operasi (penyortiran, pengelompokan, penggabungan) mengkonsumsi CPU yang signifikan, terutama pada kumpulan data besar tanpa indeks yang dioptimalkan.
- Kueri Tidak Efisien: Penggunaan CPU yang tinggi dihasilkan dari pemindaian set hasil besar dan operasi regex.
- Write-Heavy Beban kerja: Pembaruan yang sering dan penyisipan massal membebani CPU dengan pemrosesan dokumen, pembaruan indeks, dan pencatatan transaksi.

Selidiki - Kumpulkan metrik

Amazon DocumentDB menyediakan pemantauan melalui Amazon. CloudWatch Metrik Amazon DocumentDB dapat dikategorikan secara luas ke dalam metrik tingkat cluster dan tingkat instans. Untuk sumber daya yang terkait dengan CPU, memori, dan koneksi, Lihatlah metrik tingkat instans karena konsumsi sumber daya khusus untuk instance. Identifikasi instance (primer atau replika) yang menunjukkan lonjakan CPU dan catat waktu lonjakan.

CloudWatch Metrik untuk pemanfaatan CPU untuk`CPUUtilization`.

Diagnosa - Temukan akar penyebabnya

Setelah mengidentifikasi instance dan timeline lonjakan CPU, selanjutnya adalah mendiagnosis penyebab yang memengaruhi penggunaan CPU secara besar-besaran. Karena ada beberapa alasan yang dapat menyebabkan lonjakan CPU, mari kita identifikasi apa yang berkontribusi padanya dimulai dengan:

Koneksi: Periksa CloudWatch metrik termasuk`DatabaseConnectionsMax`, `DatabaseConnections` dan, jika ada lonjakan koneksi dan bertepatan dengan waktu lonjakan CPU, lonjakan koneksi dalam waktu singkat kemungkinan merupakan penyebab lonjakan CPU.

Kueri: Jika bukan koneksi, maka kueri mungkin menjadi penyebabnya. Jika Anda telah mengaktifkan insight performa, buka konsol Performance insights untuk instance tersebut dan lihat kueri pada instance tersebut. Hal lain untuk melihat kueri adalah dari profiler.

Selesaikan - Perbaiki masalah

Semburan koneksi: Jika semburan koneksi adalah penyebab lonjakan CPU, pertimbangkan untuk menggunakan kembali koneksi dalam aplikasi Anda atau menerapkan penyatuan koneksi. Untuk informasi selengkapnya tentang mengoptimalkan koneksi tinggi, lihatthe section called “Masalah Koneksi”.

Distribusi Kueri: Jika lonjakan CPU disebabkan oleh kueri baca pada instance utama, pertimbangkan untuk mengarahkan operasi baca ke instance replika jika hasil kueri Anda pada akhirnya dapat konsisten.

Pengoptimalan kueri: Jika kueri atau kumpulan kueri menyebabkan CPU melonjak, pertimbangkan untuk mengoptimalkan kueri seperti menambahkan indeks jika kueri tidak menggunakannya. Untuk informasi selengkapnya tentang mengoptimalkan kueri, lihatthe section called “Kueri Berjalan Lambat”.

Penskalaan Instance: Jika pemanfaatan CPU tetap tinggi secara konsisten setelah upaya pengoptimalan, tingkatkan ke kelas instans yang lebih besar dengan kapasitas CPU yang lebih besar untuk menangani kebutuhan beban kerja Anda secara efektif.

Tanpa server: Untuk beban kerja yang tidak dapat diprediksi yang menyebabkan lonjakan CPU berdurasi pendek, pertimbangkan untuk menggunakan. Menggunakan Amazon DocumentDB tanpa server Instans tanpa server secara otomatis menskalakan sumber daya berdasarkan permintaan, sehingga tidak perlu menyediakan ukuran instans tetap untuk beban kerja tersebut.

Masalah Koneksi dengan Amazon DocumentDB

Identifikasi - Temukan masalahnya

Penyebab Umum

Masalah koneksi biasanya berasal dari tiga bidang utama:

Kelelahan kumpulan koneksi terjadi ketika aplikasi mencapai koneksi maksimum yang diizinkan ke Amazon DocumentDB, baik melalui batas kumpulan koneksi sisi klien atau batas instans sisi server. Kondisi ini menyebabkan penurunan kinerja aplikasi, batas waktu, dan potensi kegagalan karena permintaan koneksi baru diantrian atau ditolak.

Overload otentikasi terjadi ketika Amazon DocumentDB mengalami permintaan otentikasi bersamaan yang berlebihan, terutama saat memproses lebih dari 1.000 koneksi baru dalam waktu singkat.

Selama pemeliharaan otentikasi, Amazon DocumentDB memegang kunci eksklusif pada peta sesi, menyebabkan upaya otentikasi berikutnya untuk mengantri hingga pemeliharaan selesai.

Masalah konfigurasi di Amazon DocumentDB sering kali berasal dari kesalahan konfigurasi dalam pengaturan jaringan, keamanan, dan klien. Ini termasuk item seperti pengaturan grup keamanan yang salah, konfigurasi VPC yang tidak tepat, atau masalah SSL/TLS sertifikat. Memahami konfigurasi yang tepat sangat penting untuk menjaga akses database yang aman dan andal.

Diagnosa - Temukan akar penyebabnya

Kolam koneksi

Kumpulan koneksi menginisialisasi saat membuat MongoClient instance. Setiap pool memelihara koneksi berdasarkan dua parameter utama:

min PoolSize - Jumlah minimum koneksi dipertahankan

max PoolSize - Koneksi maksimum yang diizinkan

Ketika permintaan membutuhkan koneksi:

1. Pool memeriksa koneksi idle yang tersedia
2. Jika tidak ada dan ukuran kolam < maksPoolSize, itu menciptakan koneksi baru
3. Jika maksimalPoolSize, permintaan memasuki antrian tunggu
4. Jika antrian penuh atau batas waktu tercapai, itu melempar MongoWaitQueueFullException

Perilaku antrian tunggu ditangani melalui parameter ini:

tunggu QueueTimeout MS - Waktu tunggu maksimum untuk koneksi

tunggu QueueSize - Permintaan antrian maksimum

Berikut adalah contoh pendekatan bermasalah untuk menghubungkan ke Amazon DocumentDB di mana kumpulan baru dibuat setiap kali:

```
for(Request request : requests) {
    MongoClient client = MongoClients.create(settings);
    // Process request
```

```
    client.close();
}
```

CloudWatch Metrik penting untuk dipantau adalah:

- `DatabaseConnections`- Jumlah koneksi (aktif dan idle) terbuka pada instance yang diambil pada frekuensi 1 menit.
- `DatabaseConnectionsMax`- Jumlah maksimum koneksi database terbuka (aktif dan idle) pada instance dalam periode 1 menit.
- `DatabaseConnectionsLimit`- Jumlah maksimum koneksi database bersamaan (aktif dan idle) diizinkan pada sebuah instance pada waktu tertentu.
- `LowMemNumOperationsThrottled`- Jumlah permintaan yang dibatasi karena memori yang tersedia rendah dalam periode 1 menit.

Lihat Kuota dan batas untuk batas per kelas instance.

Tanda-tanda peringatan umum masalah kumpulan koneksi di tingkat aplikasi meliputi:

- Meningkatkan waktu akuisisi koneksi
- Menumbuhkan ukuran antrian tunggu
- Meningkatnya jumlah pengecualian batas waktu

Otentikasi kelebihan

Menghubungkan ke Amazon DocumentDB mengikuti alur yang mirip dengan di bawah ini:

Permintaan Koneksi → Jabat Tangan SSL → Otentikasi → Pembuatan Sesi → Siap Koneksi

Saat memproses >1.000 koneksi baru, permintaan koneksi tambahan akan memasuki antrian untuk Otentikasi setelah menyelesaikan jabat tangan SSL. Waktu koneksi rata-rata dari aplikasi Anda akan meningkat selama kejadian kelebihan beban ini.

CloudWatch Metrik penting untuk dipantau adalah:

- `DatabaseConnections`- Jumlah koneksi (aktif dan idle) terbuka pada instance yang diambil pada frekuensi 1 menit.

- `DatabaseConnectionsMax`- Jumlah maksimum koneksi database terbuka (aktif dan idle) pada instance dalam periode 1 menit.
- `DatabaseConnectionsLimit`- Jumlah maksimum koneksi database bersamaan (aktif dan idle) diizinkan pada sebuah instance pada waktu tertentu.

Masalah konfigurasi

Masalah konfigurasi yang paling umum disebabkan ketika mencoba terhubung ke titik akhir cluster Amazon DocumentDB pribadi dari lingkungan tanpa akses ke lingkungan jaringan pribadi. Amazon DocumentDB adalah virtual private cloud (VPC)-satu-satunya dan saat ini tidak mendukung titik akhir publik. Anda tidak dapat terhubung langsung ke cluster Amazon DocumentDB dari laptop atau lingkungan pengembangan lokal di luar VPC Anda.

Ini akan terwujud dalam kesalahan seperti di bawah ini:

```
Error: couldn't connect to server...
Failed to connect to...
exception: connect failed
connection attempt failed
```

Konfigurasi grup keamanan yang salah juga dapat menyebabkan kegagalan koneksi. Cluster Amazon DocumentDB mendengarkan koneksi pada port TCP 27017 secara default. Aplikasi Anda akan gagal jika mencoba menyambung ke port yang berbeda dari apa yang digunakan oleh cluster, atau jika aplikasi tidak tercakup dalam konfigurasi grup keamanan ingress untuk cluster.

Manajemen sertifikat yang salah juga dapat menyebabkan masalah koneksi. Secara default, enkripsi dalam transit diaktifkan untuk klaster Amazon DocumentDB yang baru dibuat. Ketika enkripsi dalam perjalanan diaktifkan, koneksi aman menggunakan TLS diperlukan untuk terhubung ke cluster menggunakan sertifikat global-bundle.pem. Jika Anda mencoba menggunakan sertifikat yang salah, Anda akan menerima kesalahan seperti:

```
unable to get local issuer certificate
```

Jika mencoba terhubung ke cluster dengan TLS diaktifkan tanpa menentukan parameter TLS, Anda akan menerima kesalahan seperti:

```
Server selection timed out after 30000 ms
```

Selesaikan - Perbaiki masalah

Kumpulan koneksi: Tinjau penyatuan koneksi dengan menerapkan atau menyesuaikan ukuran kolam agar sesuai dengan persyaratan beban kerja. Konfigurasi kolam yang optimal bergantung pada beban kerja dan persyaratan Anda. Anda harus menjaga min PoolSize sedemikian rupa sehingga koneksi inti siap dan tersedia, dan maks cukup WaitTime pendek untuk gagal dengan cepat jika kolam telah habis.

Berikut adalah contoh cara menggunakan kembali satu kolam tanpa membuat yang baru setiap kali:

```
MongoClient client = MongoClients.create(settings);
    for(Request request : requests) {
    // Process request
}
```

Overload otentikasi: Kelola otentikasi dengan menerapkan peningkatan koneksi bertahap dan membatasi koneksi baru menjadi 1.000 sekaligus. Gunakan penyatuan koneksi untuk menggunakan kembali koneksi yang diautentikasi secara efektif. Untuk menghindari kelebihan beban klaster Amazon DocumentDB dengan koneksi, terapkan strategi peningkatan koneksi.

```
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;
        }
    }
}
```

Anda juga dapat mengonfigurasi pengaturan kumpulan koneksi untuk membatasi jumlah total koneksi yang diizinkan.

```
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();
```

Masalah konfigurasi: Pastikan aplikasi Anda memiliki akses ke VPC pribadi dan subnet tempat sumber daya Amazon DocumentDB Anda berada. Jika menggunakan VPC Peering, periksa panduan pengembang Memecahkan masalah koneksi peering VPC untuk informasi lebih lanjut. Anda juga dapat meninjau artikel Pusat Pengetahuan Bagaimana cara mengatasi masalah konektivitas dari internet ke instans Amazon EC2 dalam VPC saya? .

Untuk konfigurasi grup keamanan, Anda harus menyertakan aturan ingress di grup keamanan Amazon DocumentDB untuk mengizinkan koneksi dari aplikasi Anda.

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

Jika cluster dikonfigurasi dengan enkripsi TLS, unduh sertifikat TLS untuk Amazon DocumentDB Amazon bernama global-bundle.pem dan gunakan saat menghubungkan ke cluster.

```
wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
```

Long-term Solusi

Penskalaan instans mungkin diperlukan melalui peningkatan ke kelas instance yang lebih besar atau menambahkan replika baca untuk mendistribusikan beban koneksi. Implementasi load balancing yang tepat memastikan pemanfaatan sumber daya yang optimal di seluruh cluster.

Perubahan aplikasi harus fokus pada penerapan penanganan koneksi yang kuat, pemantauan komprehensif, dan kepatuhan terhadap praktik terbaik penyatuan koneksi. Ini termasuk penanganan kesalahan yang tepat dan manajemen siklus hidup koneksi.

Perbaikan arsitektur mungkin melibatkan adopsi Amazon DocumentDB Tanpa Server untuk beban kerja variabel, menerapkan logika coba ulang yang canggih, dan merancang untuk toleransi kesalahan. Pertimbangkan restrukturisasi arsitektur aplikasi untuk menangani manajemen koneksi dengan lebih baik.

Praktik Terbaik

Kolam koneksi

Melalui manajemen dan pemantauan kumpulan koneksi yang tepat, aplikasi dapat mempertahankan konektivitas database yang stabil sekaligus mencegah skenario kelelahan yang dapat memengaruhi keandalan dan kinerja sistem. Konfigurasikan batas waktu yang sesuai dan ukuran kumpulan Anda berdasarkan karakteristik beban kerja Anda.

Contoh pengaturan kolam koneksi

```
MongoClientSettings settings = MongoClientSettings.builder()
    .applyToConnectionPoolSettings(builder ->
        builder.maxSize(10))
    .applyToConnectionPoolSettings(builder ->
        builder.maxWaitQueueSize(2))
```

```
    .applyToConnectionPoolSettings(builder ->
        builder.maxConnectionIdleTime(10, TimeUnit.MINUTES))
    .build();
```

Untuk informasi lebih lanjut, lihat: https://aws.amazon.com/blogs/database/building-resilient-applications-with-amazon-documentdb-with-mongodb-compatibility-part-1-client-configuration/

Otentikasi kelebihan

Selalu terapkan penyatuan koneksi dengan nilai yang sesuai untuk parameter berdasarkan beban kerja Anda. Gunakan teknik pembentukan koneksi bertahap dan pertahankan koneksi persisten, jika memungkinkan. Menerapkan pembersihan koneksi yang tepat untuk memastikan tidak ada sumber daya idle yang terbuang sia-sia.

Masalah konfigurasi

Pastikan Anda telah mengonfigurasi perutean yang sesuai dari aplikasi Anda ke sumber daya Amazon DocumentDB. Memanfaatkan TLS untuk enkripsi dalam perjalanan dan menerapkan akses hak istimewa paling sedikit. Verifikasi kredensyal Amazon DocumentDB Anda dan validasi nilai string koneksi.

Kueri Berjalan Lambat

Identifikasi - Temukan masalahnya

Kueri yang berjalan lambat adalah kueri yang melebihi waktu eksekusi yang diharapkan untuk kebutuhan bisnis Anda. Definisi dari apa yang merupakan kueri lambat bervariasi di seluruh beban kerja yang berbeda. Anda dapat mengidentifikasi kueri lambat melalui log profiler atau Performance Insights.

1. Aktifkan profiler jika belum diaktifkan. Log profiler ditulis ke Amazon CloudWatch di bawah Grup Log:`/aws/docdb/<cluster-identifier>/profiler`. Gunakan Wawasan CloudWatch Log untuk menanyainya.

   Contoh kueri analisis CloudWatch log untuk mendapatkan 10 kueri paling lambat untuk koleksi ecommerce.products:

   ```
   filter ns="ecommerce.products" | sort millis desc | limit 10
   ```

2. Gunakan Performance Insights untuk mengidentifikasi kueri mahal dalam waktu dekat. Aktifkan Performance Insights jika belum diaktifkan.
   a. Buka AWS Konsol, navigasikan ke Amazon Amazon DocumentDB, lalu Performance Insights, dan pilih instance cluster Anda.
   b. Tinjau timeline DB Load (AAS) dan Kueri Top (berdasarkan beban DB). Perluas intisari kueri untuk melihat pernyataan turunan literal.
   c. Tangkap kueri yang membutuhkan analisis.

> **Note**
>
> Tidak semua kueri dalam Performance Insights mungkin merupakan kueri yang tidak efisien atau lambat.

Selidiki - Kumpulkan informasi

1. Profiler menyediakan rencana eksekusi kueri dan metrik kunci yang terkait dengannya, termasuk millis, nreturned, dan planSummary (penggunaan indeks):

```
{
    "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"
}
```

Untuk menemukan kueri COLLSCAN:

```
filter planSummary="COLLSCAN" | sort millis desc | limit 20
```

2. Gunakan Performance Insights untuk menganalisis tren eksekusi kueri, seperti menunggu, memuat, dan dampak sumber daya (misalnya, IOPS atau CPU per kueri) secara real time.

   Karena Performance Insights tidak menyediakan paket eksekusi kueri, ambil kueri dan jalankan penjelasan (“ExecutionStats”) pada kueri di klaster Amazon DocumentDB Anda:

```
db.ecommerce.products.find().explain("executionStats")
```

3. Secara opsional, korelasikan metrik profiler dengan data Performance Insights (misalnya, mencocokkan kueri millis tinggi di profiler dengan kueri teratas di Performance Insights).

Diagnosa — Temukan akar penyebab

Pada langkah ini, diagnosa rencana kueri untuk mengidentifikasi potensi pengoptimalan. Gunakan aliran berikut - gejala → kemungkinan penyebabnya:

- Gejala: PlanSummary: “COLLSCAN”

  Penyebab: Indeks hilang atau salah.
- Gejala: agregasi lambat dengan $group/$sort

  Penyebab: Pipa agregasi mungkin memproses terlalu banyak data dalam memori.
- Gejala: latensi tinggi sementara Performance Insights menunjukkan DB Load dikelompokkan berdasarkan IO waits, meskipun indeks digunakan (PlanSummary: “IXSCAN”).

  Penyebab: kueri I/O terikat; indeks atau set kerja melebihi cache buffer yang tersedia pada instance.
- Gejala: PI menunjukkan status menunggu CPU, AAS tinggi karena beberapa kueri

  Penyebab: CPU-bound kueri (agregasi kompleks, $regex).
- Gejala: banyak kueri lambat tetapi tidak ada yang menunjukkan PlanSummary yang mahal

  Penyebab: kemacetan eksternal (jaringan, pelambatan, tugas pemeliharaan, volume kueri).

Selesaikan — Perbaiki masalah

Saat Anda menerapkan perbaikan, selalu validasi dengan explain (“ExecutionStats”) dan pantau Performance Insights DB Load.

1. PlanSummary: “COLLSCAN”
   - Buat indeks yang ditargetkan.

     Contoh: Untuk kueri yang sering memfilter berdasarkan {kategori, harga} dan mengurutkan berdasarkan harga turun:

```
db.products.createIndex({ category: 1, price: -1 })
```

2. agregasi lambat dengan $group/$sort
   - Dorong $match dan $project lebih awal untuk mengurangi dokumen yang mengalir ke $group/$sort.
   - Batasi jumlah bidang di awal pipa:

```
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. latensi tinggi sementara Performance Insights menunjukkan DB Load dikelompokkan berdasarkan IO waits
   - Validasi BufferCacheHitRatio jika rendah atau ReadiOps pada instance tinggi. Tingkatkan memori instance (tingkatkan kelas instance, misalnya, r6g.large → r6g.xlarge) atau gunakan kelas instance NVMe.
   - Kurangi jejak indeks.
   - Tambahkan replika baca untuk membongkar lalu lintas baca (gunakan pengaturan ReadPreference untuk mengarahkan lalu lintas pada replika jika kueri toleran dengan konsistensi akhirnya).
4. PI menunjukkan status menunggu CPU, AAS tinggi karena beberapa kueri
   - Ganti $regex yang mahal dengan pencarian awalan yang diindeks atau indeks $text.
   - Batch write (insert) untuk mengurangi amplifikasi tulis.

> **Note**
>
> Selalu uji perubahan Anda di lingkungan yang lebih rendah (non-produksi) sebelum mempromosikan perubahan tersebut ke Produksi.

Anti-patterns

Bagian ini mencakup anti-pola umum yang dapat berdampak negatif pada kinerja Amazon DocumentDB.

Topik

- Indeks majemuk dengan lebih dari 3 atribut
- Kueri Berjalan Panjang
- Indeks yang digunakan rendah dan redundan
- Multi-Key Indeks dengan Array Besar

Indeks majemuk dengan lebih dari 3 atribut

Ikhtisar

Indeks senyawa atau komposit mempertahankan referensi ke beberapa bidang dalam struktur indeks tunggal. Indeks ini mengoptimalkan kinerja untuk kueri yang memfilter pada beberapa bidang secara bersamaan atau menggabungkan pemfilteran dengan operasi penyortiran. Mereka juga efektif untuk kueri kondisi tunggal di bidang terindeks paling kiri. Basis data memanfaatkan entri indeks ini untuk secara efisien menemukan dokumen yang cocok tanpa melakukan pemindaian koleksi lengkap.

Amazon DocumentDB dapat menggunakan indeks gabungan untuk mendukung kueri yang menyertakan subset terdepan dari bidang yang diindeks, sebuah konsep yang dikenal sebagai Index Prefixes. Misalnya, jika Anda memiliki indeks gabungan`{state: 1, city: 1, zipcode: 1}`, Amazon DocumentDB dapat secara efisien memproses kueri yang menggunakan bidang 'status' saja, bidang 'status' dan 'kota' bersama-sama, atau ketiga bidang 'status', 'kota', dan 'zipcode'. Namun, kueri harus menggunakan bidang dari kiri ke kanan tanpa melewatkan bidang apa pun di antaranya. Ini berarti kueri hanya menggunakan bidang 'kota' atau 'kode pos', atau kombinasi seperti 'negara' dan 'kode pos' (melewatkan 'kota'), tidak dapat sepenuhnya memanfaatkan indeks.

Dalam sebagian besar skenario dunia nyata, indeks gabungan dengan tiga atau lebih sedikit atribut biasanya mencapai kinerja optimal dan efisiensi sumber daya. Meskipun layak untuk memiliki lebih dari 3 atribut dalam indeks komposit, sering kali mengarah pada peningkatan konsumsi sumber daya, melebihi manfaatnya.

Dampak pada cluster

Meskipun tergoda untuk membuat indeks yang mencakup semua kondisi dalam kueri untuk mencapai kinerja teoritis maksimum, sebagian besar penyaringan data terjadi melalui 1-3 atribut pertama dalam indeks komposit. Bidang tambahan di luar ambang batas ini berkontribusi terutama pada ukuran indeks daripada pengoptimalan kueri yang berarti.

Penyimpanan dan I/O Overhead: Indeks komposit dengan banyak atribut mengkonsumsi ruang penyimpanan yang jauh lebih banyak daripada alternatif yang lebih sederhana. Ukurannya berbanding lurus dengan jumlah atribut yang diindeks, dan ukuran nilai yang diindeks itu sendiri.

Jejak memori: Jejak penyimpanan besar indeks komposit dengan banyak atribut menciptakan jejak yang sesuai dalam memori, membuat set kerja Anda lebih besar dan memindahkan data lain yang sering diakses dari kumpulan buffer.

Operasi Tulis: Setiap modifikasi dokumen yang memengaruhi beberapa bidang yang diindeks memerlukan pembaruan seluruh entri indeks komposit, mengalikan pekerjaan yang diperlukan untuk menyelesaikan operasi penulisan.

Bagaimana cara mengidentifikasi

Mulailah dengan meninjau semua indeks dalam koleksi Anda untuk mengidentifikasi indeks komposit dengan lebih dari tiga atribut:

```
// 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 }
```

Remediasi

Jika indeks komposit dengan lebih dari tiga atribut sedang digunakan, identifikasi kueri yang menggunakannya dan cari peluang optimasi. Pertimbangkan untuk mengganti indeks ini dengan yang lebih efisien yang mengandung tiga atau lebih sedikit atribut. Setelah menerapkan indeks baru, hapus indeks lama yang berisi lebih dari tiga atribut.

> **Note**
>
> Selalu berkoordinasi dengan pemangku kepentingan dan validasi dampak kinerja sebelum menjatuhkan indeks apa pun.

Ikuti aturan Equality, Sort, Range (ESR) saat membuat indeks komposit.

Aturan Equality Sort Range (ESR): Urutan ini memaksimalkan efisiensi indeks dengan memfilter kumpulan data terlebih dahulu dengan kondisi kesetaraan, kemudian menerapkan operasi pengurutan pada set yang dikurangi, dan akhirnya melakukan pemindaian rentang pada subset sekecil mungkin

- bidang pesanan sebagai Kesetaraan (kecocokan persis),
- Urutkan (memesan),
- Rentang (>, <, $ 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
```

Kueri Berjalan Panjang

Ikhtisar

Long-running kueri dapat mengalir ke masalah kinerja di seluruh klaster. Kueri ini dapat mengganggu proses pengumpulan sampah Multi-Version Concurrency Control (MVCC) Amazon DocumentDB, yang menyebabkan akumulasi dokumen berversi lama dan penurunan kinerja di seluruh cluster. Amazon DocumentDB menerapkan batas waktu sisi server 2 jam sebagai mekanisme keamanan untuk membatasi kueri pelarian dari mengkonsumsi sumber daya tanpa batas waktu.

Apa itu Kueri yang berjalan lama:

- Kueri yang dijalankan untuk waktu yang lama (biasanya> 30 menit)
- Buka kursor yang tetap aktif selama berjam-jam (waktu habis 2 jam tidak akan berlaku jika kursor aktif)

Dampak pada cluster

Kueri yang berjalan lama dapat mengganggu Proses pengumpulan Sampah

- Versi Lama Akumulasi: Pengumpul sampah tidak dapat merebut kembali versi dokumen lama
- Collection and Index Bloat: Entri Collection and Index terakumulasi dari waktu ke waktu, kembung meningkat dan yang dapat menghasilkan lebih banyak biaya penyimpanan.
- CPU dan Tekanan Memori: CPU dan tekanan memori meningkat karena pemrosesan yang tidak efisien dari peningkatan jumlah versi dokumen lama, entri indeks, dan ID transaksi yang tidak efisien.

Kueri Berjalan Panjang → Memblokir GC → Pertumbuhan Penyimpanan → CPU dan Tekanan Memori → Kueri Lebih Panjang

Memantau dan mendeteksi

1. Untuk menemukan kueri yang berjalan lama, gunakan perintah 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. Untuk menemukan kursor yang aktif selama lebih dari 30 menit

```
// 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. Memantau kemajuan pengumpul sampah melalui CloudWatch

`LongestRunningGCProcess`— Durasi dalam hitungan detik dari proses pengumpulan sampah aktif terpanjang. Memperbarui setiap menit dan hanya melacak operasi aktif, tidak termasuk proses yang selesai dalam jendela satu menit.

`AvailableMVCCIds`-Penghitung yang menunjukkan jumlah operasi penulisan yang tersisa yang tersedia sebelum mencapai nol. Saat penghitung ini mencapai nol, klaster Anda memasuki mode hanya-baca hingga ID direklamasi dan didaur ulang. Penghitung berkurang dengan setiap operasi penulisan dan meningkat saat pengumpulan sampah mendaur ulang ID MVCC lama.

> **Note**
>
> ID MVCC yang lebih rendah dan durasi pengumpulan sampah yang diperpanjang tidak secara eksklusif dikaitkan dengan kueri yang berjalan lama. Write-intensive beban kerja pada instans yang dibatasi sumber daya juga dapat mengakibatkan berkurangnya ketersediaan ID MVCC dan siklus pengumpulan sampah yang berkepanjangan.

Strategi Remediasi

- Menerapkan batas waktu kueri dalam aplikasi
- Jangan biarkan kursor tetap hidup untuk jangka waktu yang lebih lama
- Optimalkan kueri untuk kinerja yang lebih baik.
- Lebih suka pengelompokan operasi tulis

Indeks yang digunakan rendah dan redundan

Ikhtisar

Indeks meningkatkan kinerja kueri keseluruhan beban kerja, tetapi datang dengan biaya. Setiap indeks mengkonsumsi penyimpanan dan I/O sumber daya, dan operasi penulisan perlu memperbarui semua indeks pada koleksi. Jika database Anda berisi indeks yang tidak digunakan, kurang dimanfaatkan, atau berlebihan, itu menghasilkan overhead ke cluster, yang menyebabkan penurunan kinerja. Beberapa indeks yang tidak perlu menciptakan anti-pola yang menyebabkan masalah kinerja, peningkatan biaya penyimpanan, dan overhead operasional yang lebih tinggi.

Skenario pengindeksan sub optimal

- Indeks yang Tidak Digunakan: Indeks dibuat untuk kueri satu kali, atau dalam iterasi produk sebelumnya yang tidak lagi diakses oleh pola kueri saat ini. Untuk mengidentifikasi indeks yang tidak digunakan pada koleksi, Anda dapat menggunakan IndexStats. Untuk informasi selengkapnya, lihat the section called “Menganalisis penggunaan indeks dan mengidentifikasi indeks yang tidak digunakan”.
- Indeks Redundan: Beberapa indeks yang mencakup pola kunci yang tumpang tindih atau pola kueri yang identik. Contoh: Indeks kunci tunggal pada atribut A menjadi berlebihan ketika indeks komposit mencakup atribut A dan B, karena indeks komposit dapat menangani kueri pada atribut A saja.
- Over-Indexing: Membuat indeks “berjaga-jaga” tanpa menganalisis pola kueri aktual atau persyaratan kinerja.
- Indeks Kardinalitas Rendah: Indeks pada bidang dengan beberapa nilai berbeda (misalnya, bidang boolean, bendera status) yang memberikan manfaat pengoptimalan kueri minimal.

Dampak pada Kinerja Cluster

- Penyimpanan dan IO: Setiap indeks mengkonsumsi ruang penyimpanan dan sumber daya IO, berkontribusi terhadap biaya keseluruhan. Jika indeks tidak digunakan atau berlebihan, itu mewakili biaya yang tidak perlu pada cluster Anda.
- Kinerja Tulis Terdegradasi: Operasi penyisipan, perbarui, dan hapus harus mempertahankan indeks, membuat overhead yang menghabiskan sumber daya. Indeks yang tidak digunakan atau berlebihan menambahkan overhead yang tidak perlu untuk operasi ini.
- Memori dan Tekanan CPU: Indeks bersaing untuk memori kumpulan buffer dan dapat mengusir data yang sering diakses jika set kerja tidak sesuai dengan memori instance yang mengakibatkan

suboptimal. BufferCacheHitRatio Sistem ini juga mengkonsumsi sumber daya CPU yang mempertahankan struktur indeks.

Alat yang tersedia untuk mengidentifikasi peluang pengoptimalan indeks

- Alat Tinjauan Indeks:
- Alat Deteksi Kardinalitas Indeks

Mengumpulkan toolkit:

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

1. Alat Tinjauan Indeks

Alat peninjauan indeks menganalisis semua koleksi dan indeks untuk menyediakan daftar indeks, pola penggunaannya, dan mengidentifikasi indeks redundan di klaster Amazon DocumentDB Anda. Jalankan alat ini pada semua instance cluster untuk analisis komprehensif. Untuk informasi selengkapnya, lihat Alat Tinjauan Indeks.

Pemakaian:

```
cd performance/index-review/

python3 index-review.py --server-alias <server-alias> --uri <mongodb-uri>
```

Berkas Keluaran

- `{server-alias}-collections.csv`: Collection-level statistik
- `{server-alias}-indexes.csv`: Metrik penggunaan indeks terperinci
- `{server-alias}-{timestamp}-index-review.json`: Data mentah untuk analisis lebih lanjut

> **Note**
>
> Jalankan untuk setiap instance di cluster dengan alias independen untuk menghasilkan statistik dari setiap instance, lalu jalankan alat untuk memberi Anda analisis kumulatif

Misalnya: `python3 index-review.py --files "{server-alias}-{timestamp}-index-review.json, {server-alias}-{timestamp}-index-review.json" --server-alias full-review`

2. Alat Deteksi Kardinalitas Indeks

Alat ini mengidentifikasi indeks dengan kardinalitas rendah yang tidak efisien untuk kinerja kueri.

Penggunaan

```
cd performance/index-cardinality-detection/

python3 detect-cardinality.py --uri <mongodb-uri>
```

Parameter

- `--threshold`: Persentase ambang kardinalitas (default: 1%)
- `--sample-count`: Jumlah dokumen untuk sampel (default: 100.000)
- `--max-collections`: Koleksi maksimum untuk dipindai per database (default: 100)

Untuk informasi selengkapnya, lihat https://aws.amazon.com/blogs/database/detect-and-fix-low-cardinality-indexes-in-amazon-documentdb/

Strategi Remediasi

1. Penghapusan Indeks yang tidak terpakai dan berlebihan

```
// Example: Drop unused index
db.collection.dropIndex("unused_index_name")
```

> **Note**
>
> Indeks tidak boleh dijatuhkan tanpa berdiskusi dengan semua pihak yang berkepentingan dan menguji kinerja.

2. Optimalisasi Indeks Kardinalitas Rendah

- Gunakan indeks sebagian dengan filter
- Ubah indeks kardinalitas rendah tunggal menjadi indeks majemuk

Multi-Key Indeks dengan Array Besar

Ikhtisar

Multi-key indeks di Amazon DocumentDB memungkinkan Anda melakukan kueri bidang array secara efisien. Strategi pengindeksan ini memberikan latensi kueri yang jauh lebih rendah saat mencari dalam data array, menjadikannya pilihan yang menarik untuk aplikasi dengan struktur data yang kompleks.

Namun, ketika bekerja dengan array besar atau beberapa bidang array dalam koleksi, penting untuk memahami implikasi sumber daya dan karakteristik kinerja untuk membuat keputusan arsitektur yang tepat. Saat Anda membuat indeks pada bidang array, Amazon DocumentDB menghasilkan entri indeks individual untuk setiap elemen dalam array.

Dampak pada cluster

- Penyimpanan dan IO Overhead: Multi-key indeks terkadang dapat mengkonsumsi ruang penyimpanan hingga beberapa kali dari tabel dasar, ukurannya berbanding lurus dengan dokumen dalam koleksi dengan atribut array yang diindeks, jumlah elemen dalam array dan ukuran elemen.
- Penggunaan Memori: Karena jejak penyimpanan relatif besar, indeks ini memiliki jejak yang sesuai dalam memori, menghasilkan set kerja yang lebih besar.
- Operasi tulis: Multi-key indeks membuat overhead selama operasi penulisan. Setiap elemen array menghasilkan entri indeks terpisah, mengalikan pekerjaan yang diperlukan untuk menyelesaikan operasi penulisan.

Strategi Remediasi

- Buat indeks multi-kunci jika diperlukan dan batasi jumlah bidang dalam array.
- Batasi jumlah indeks multi kunci pada koleksi
- Pertimbangkan untuk memodifikasi model data Anda

Tips Peningkatan Kinerja

Bagian ini menyediakan lima strategi pengoptimalan kinerja untuk Amazon DocumentDB untuk meningkatkan efisiensi aplikasi dan eksekusi kueri.

1. Gunakan $match sebagai Tahap Pertama di Agregation Pipelines

Selalu tempatkan $match sebagai tahap pertama untuk memfilter di pipeline agregasi Anda untuk memaksimalkan kinerja. Amazon DocumentDB akan menggunakan indeks secara efektif saat $match memimpin pipeline, memungkinkan database untuk memfilter data lebih awal dan mengurangi overhead pemrosesan.

```
// Optimized approach
db.orders.aggregate([
  { $match: { status: "active", category: "electronics" } }, // Index utilization
  { $group: { _id: "$category", total: { $sum: "$price" } } },
  { $sort: { total: -1 } }
])
```

Dampak: Pemfilteran awal mengurangi jumlah dokumen yang diproses pada tahap pipeline berikutnya, menghasilkan eksekusi kueri yang lebih cepat dan konsumsi sumber daya yang lebih rendah.

2. Gunakan $project di Agregation Pipeline untuk Meminimalkan Ukuran Data Pipeline

Bawa hanya bidang penting melalui tahapan alur agregasi Anda untuk meminimalkan ukuran data dan meningkatkan kinerja. Gunakan $project secara strategis untuk memasukkan hanya data yang Anda butuhkan.

```
// 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" } } }
])
```

Dampak: Dokumen yang lebih kecil mengurangi penggunaan memori dan meningkatkan efisiensi pemrosesan pipa, menghasilkan peningkatan kinerja kueri secara keseluruhan.

3. Aktifkan Kompresi Dokumen untuk Biaya Penyimpanan yang Lebih Rendah, I/O Biaya, dan Peningkatan Kinerja Kueri

Aktifkan kompresi dokumen dari grup parameter cluster untuk menurunkan biaya penyimpanan, I/O biaya, dan meningkatkan kinerja kueri. Amazon DocumentDB menyimpan dokumen terkompresi pada disk serta RAM, mengurangi jejak memori dan biaya. I/O

Dampak:

- Lebih banyak dokumen sesuai dengan memori yang tersedia
- Akses data lebih cepat dengan pembacaan disk yang dikurangi
- Biaya penyimpanan yang lebih rendah, I/O biaya, dan peningkatan kinerja kueri

> **Note**
>
> Amazon DocumentDB tidak mengaktifkan kompresi secara default untuk versi 5.0. Anda dapat mengaktifkan kompresi pada koleksi atau tingkat cluster untuk 5.0 cluster. Gunakan utilitas tinjauan kompresi Amazon DocumentDB untuk menganalisis rasio kompresi untuk koleksi Anda.
> Untuk Amazon DocumentDB 8.0, kompresi diaktifkan secara default.

4. Indeks Leverage untuk Kinerja Kueri Optimal

Pastikan kueri Anda selalu menggunakan indeks untuk kinerja yang optimal. Amazon DocumentDB menawarkan beberapa jenis indeks untuk mencocokkan kasus penggunaan yang berbeda.

Prinsip pengindeksan:

- Setiap kueri harus memanfaatkan indeks yang sesuai
- Amazon DocumentDB menyediakan beberapa jenis indeks
- Indeks majemuk menawarkan fleksibilitas paling besar dengan mendukung berbagai bentuk kueri dengan satu indeks
- Merancang indeks untuk mendukung operasi penyortiran dan penyaringan bersama

Memahami Awalan Indeks: Indeks gabungan bekerja melalui awalan indeks - Amazon DocumentDB dapat menggunakan subset kiri-ke-kanan bidang indeks. Misalnya, indeks `{ category: 1, price: -1, inStock: 1 }` membuat awalan yang dapat digunakan ini:

- `{ category: 1 }`- Mendukung query penyaringan berdasarkan kategori saja
- `{ category: 1, price: -1 }`- Mendukung pemfilteran kueri berdasarkan kategori dan harga sorting/filtering
- `{ category: 1, price: -1, inStock: 1 }`- Mendukung kueri senyawa lengkap

Kueri pada harga, InStock, atau InStock saja tidak akan menggunakan indeks ini karena tidak dimulai dengan bidang pertama (kategori).

Cara Mengidentifikasi Kueri yang Tidak Menggunakan Indeks: Gunakan metode explain () untuk menganalisis eksekusi kueri dan mengidentifikasi kueri yang melakukan pemindaian koleksi alih-alih menggunakan indeks.

Dampak: Kueri tanpa pemanfaatan indeks menghasilkan pemindaian koleksi, menyebabkan peningkatan memori dan tekanan CPU pada instance dan latensi kueri yang meningkat.

5. Optimalkan Model Data Berdasarkan Pola Kueri

Sejajarkan model data Anda dengan cara kueri aplikasi Anda dan memperbarui data. Pemodelan data adalah dasar dari aplikasi Amazon DocumentDB berkinerja tinggi.

Strategi optimasi:

Penyematan untuk Kinerja

- Menyimpan data terkait bersama-sama ketika sering diakses sebagai satu unit
- Sematkan dokumen yang selalu diambil bersama

- Cocokkan untuk hubungan satu-ke-sedikit

```
// 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" }
  ]
}
```

Referensi untuk Fleksibilitas

- Gunakan referensi untuk data besar atau jarang diakses
- Direkomendasikan untuk hubungan satu-ke-banyak dengan kumpulan data besar
- Mencegah pembongkahan dokumen dan meningkatkan kinerja pembaruan

Strategi Pemisahan Koleksi

Ketika hanya beberapa bidang dalam dokumen besar yang sering diperbarui, atau ketika data besar yang jarang diakses membengkak dokumen, pertimbangkan untuk membagi koleksi:

- Simpan bidang yang sering diperbarui dalam koleksi terpisah yang lebih kecil
- Menyimpan data statis atau jarang diakses di koleksi lain
- Tautkan mereka dengan referensi saat dibutuhkan

```
// 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 */]
}
```

Perolehan kinerja: Dokumen yang lebih kecil berarti pembaruan yang lebih cepat, penggunaan memori berkurang, dan efisiensi cache yang lebih baik.

Dampak: Pemodelan data yang tidak efisien menghasilkan kueri suboptimal, peningkatan ukuran dokumen, dan peningkatan penggunaan memori, yang menyebabkan kinerja aplikasi menurun dan biaya operasional yang lebih tinggi.

Analisis Rencana Kueri

Analisis rencana kueri melalui rencana penjelasan memberikan wawasan penting tentang kinerja kueri Amazon DocumentDB. Paket kueri dengan ExecutionStats mengungkapkan metrik utama termasuk:

- Dokumen dikembalikan per tahap (nReturns)
- Stage-specific waktu eksekusi (eksekusiTimeMillisEstimate)
- Durasi pembuatan rencana (perencanaanTimeMillis)

Dengan memeriksa output rencana kueri, pengembang dapat menganalisis pola eksekusi, mengevaluasi pemanfaatan indeks, dan mengidentifikasi peluang optimasi potensial di seluruh tahapan pipa kueri.

Untuk menganalisis rencana kueri, Anda dapat menggunakan perintah explain () dalam format berikut.

```
db.runCommand({explain: {query document}, verbosity: "executionStats"})
db.collection.find().explain("executionStats");
```

Berikut ini adalah contoh operasi:

```
db.collection.find({ companyname: { '$eq': 'ANYCOMPANY' }, isDeleted: { '$eq':
 false } }).sort({"createdAt":1}).limit(2).explain("executionStats");
```

Keluaran dari operasi ini terlihat seperti berikut ini:

```
{
  queryPlanner: {
    plannerVersion: 2,
    namespace: 'limit_test.test',
    winningPlan: {
      stage: 'LIMIT_SKIP',
      inputStage: {
        stage: 'SORT',
        sortPattern: { createdAt: 1 },
        inputStage: {
          stage: 'IXSCAN',
          indexName: 'companyname_1_createdAt_1_isDeleted_1',
          direction: 'forward',
          indexCond: {
            '$and': [
```

```
              { companyname: { '$eq': 'ANYCOMPANY' } },
              { isDeleted: { '$eq': false } }
            ]
          }
        }
      }
    }
  },
  indexFilterSet: false,
  indexFilterApplied: false,
  executionStats: {
    executionSuccess: true,
    executionTimeMillis: '4.186',
    planningTimeMillis: '3.909',
    executionStages: {
      stage: 'LIMIT_SKIP',
      nReturned: '2',
      executionTimeMillisEstimate: '0.199',
      inputStage: {
        stage: 'SORT',
        nReturned: '2',
        executionTimeMillisEstimate: '0.197',
        sortPattern: { createdAt: 1 },
        inputStage: {
          stage: 'IXSCAN',
          nReturned: '34',
          executionTimeMillisEstimate: '0.151',
          indexName: 'companyname_1_createdAt_1_isDeleted_1',
          direction: 'forward',
          indexCond: {
            '$and': [
              { companyname: { '$eq': 'ANYCOMPANY' } },
              { isDeleted: { '$eq': false } }
            ]
          }
        }
      }
    }
  },
  serverInfo: { host: 'demo-cluster', port: 27017, version: '5.0.0' },
  ok: 1,
  operationTime: Timestamp({ t: 1759915116, i: 1 })
}
```

Di bawah ini adalah analisis terperinci dari rencana eksekusi kueri Amazon DocumentDB, yang memecah setiap komponen dan karakteristik kinerjanya.

Waktu Keseluruhan

eksekusi TimeMillis mewakili total waktu yang diambil oleh query termasuk waktu perencanaan.

perencanaan TimeMillis mewakili total waktu perencanaan yang diambil oleh kueri.

Tahapan Eksekusi

Ini menjelaskan proses langkah demi langkah yang digunakan Amazon DocumentDB untuk mengeksekusi kueri, menunjukkan bagaimana data mengalir melalui operasi yang berbeda.

```
"executionStages": {
    "stage": "[STAGE_NAME]",
    "nReturned": "[NUMBER_OF_DOCS]",
    "executionTimeMillisEstimate": "[TIME]",
    "inputStage": {
        // Nested stages
    }
}
```

Tahapan Umum dalam Rencana Kueri

Di bawah ini adalah tahapan eksekusi umum dalam rencana kueri. Setiap tahap mengembalikan metrik eksekusi TimeMillisEstimate (waktu eksekusi) dan nReturn (jumlah dokumen) untuk membantu mengevaluasi kinerja kueri di setiap tahap.

COLLSCAN (Pemindaian Koleksi)

- Memindai seluruh dokumen koleksi berdasarkan dokumen
- nReturned: Mengembalikan semua dokumen yang cocok dalam koleksi
- Bisa menjadi operasi yang mahal, terlihat ketika tidak ada indeks yang tersedia

IXSCAN (Pemindaian Indeks)

- Menggunakan indeks untuk menemukan dokumen yang cocok
- NReturns: Hanya dokumen yang cocok berdasarkan indeks
- Operasi yang efisien, lebih disukai daripada COLLSCAN

SORT

- Mengurutkan dokumen berdasarkan bidang tertentu
- Jika atribut sort dalam query bukan bagian dari indeks, tahap ini akan muncul secara eksplisit
- nReturns: Nomor yang sama dengan dokumen masukan
- Memory-intensive untuk set hasil besar. Untuk mengoptimalkan, tambahkan nama bidang sortir ke indeks

LIMIT_LEWATI

- Mengontrol jumlah dokumen yang dikembalikan dan dilewati
- nReturns: Dibatasi oleh nilai LIMIT
- Digunakan untuk pagination atau operasi LIMIT

SUBSCAN

- Melakukan operasi kueri bersarang
- nReturned: Bervariasi berdasarkan hasil subquery
- Digunakan dalam kueri kompleks dengan beberapa tahap

Tahapan dengan eksekusi tinggi TimeMillisEstimate adalah kandidat yang baik untuk optimasi.

> **Note**
>
> Parameter ExecutionStats saat ini tidak mendukung perintah update dan delete.

Memahami DocsExamined dalam Menjelaskan Statistik Eksekusi Rencana

Saat Anda menjalankan kueri`explain("executionStats")`, Amazon DocumentDB menyediakan metrik pemeriksaan yang membantu Anda memahami berapa banyak dokumen yang dipindai untuk menghasilkan hasil kueri. Metrik ini berguna untuk mengidentifikasi rencana kueri yang tidak efisien dan mengoptimalkan kinerja.

> **Note**
>
> Hanya tersedia di Amazon DocumentDB 8.0.0+.

Bidang

Bidang	Deskripsi	Tingkat
total DocsExamined	Jumlah total dokumen yang diperiksa di semua tahap eksekusi.	Top-level Execution Stats
DocsExamined	Jumlah dokumen diperiksa oleh tahap eksekusi tertentu.	Stage-level

`docsExamined`Bidang muncul pada tahapan berikut:

Stage	Deskripsi
COLLSCAN	Pemindaian koleksi. Semua dokumen dalam koleksi diperiksa.
IXSCAN	Pemindaian indeks. Hanya dokumen yang cocok dengan kondisi indeks yang diperiksa.
AMBIL	Ketika pengoptimal mengambil dokumen dalam tahap terpisah dari IXSCAN, tahap FETCH melaporkan DocsExamined. Dalam rencana pemindaian indeks, tahapan IXSCAN anak tidak melaporkan DocsExamined. Dalam rencana $lookup, tahap FETCH dan tahapan turunannya dapat melaporkan DocsExamined.

> **Note**
>
> `docsExamined`tidak muncul pada tahap IXONLYSCAN karena kueri dipenuhi sepenuhnya dari indeks tanpa mengakses dokumen.

Tampilkan DocsExamined di Expline Plan ExecutionStats Output

Contoh berikut menggunakan koleksi dengan 500.000 dokumen dan menunjukkan bagaimana `docsExamined` perubahan di berbagai tahap eksekusi kueri.

Buat data sampel:

```
for (let i = 0; i < 500000; i++) {
    db.coll.insertOne({ number: i, arr: [i, [i+1]], value: "test", bool: i % 2 === 
 0 });
}
```

Pemindaian Koleksi (COLLSCAN)

Tanpa indeks, Amazon DocumentDB melakukan pemindaian koleksi lengkap.

```
db.coll.find({ "number": { "$lt": 500 } }).explain("executionStats").executionStats
```

Output:

```
{
    "executionSuccess" : true,
    "nReturned" : "500",
    "executionTimeMillis" : "282.055",
    "planningTimeMillis" : "0.085",
    "totalDocsExamined" : "500000",
    "executionStages" : {
        "stage" : "COLLSCAN",
        "nReturned" : "500",
        "executionTimeMillisEstimate" : "281.915",
```

```
        "docsExamined" : "500000",
        "filter" : {
            "number" : {
                "$lt" : 500
            }
        }
    }
}
```

Semua 500.000 dokumen diperiksa untuk mengembalikan 500 hasil. Membuat indeks pada bidang angka meningkatkan ini.

Pemindaian Indeks (IXSCAN)

```
db.coll.createIndex({ number: 1 });
db.coll.find({ "number": { "$lt": 5000 } }).explain("executionStats").executionStats
```

Output:

```
{
    "executionSuccess" : true,
    "nReturned" : "5000",
    "executionTimeMillis" : "3.047",
    "planningTimeMillis" : "0.296",
    "totalDocsExamined" : "5000",
    "executionStages" : {
        "stage" : "IXSCAN",
        "nReturned" : "5000",
        "executionTimeMillisEstimate" : "2.576",
        "indexName" : "number_1",
        "direction" : "forward",
        "docsExamined" : "5000",
        "indexCond" : {
            "$and" : [
                {
                    "number" : {
                        "$lt" : 5000
                    }
                }
```

```
            ]
        }
    }
}
```

Dengan indeks tersebut, hanya 5.000 dari 500.000 dokumen yang diperiksa dan diambil, sesuai dengan jumlah yang dikembalikan. Kondisi indeks menyaring 495.000 dokumen yang tidak cocok dengan kueri.

Pemindaian Indeks dengan Filter Sisa

```
db.coll.find({ "number": { "$lt": 5000 }, "arr": { "$gt":
 4000 } }).explain("executionStats").executionStats
```

Output:

```
{
    "executionSuccess" : true,
    "nReturned" : "999",
    "executionTimeMillis" : "15.367",
    "planningTimeMillis" : "0.115",
    "totalDocsExamined" : "5000",
    "executionStages" : {
        "stage" : "IXSCAN",
        "nReturned" : "999",
        "executionTimeMillisEstimate" : "15.170",
        "indexName" : "number_1",
        "direction" : "forward",
        "docsExamined" : "5000",
        "indexCond" : {
            "$and" : [
                {
                    "number" : {
                        "$lt" : 5000
                    }
                }
            ]
        },
```

```
        "filter" : {
            "arr" : {
                "$gt" : 4000
            }
        }
    }
}
```

Kondisi indeks cocok dengan 5.000 dokumen dari 500.000, kemudian filter sisa pada `arr` mengurangi hasilnya menjadi 999. `docsExamined`Nilai 5.000 mencerminkan semua dokumen yang diperiksa setelah kondisi indeks tetapi sebelum filter sisa diterapkan.

Ambil dengan IXSCAN

```
db.coll.find({ "$or": [{ "number": { "$lt": 100000 } }, { "number": { "$gt":
 400000 } }] }).explain("executionStats").executionStats
```

Output:

```
{
    "executionSuccess" : true,
    "nReturned" : "199999",
    "executionTimeMillis" : "899.801",
    "planningTimeMillis" : "0.183",
    "totalDocsExamined" : "199999",
    "executionStages" : {
        "stage" : "FETCH",
        "nReturned" : "199999",
        "executionTimeMillisEstimate" : "894.141",
        "docsExamined" : "199999",
        "inputStage" : {
            "stage" : "IXOR",
            "nReturned" : "0",
            "executionTimeMillisEstimate" : "874.897",
            "inputStages" : [
                {
                    "stage" : "IXSCAN",
                    "nReturned" : "100000",
```

```
                    "executionTimeMillisEstimate" : "462.208",
                    "indexName" : "number_1",
                    "indexCond" : {
                        "$and" : [
                            {
                                "number" : {
                                    "$lt" : 100000
                                }
                            }
                        ]
                    }
                },
                {
                    "stage" : "IXSCAN",
                    "nReturned" : "99999",
                    "executionTimeMillisEstimate" : "412.684",
                    "indexName" : "number_1",
                    "indexCond" : {
                        "$and" : [
                            {
                                "number" : {
                                    "$gt" : 400000
                                }
                            }
                        ]
                    }
                }
            ]
        }
    }
}
```

Saat pengoptimal Amazon DocumentDB menggunakan tahap pengambilan untuk mengambil dokumen, tahap FETCH akan melaporkan. `docsExamined` Tahapan IXSCAN anak tidak melaporkan `docsExamined` karena mereka hanya memindai kunci indeks tanpa mengakses dokumen secara langsung.

FETCH dengan Pencarian Agregat

```
db.coll.explain("executionStats").aggregate([
    { $match: { "number": { "$lt": 5 } } },
```

```
    { $lookup: { from: "coll", pipeline: [{ $match: { "number": { "$lt": 3 } } }], as:
 "sub" } }
]).executionStats
```

Output:

```
{
    "executionSuccess" : true,
    "nReturned" : "5",
    "executionTimeMillis" : "0.525",
    "planningTimeMillis" : "0.327",
    "totalDocsExamined" : "9",
    "executionStages" : {
        "stage" : "NESTED_LOOP_LOOKUP",
        "nReturned" : "5",
        "executionTimeMillisEstimate" : "0.163",
        "inputStages" : [
            {
                "stage" : "IXSCAN",
                "nReturned" : "5",
                "executionTimeMillisEstimate" : "0.039",
                "indexName" : "number_1",
                "direction" : "forward",
                "docsExamined" : "5",
                "indexCond" : {
                    "$and" : [
                        {
                            "number" : {
                                "$lt" : 5
                            }
                        }
                    ]
                }
            },
            {
                "stage" : "FETCH",
                "nReturned" : "1",
                "executionTimeMillisEstimate" : "0.009",
                "docsExamined" : "1",
                "inputStage" : {
                    "stage" : "AGGREGATE",
```

```
                    "nReturned" : "1",
                    "executionTimeMillisEstimate" : "0.044",
                    "inputStage" : {
                        "stage" : "IXSCAN",
                        "nReturned" : "3",
                        "executionTimeMillisEstimate" : "0.013",
                        "indexName" : "number_1",
                        "direction" : "forward",
                        "docsExamined" : "3",
                        "indexCond" : {
                            "$and" : [
                                {
                                    "number" : {
                                        "$lt" : 3
                                    }
                                }
                            ]
                        }
                    }
                }
            }
        ]
    }
}
```

IXSCAN luar memeriksa 5 dokumen yang cocok dengan nomor < 5. IXSCAN bagian dalam memeriksa 3 dokumen yang cocok dengan nomor < 3, dan tahap FETCH memeriksa 1 hasil agregat. `totalDocsExamined`Dari 9 adalah jumlah di semua tahap (5 + 3 + 1).

Pemecahan Masalah Amazon DocumentDB

Bagian berikut ini menyediakan informasi tentang cara untuk memecahkan masalah yang mungkin Anda alami ketika menggunakan Amazon DocumentDB (dengan kompatibilitas MongoDB).

Topik

- Memecahkan masalah konektivitas
- Pemecahan masalah indeks
- Pemecahan masalah kinerja dan pemanfaatan sumber daya
- Pengumpulan sampah di Amazon DocumentDB

Memecahkan masalah konektivitas

Mengalami masalah saat menghubungkan? Berikut ini adalah beberapa skenario umum dan cara untuk mengatasinya.

Topik

- Tidak dapat terhubung ke titik akhir Amazon DocumentDB
- Menguji koneksi ke instans Amazon DocumentDB
- Menghubungkan ke titik akhir yang tidak valid
- Konfigurasi driver memengaruhi jumlah koneksi

Tidak dapat terhubung ke titik akhir Amazon DocumentDB

Ketika Anda mencoba untuk menyambungkan ke Amazon DocumentDB, berikut ini adalah salah satu pesan kesalahan yang paling umum yang mungkin Anda terima.

```
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
```

Arti pesan kesalahan ini biasanya adalah bahwa klien Anda (mongo shell dalam contoh ini) tidak dapat mengakses titik akhir Amazon DocumentDB. Hal ini dapat terjadi karena beberapa alasan:

Topik

- Menghubungkan dari titik akhir publik
- Koneksi lintas wilayah
- Menghubungkan dari berbagai VPC Amazon
- Grup keamanan memblokir koneksi masuk
- Driver Java Mongo membaca masalah preferensi

Menghubungkan dari titik akhir publik

Anda mencoba terhubung ke cluster Amazon DocumentDB langsung dari laptop atau mesin pengembangan lokal Anda.

Mencoba untuk menghubungkan ke klaster Amazon DocumentDB langsung dari titik akhir publik, seperti laptop Anda atau mesin pengembangan lokal, akan gagal. Amazon DocumentDB adalah virtual private cloud (VPC)-satu-satunya dan saat ini tidak mendukung titik akhir publik. Dengan demikian, Anda tidak dapat menghubugkan secara langsung ke klaster Amazon DocumentDB Anda dari laptop Anda atau lingkungan pengembangan lokal di luar VPC Anda.

Untuk menghubungkan ke klaster Amazon DocumentDB dari luar Amazon VPC, Anda dapat menggunakan terowongan SSH. Untuk informasi selengkapnya, lihat Menghubungkan ke cluster Amazon DocumentDB dari luar Amazon VPC. Selain itu, jika lingkungan pengembangan Anda adalah dalam Amazon VPC yang berbeda, Anda juga dapat menggunakan Peering VPC dan menghubungkan ke klaster Amazon DocumentDB Anda dari Amazon VPC lainnya di wilayah yang sama atau wilayah yang berbeda.

Koneksi lintas wilayah

Anda mencoba terhubung ke cluster Amazon DocumentDB di wilayah lain.

Jika Anda mencoba untuk menghubungkan ke klaster Amazon DocumentDB dari instans Amazon EC2 di Wilayah selain Wilayah klaster ini—sebagai contoh, mencoba untuk menghubungkan ke

klaster di Wilayah US East (N. Virginia) (us-east-1) dari Wilayah US West (Oregon) (us-west-2)—koneksi akan gagal.

Untuk memverifikasi Wilayah klaster Amazon DocumentDB Anda, jalankan perintah berikut ini. Wilayah ini berada di titik akhir.

```
aws docdb describe-db-clusters \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].Endpoint'
```

Keluaran dari operasi ini terlihat seperti berikut ini.

```
[
    "sample-cluster.node.us-east-1.docdb.amazonaws.com"
]
```

Untuk memverifikasi Wilayah instans EC2 Anda, jalankan perintah berikut ini.

```
 aws ec2 describe-instances \
     --query 'Reservations[*].Instances[*].Placement.AvailabilityZone'
```

Keluaran dari operasi ini terlihat seperti berikut ini.

```
[
    [
        "us-east-1a"
    ]
]
```

Menghubungkan dari berbagai VPC Amazon

Anda mencoba terhubung ke cluster Amazon DocumentDB dari VPC yang berbeda dari VPC Amazon yang digunakan cluster Anda.

Jika cluster Amazon DocumentDB dan instans Amazon EC2 Anda sama, tetapi tidak di VPC Amazon Wilayah AWS yang sama, Anda tidak dapat terhubung langsung ke cluster Amazon DocumentDB Anda kecuali VPC Peering diaktifkan di antara dua VPC Amazon.

Untuk memverifikasi Amazon VPC dari instans Amazon DocumentDB Anda, jalankan perintah berikut ini.

```
aws docdb describe-db-instances \
   --db-instance-identifier sample-instance \
   --query 'DBInstances[*].DBSubnetGroup.VpcId'
```

Untuk memverifikasi Amazon VPC dari instans Amazon EC2 Anda, jalankan perintah berikut ini.

```
aws ec2 describe-instances \
   --query 'Reservations[*].Instances[*].VpcId'
```

Grup keamanan memblokir koneksi masuk

Anda mencoba terhubung ke cluster Amazon DocumentDB, dan grup keamanan klaster tidak mengizinkan koneksi masuk pada port cluster (port default: 27017).

Anggap bahwa klaster Amazon DocumentDB Anda dan instans Amazon EC2 keduanya di Wilayah dan Amazon VPC yang sama dan menggunakan grup keamanan Amazon VPC yang sama. Jika Anda tidak dapat menghubungkan ke klaster Amazon DocumentDB Anda, kemungkinan penyebabnya adalah bahwa grup keamanan Anda (yaitu, firewall) untuk klaster Anda tidak mengizinkan koneksi inbound pada port yang Anda pilih untuk klaster Amazon DocumentDB Anda (port default adalah 27017).

Untuk memverifikasi port untuk klaster Amazon DocumentDB Anda , jalankan perintah berikut ini.

```
aws docdb describe-db-clusters \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].[DBClusterIdentifier,Port]'
```

Untuk mendapatkan grup keamanan Amazon DocumentDB Anda untuk klaster Anda, jalankan perintah berikut ini.

```
aws docdb describe-db-clusters \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].[VpcSecurityGroups[*],VpcSecurityGroupId]'
```

Untuk memeriksa aturan inbound untuk grup keamanan Anda, lihat topik berikut ini dalam dokumentasi Amazon EC2:

- Otorisasi Lalu Lintas Masuk untuk Instans Linux Anda

- Mengotorisasi Lalu Lintas Masuk untuk Instans Windows Anda

Driver Java Mongo membaca masalah preferensi

Preferensi baca klien tidak dihormati dan beberapa klien tidak dapat menulis ke Amazon DocumentDB setelah failover kecuali mereka reboot.

Masalah ini, pertama kali ditemukan di Java Mongo Driver 3.7.x, terjadi ketika klien membuat koneksi ke Amazon DocumentDB menggunakan `MongoClientSettings` dan, khususnya, saat merantai metode. `applyToClusterSettings` Pengaturan MongoClient Cluster dapat didefinisikan menggunakan beberapa metode yang berbeda, sepertihosts(),requiredReplicaSetName(), danmode().

Ketika klien hanya menentukan satu host dalam `hosts()` metode, mode diatur ke `ClusterConnectionMode.SINGLE` bukan `ClusterConnectionMode.MULTIPLE` Ini menyebabkan klien mengabaikan preferensi baca dan hanya terhubung ke server yang dikonfigurasi. `hosts()` Jadi, bahkan jika pengaturan klien diinisialisasi seperti di bawah ini, semua pembacaan akan tetap pergi ke primer, bukan yang sekunder.

```
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);
```

Kasus failover

Menggunakan pengaturan koneksi klien di atas, jika ada failover dan pembaruan catatan DNS yang tertunda untuk titik akhir penulis cluster, klien masih akan mencoba mengeluarkan penulisan ke penulis lama (sekarang pembaca setelah failover). Ini menghasilkan kesalahan sisi server (bukan

primer) yang tidak ditangani dengan tepat oleh driver Java (ini masih dalam penyelidikan). Dengan demikian, klien dapat dibiarkan dalam keadaan buruk sampai server aplikasi di-boot ulang, misalnya.

Ada dua solusi untuk ini:

- Klien yang terhubung ke Amazon DocumentDB melalui string koneksi tidak akan mengalami masalah ini, `ClusterConnectionMode` karena akan disetel `MULTIPLE` ke saat menyetel preferensi baca.

  ```
  MongoClientURI mongoClientURI = new MongoClientURI("mongodb://usr:pass:cluster-
  endpoint:27317/test?ssl=false&replicaSet=rs0&readpreference=secondaryPreferred");
  MongoClient mongoClient = MongoClients.create(mongoClientURI.getURI());
  ```

  Atau menggunakan `MongoClientSettings` pembangun dengan `applyConnectionString` metode ini.

  ```
  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);
  ```

- Secara eksplisit diatur `ClusterConnectionMode` ke. `MULTIPLE` Ini hanya diperlukan saat menggunakan `applyToClusterSettings` dan`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);
```

Menguji koneksi ke instans Amazon DocumentDB

Anda dapat menguji koneksi ke klaster menggunakan alat Linux atau Windows umum.

Dari terminal Linux atau Unix, Anda dapat menguji koneksi dengan memasukkan hal berikut ini (ganti `cluster-endpoint` dengan titik akhir, dan ganti `port` dengan port instans Anda):

```
nc -zv cluster-endpoint port
```

Berikut ini adalah contoh operasi sampel dan nilai balik:

```
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!
```

Menghubungkan ke titik akhir yang tidak valid

Ketika menghubungkan ke klaster Amazon DocumentDB dan Anda menggunakan titik akhir klaster yang tidak valid, muncul kesalahan yang mirip dengan berikut ini.

```
mongo --ssl \
   --host sample-cluster.node.us-east-1.docdb.amazonaws.com:27017 \
   --sslCAFile global-bundle.pem \
   --username <user-name> \
   --password <password>
```

Keluaran terlihat seperti ini:

```
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
```

Untuk mendapatkan titik akhir yang valid untuk klaster, jalankan perintah berikut ini:

```
aws docdb describe-db-clusters \
   --db-cluster-identifier sample-cluster \
   --query 'DBClusters[*].[Endpoint,Port]'
```

Untuk mendapatkan titik akhir yang valid untuk instans, jalankan perintah berikut ini:

```
aws docdb describe-db-instances \
   --db-instance-identifier sample-instance \
   --query 'DBInstances[*].[Endpoint.Address,Endpoint.Port]'
```

Untuk informasi selengkapnya, lihat Memahami titik akhir Amazon DocumentDB.

Konfigurasi driver memengaruhi jumlah koneksi

Saat menggunakan driver klien untuk terhubung ke cluster Amazon DocumentDB, penting untuk mempertimbangkan `maxPoolSize` parameter konfigurasi. `maxPoolSize`Pengaturan menentukan jumlah maksimum koneksi yang akan dipertahankan oleh driver klien di kumpulan koneksinya.

Pemecahan masalah indeks

Topik berikut ini membahas apa yang harus dilakukan jika indeks atau pembuatan indeks latar belakang gagal.

Topik

- Pembuatan indeks gagal
- Indeks latar belakang membangun masalah latensi dan gagal
- Indeks basis data kembung

Pembuatan indeks gagal

Amazon DocumentDB menggunakan penyimpanan lokal pada instans sebagai bagian dari proses pembuatan indeks. Anda dapat memantau penggunaan disk ini menggunakan FreeLocalStorage

CloudWatch metrik (`CloudWatch -> Metrics -> DocDB -> Instance Metrics`). Ketika pembuatan indeks mengonsumsi semua disk lokal dan gagal, Anda akan menerima kesalahan. Ketika memigrasi data ke Amazon DocumentDB, kami mendorong Anda untuk membuat indeks terlebih dahulu dan kemudian memasukkan data. Untuk informasi selengkapnya tentang strategi migrasi dan membuat indeks, lihat Migrasi ke Amazon DocumentDB dalam dokumentasi Amazon DocumentDB dan blog: Bermigrasi dari MongoDB ke Amazon DocumentDB menggunakan metode offline.

Ketika membuat indeks pada klaster yang ada, jika pembuatan indeks membutuhkan waktu lebih lama dari yang diharapkan atau gagal, kami sarankan Anda menskalakan ke atas instans untuk membuat indeks kemudian, setelah indeks dibuat, menskalakan kembali ke bawah. Amazon DocumentDB memungkinkan Anda menskalakan ukuran instans dengan cepat dalam hitungan menit menggunakan Konsol Manajemen AWS atau file. AWS CLI Untuk informasi selengkapnya, lihat Mengelola kelas instance. Dengan harga instans per detik, Anda hanya membayar untuk sumber daya yang Anda gunakan hingga hitungan detik.

Indeks latar belakang membangun masalah latensi dan gagal

Indeks latar belakang yang dibangun di Amazon DocumentDB tidak mulai hingga semua kueri pada instans primer yang dimulai sebelum pembuatan indeks dijalankan secara lengkap saat dimulai. Jika terdapat kueri yang berjalan lama, pembuatan indeks latar belakang akan memblokir hingga kueri selesai dan dengan demikian dapat membutuhkan waktu lebih lama dari yang diharapkan untuk menyelesaikan. Hal ini berlaku bahkan jika koleksi kosong.

Pembuatan indeks latar depan tidak menunjukkan perilaku pemblokiran yang sama. Sebaliknya, pembuatan indeks latar depan mengambil kunci eksklusif pada koleksi hingga pembuatan indeks selesai. Dengan demikian, untuk membuat indeks pada koleksi kosong dan untuk menghindari pemblokiran pada setiap kueri yang berjalan lama, kami sarankan menggunakan pembuatan indeks latar depan.

> **Note**
>
> Amazon DocumentDB hanya mengizinkan satu pembuatan indeks latar belakang terjadi pada koleksi pada waktu tertentu. Jika operasi DDL (Data Definition Language) seperti `createIndex()` atau `dropIndex()` terjadi pada koleksi yang sama selama pembuatan indeks latar belakang, pembuatan indeks latar belakang akan gagal.

Indeks basis data kembung

Amazon DocumentDB Multi-Version menggunakan Concurrency Control (MVCC) untuk mengelola transaksi bersamaan. Ketika dokumen dihapus atau diperbarui, versi sebelumnya tetap dalam koleksi dan indeks sebagai versi “mati”. Proses pengumpulan sampah secara otomatis merebut kembali ruang dari versi mati ini untuk operasi masa depan.

Indeks kembung terjadi ketika indeks koleksi menjadi lebih besar karena akumulasi entri indeks mati atau usang atau fragmentasi dalam halaman. Persentase yang dilaporkan mewakili jumlah ruang indeks yang dapat digunakan oleh entri indeks future. Bloat ini menghabiskan ruang di cache buffer dan penyimpanan. Jika Anda ingin menghapus kembung, Anda perlu membangun kembali indeks.

Example Contoh

Jalankan perintah berikut untuk menentukan penyimpanan yang tidak digunakan untuk indeks Anda:

```
db.coll.aggregate({$indexStats:{}});
```

Hasilnya terlihat mirip dengan ini:

```
{
    "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
    }
```

```
}
```

Anda dapat membangun kembali indeks tanpa downtime menggunakan `reIndex` perintah, yang memerlukan pemindaian seluruh koleksi. Lihat Pemeliharaan indeks menggunakan `reIndex`.

Pemecahan masalah kinerja dan pemanfaatan sumber daya

Bagian ini menyediakan pertanyaan dan solusi untuk masalah diagnostik umum dalam deployment Amazon DocumentDB. Contoh disediakan menggunakan mongo shell dan dicakupkan ke instans individu. Untuk menemukan titik akhir instans, lihat Memahami titik akhir Amazon DocumentDB.

Topik

- Bagaimana cara menentukan jumlah operasi penyisipan, pembaruan, dan penghapusan yang dilakukan pada koleksi saya melalui Mongo API?
- Bagaimana cara menganalisis kinerja cache?
- Bagaimana cara menemukan dan menghentikan kueri yang berjalan lama atau diblokir?
- Bagaimana saya bisa melihat rencana kueri dan mengoptimalkan kueri?
- Bagaimana saya bisa melihat rencana kueri dalam cluster elastis?
- Bagaimana cara saya mencantumkan semua operasi yang berjalan pada sebuah instance?
- Bagaimana saya tahu kapan kueri membuat kemajuan?
- Bagaimana cara menentukan mengapa sistem tiba-tiba berjalan lambat?
- Bagaimana cara menentukan penyebab pemanfaatan CPU yang tinggi pada satu atau lebih instance cluster?
- Bagaimana cara menentukan kursor terbuka pada sebuah instance?
- Bagaimana cara menentukan versi mesin Amazon DocumentDB saat ini?
- Bagaimana cara menganalisis penggunaan indeks dan mengidentifikasi indeks yang tidak digunakan?
- Bagaimana cara mengidentifikasi indeks yang hilang?
- Bagaimana cara menentukan pengumpulan database bloat?
- Ringkasan pertanyaan yang berguna

Bagaimana cara menentukan jumlah operasi penyisipan, pembaruan, dan penghapusan yang dilakukan pada koleksi saya melalui Mongo API?

Untuk melihat jumlah operasi penyisipan, pembaruan, dan penghapusan yang dilakukan pada koleksi tertentu, jalankan perintah berikut pada koleksi tersebut:

```
db.collection.stats()
```

Output dari perintah ini menjelaskan hal berikut di bawah `opCounters` bidangnya:

- num DocsIns - Jumlah dokumen yang dimasukkan ke dalam koleksi ini. Ini termasuk dokumen yang dimasukkan menggunakan `insertMany` perintah `insert` dan, serta dokumen yang dimasukkan oleh upsert.
- num DocsUpd - Jumlah pembaruan dokumen dalam koleksi ini. Ini termasuk dokumen yang diperbarui menggunakan `findAndModify` perintah `update` dan.
- num DocsDel - Jumlah dokumen yang dihapus dari koleksi ini. Ini termasuk dokumen yang dihapus menggunakan`deleteOne`,`deleteMany`,`remove`, dan `findAndModify` perintah.
- LastReset - Waktu penghitung ini telah diatur ulang terakhir. Statistik yang disediakan oleh perintah ini diatur ulang saat starting/stopping cluster atau penskalaan up/down instance.

Contoh output dari menjalankan `db.collection.stats()` ditunjukkan di bawah ini.

```
{
    "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)
}
```

Perintah statistik ini harus digunakan saat melihat penghitung khusus koleksi untuk menyisipkan, memperbarui, dan menghapus operasi melalui Mongo API. Cara lain untuk melihat penghitung operasi khusus koleksi adalah dengan mengaktifkan audit DHTML. Jumlah operasi penyisipan, perbarui, dan hapus pada semua koleksi selama interval waktu satu menit dapat dilihat diMemantau Amazon DocumentDB dengan CloudWatch.

Bagaimana cara menganalisis kinerja cache?

Menganalisis kinerja cache dapat memberikan wawasan tentang efisiensi pengambilan data dan kinerja sistem, dan didasarkan pada berapa banyak data yang dibaca dari disk versus cache. Kami menyediakan statistik cache tentang jumlah klik cache (data yang dibaca dari cache) dan cache yang salah (data yang tidak ditemukan di cache dan dibaca dari disk) untuk memberikan wawasan tentang kinerja cache. Statistik cache untuk koleksi tertentu dapat ditemukan dengan menjalankan perintah berikut pada koleksi itu:

```
db.collection.stats()
```

Nilai-nilai di `cacheStats` bidang dalam output dari perintah ini menyediakan statistik cache untuk koleksi serta statistik cache total untuk indeks yang dibuat pada koleksi. Statistik ini tercantum di bawah ini:

- **`collBlksHit`**- Jumlah blok yang dibaca dari cache selama operasi pada koleksi ini.
- **`collBlksRead`**- Jumlah blok yang dibaca dari disk (cache meleset) selama operasi pada koleksi ini.

- **collHitRatio**- Rasio hit cache untuk koleksi ini (`100 * [collBlksHit / (collBlksHit + collBlksRead)]`).
- **idxBlksHit**- Jumlah blok yang dibaca dari cache untuk indeks apa pun yang dibuat pada koleksi ini.
- **idxBlksRead**- Jumlah blok yang dibaca dari disk (cache meleset) untuk indeks apa pun yang dibuat pada koleksi ini.
- **idxHitRatio**- Rasio hit cache untuk indeks yang dibuat pada koleksi ini (`100 * [idxBlksHit / (idxBlksHit + idxBlksRead)]`).
- **lastReset**- Waktu statistik ini telah diatur ulang terakhir. Statistik yang disediakan oleh `db.collection.stats()` diatur ulang saat starting/stopping cluster atau penskalaan up/down instance.

Rincian `idxBlksRead` bidang `idxBlksHit` dan untuk setiap indeks juga dapat ditemukan menggunakan `indexStats` perintah. Statistik cache indeks tertentu dapat ditemukan dengan menjalankan perintah berikut:

```
db.collection.aggregate([{$indexStats:{}}]).pretty()
```

Untuk setiap indeks, statistik cache berikut dapat ditemukan di bawah `cacheStats` bidang:

- **blksHit**- Jumlah blok yang dibaca dari cache untuk indeks ini.
- **blksRead**- Jumlah blok yang dibaca dari disk untuk indeks ini.
- **blksHitRatio**- Rasio hit cache dibulatkan ke empat tempat desimal, dihitung dengan. `100 * [blksHit / (blksHit + blksRead)]`

Bagaimana cara menemukan dan menghentikan kueri yang berjalan lama atau diblokir?

Kueri pengguna dapat berjalan lambat karena rencana kueri suboptimal atau dapat diblokir akibat kontensi sumber daya.

Untuk menemukan kueri yang berjalan lama yang melambat akibat rencana kueri suboptimal, atau kueri yang diblokir akibat kontensi sumber daya, gunakan perintah `currentOp`. Anda dapat menyaring perintah untuk membantu mempersempit daftar kueri yang relevan untuk mengakhiri.

Anda harus memiliki `opid` yang berkaitan dengan kueri yang berjalan lama untuk dapat mengakhiri kueri.

Kueri berikut ini menggunakan perintah `currentOp` untuk mendaftar semua kueri yang diblokir maupun yang berjalan selama lebih dari 10 detik.

```
db.adminCommand({
    aggregate: 1,
    pipeline: [
        {$currentOp: {}},
        {$match:
            {$or: [
                {secs_running: {$gt: 10}},
                {WaitState: {$exists: true}}]}},
        {$project: {_id:0, opid: 1, secs_running: 1}}],
    cursor: {}
});
```

Selanjutnya, Anda dapat mempersempit kueri untuk menemukan `opid` dari kueri yang berjalan selama lebih dari 10 detik dan mengakhirinya.

Untuk menemukan dan mengakhiri kueri yang berjalan selama lebih dari 10 detik

1. Temukan `opid` kueri.

   ```
   db.adminCommand({
       aggregate: 1,
       pipeline: [
           {$currentOp: {}},
           {$match:
               {$or:
                   [{secs_running: {$gt: 10}},
                    {WaitState: {$exists: true}}]}}],
       cursor: {}
   });
   ```

   Keluaran dari operasi ini terlihat seperti berikut ini (format JSON).

   ```
   {
       "waitedMS" : NumberLong(0),
       "cursor" : {
           "firstBatch" : [
   ```

```
            {
                "opid" : 24646,
                "secs_running" : 12
            }
        ],
        "id" : NumberLong(0),
        "ns" : "admin.$cmd"
    },
    "ok" : 1
}
```

2. Mengakhiri kueri menggunakan operasi `killOp`.

```
db.adminCommand({killOp: 1, op: 24646});
```

Bagaimana saya bisa melihat rencana kueri dan mengoptimalkan kueri?

Jika kueri berjalan lambat, itu bisa karena eksekusi kueri memerlukan pemindaian penuh koleksi untuk memilih dokumen yang relevan. Kadang-kadang membuat indeks yang sesuai memungkinkan kueri untuk berjalan lebih cepat. Untuk mendeteksi skenario ini dan memutuskan bidang untuk membuat indeks, gunakan perintah `explain`.

> **Note**
>
> Amazon DocumentDB mengemulasi MongoDB 3.6 API pada mesin basis data yang dibuat khusus yang memanfaatkan sistem penyimpanan yang didistribusikan, toleran terhadap kesalahan, dan penyembuhkan diri sendiri. Akibatnya, rencana kueri dan keluaran dari `explain()` mungkin berbeda antara Amazon DocumentDB dan MongoDB. Pelanggan yang ingin kontrol atas rencana kueri mereka dapat menggunakan operator `$hint` untuk memaksa pemilihan indeks yang diutamakan.

Jalankan kueri yang ingin Anda tingkatkan di bawah perintah `explain` sebagai berikut.

```
db.runCommand({explain: {<query document>}})
```

Berikut ini adalah operasi contoh.

```
db.runCommand({explain:{
```

```
    aggregate: "sample-document",
    pipeline: [{$match: {x: {$eq: 1}}}],
    cursor: {batchSize: 1}}
});
```

Keluaran dari operasi ini terlihat seperti berikut ini (format JSON).

```
{
    "queryPlanner" : {
        "plannerVersion" : 1,
        "namespace" : "db.test",
        "winningPlan" : {
            "stage" : "COLLSCAN"
        }
    },
    "serverInfo" : {
        "host" : "...",
        "port" : ...,
        "version" : "..."
    },
    "ok" : 1
}
```

Keluaran sebelumnya menunjukkan bahwa tahap `$match` membutuhkan pemindaian seluruh koleksi dan memeriksa apakah bidang `"x"` di setiap dokumen adalah sama dengan 1. Jika ada banyak dokumen dalam koleksi, pemindaian koleksi (dan karena itu performa kueri keseluruhan) sangat lambat. Dengan demikian adanya `"COLLSCAN"` dalam keluaran dari perintah `explain` menunjukkan bahwa performa kueri dapat ditingkatkan dengan membuat indeks yang sesuai.

Dalam contoh ini, kueri memeriksa apakah bidang `"x"` sama dengan 1 di semua dokumen. Jadi embuat indeks di bidang `"x"` memungkinkan kueri untuk menghindari pemindaian koleksi lengkap dan menggunakan indeks untuk mengembalikan dokumen yang relevan dengan lebih cepat.

Setelah membuat indeks pada bidang `"x"`, keluaran `explain` adalah sebagai berikut.

```
{
    "queryPlanner" : {
         "plannerVersion" : 1,
         "namespace" : "db.test",
         "winningPlan" : {
             "stage" : "IXSCAN",
             "indexName" : "x_1",
```

```
            "direction" : "forward"
        }
    },
    "serverInfo" : {
        "host" : "...",
        "port" : ...,
        "version" : "..."
    },
    "ok" : 1
}
```

Membuat indeks pada bidang `"x"` mengaktifkan tahap `$match` untuk menggunakan pemindaian indeks untuk mengurangi jumlah dokumen diana predikat `"x = 1"` harus dievaluasi.

Untuk koleksi kecil, prosesor kueri Amazon DocumentDB dapat memilih untuk tidak menggunakan indeks jika keuntungan performa diabaikan.

Bagaimana saya bisa melihat rencana kueri dalam cluster elastis?

Untuk memeriksa rencana kueri dalam cluster elastis, gunakan `explain` perintah. Berikut ini adalah contoh `explain` operasi pada kueri find yang menargetkan koleksi sharded:

```
db.runCommand(
   {
     explain: { find: "cities", filter: {"name": "Seoul"}}
   }
)
```

> **Note**
>
> Amazon DocumentDB mengemulasi MongoDB pada mesin database yang dibuat khusus. Akibatnya, rencana kueri dan keluaran dari `explain()` mungkin berbeda antara Amazon DocumentDB dan MongoDB. Anda dapat mengontrol rencana kueri dengan menggunakan `$hint` operator untuk menerapkan pemilihan indeks pilihan.

Output dari operasi ini mungkin terlihat seperti berikut (format 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)
}
```

Output sebelumnya menunjukkan rencana kueri untuk `find` kueri pada cluster tiga-shard. Setiap pecahan memiliki beberapa partisi data yang dapat memiliki tahapan input yang berbeda. Dalam contoh ini, “COLLSCAN “(pemindaian koleksi) dijalankan pada semua partisi sebelum hasilnya digabungkan pada tahap” PARTITION_MERGE “dalam setiap pecahan. Hasil di seluruh pecahan kemudian digabungkan bersama pada tahap” SHARD_MERGE “sebelum dikirim kembali ke klien.

Bagaimana cara saya mencantumkan semua operasi yang berjalan pada sebuah instance?

Sebagai pengguna atau pengguna utama, Anda sering ingin mencantumkan semua operasi saat ini yang berjalan pada sebuah instance untuk tujuan diagnostik dan pemecahan masalah. (Untuk informasi tentang mengelola pengguna, lihat Mengelola pengguna Amazon DocumentDB.)

Dengan `mongo` shell, Anda dapat menggunakan kueri berikut ini untuk mendaftar semua operasi yang berjalan pada instans Amazon DocumentDB.

```
db.adminCommand({currentOp: 1, $all: 1});
```

Kueri mengembalikan daftar lengkap dari semua kueri pengguna dan tugas sistem internal yang saat ini beroperasi pada instans.

Keluaran dari operasi ini terlihat seperti berikut ini (format 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
}
```

Berikuti ini adalah nilai valid untuk bidang `"desc"`:

- **`INTERNAL`** — Tugas sistem internal seperti pembersihan kursor atau tugas pembersihan pengguna usang.
- **`TTLMonitor`** — Urutan monitor Waktu untuk Tayang (TTL). Statusnya yang sedang berjalan tercermin di bidang `"active"`.
- **`GARBAGE_COLLECTION`** — Urutan pengumpul sampah internal.
- **`CONN`** — Kueri pengguna.
- **`CURSOR`** — Operasi adalah kursor diam yang menunggu pengguna untuk memanggil perintah "getMore" untuk mendapatkan hasil batch berikutnya. Dalam keadaan ini, kursor mengonsumsi memori, tetapi tidak mengonsumsi komputasi apa pun.

Keluaran sebelumnya juga mendaftar semua kueri pengguna yang berjalan di sistem. Setiap kueri pengguna berjalan dalam konteks basis data dan koleksi, dan penyatuan keduanya disebut namespace. Namespace dari setiap kueri pengguna tersedia di bidang `"ns"`.

Kadang-kadang Anda perlu mendaftar semua kueri pengguna yang berjalan di namespace tertentu. Sehingga keluaran sebelumnya harus difilter pada bidang `"ns"`. Berikut ini adalah contoh kueri untuk mencapai keluaran untuk filter. Kueri mendaftar semua kueri pengguna yang saat ini berjalan di basis data `"db"` dan koleksi `"test"` (yaitu, namespace `"db.test"`).

```
db.adminCommand({aggregate: 1,
    pipeline: [{$currentOp: {allUsers: true, idleConnections: true}},
               {$match: {ns: {$eq: "db.test"}}}],
    cursor: {}
});
```

Sebagai pengguna utama sistem, Anda dapat melihat kueri semua pengguna dan juga semua tugas sistem internal. Semua pengguna lain hanya dapat melihat kueri mereka masing-masing.

Jika jumlah total kueri dan tugas sistem internal melebihi ukuran kursor batch default , `mongo` shell secara otomatis menghasilkan sebuah objek iterator `'it'` untuk melihat sisa hasil. Tetap jalankan perintah `'it'` hingga semua hasil telah dikeluarkan.

Bagaimana saya tahu kapan kueri membuat kemajuan?

Kueri pengguna dapat berjalan lambat akibat rencana kueri suboptimal, atau mereka dapat diblokir akibat kontensi sumber daya. Debugging kueri semacam itu adalah proses multi-langkah yang mungkin perlu menjalankan langkah yang sama beberapa kali.

Langkah pertama debugging adalah untuk mendaftar semua kueri yang berjalan lama atau diblokir. Kueri berikut iniendaftar semua kueri pengguna yang telah berjalan selama lebih dari 10 detik atau yang menunggu sumber daya.

```
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: {}
                });
```

Ulangi kueri sebelumnya secara berkala untuk menentukan apakah daftar kueri berubah dan untuk mengidentifikasi kueri yang berjalan lama atau diblokir.

Jika dokumen keluaran untuk kueri yang menjadi perhatian memiliki bidang `WaitState`, ini menunjukkan bahwa kontensi sumber daya adalah mengapa kueri berjalan lambat atau diblokir.

Perselisihan sumber daya dapat disebabkan oleh I/O, tugas sistem internal, atau permintaan pengguna lainnya.

Keluaran dari operasi ini terlihat seperti berikut ini (format 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 dapat menjadi hambatan jika banyak kueri di koleksi yang berbeda berjalan secara bersamaan pada instance yang sama, atau jika ukuran instance terlalu kecil untuk kumpulan data tempat kueri berjalan. Jika kueri adalah kueri hanya baca, Anda dapat memitigasi situasi sebelumnya dengan memisahkan kueri untuk setiap koleksi di seluruh replika terpisah. Untuk pembaruan bersamaan di koleksi yang berbeda, atau ketika ukuran instans terlalu kecil untuk set data, Anda dapat memitigasi dengan menskalakan ke atas instans.

Jika kontensi sumber daya adalah akibat kueri pengguna lain, bidang `"blockedOn"` di dokumen keluaran akan memiliki `"opid"` dari kueri yang memengaruhi kueri ini. Dengan menggunakan `"opid"` ikuti rantai bidang `"WaitState"` dan `"blockedOn"` dari semua kueri untuk menemukan kueri pada kepala rantai.

Jika tugas kepala rantai adalah tugas internal, satu-satunya mitigasi dalam kasus ini akan mengakhiri kueri dan nantinya akan menjalankan kembali.

Berikut ini adalah keluaran contoh di mana kueri penemuan diblokir pada kunci koleksi yang dimiliki oleh tugas lain.

```
{
    "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
}
```

Jika `"WaitState"` memiliki nilai `"Latch"`, `"SystemLock"`, `"BufferLock"`, `"BackgroundActivity"`, atau `"Other"`, sumber kontensi sumber daya adalah tugas sistem internal. Jika situasi terus berlangsung selama waktu yang lama, hanya mitigasi yang akan mengakhiri kueri dan nantinya akan menjalankan kembali.

Bagaimana cara menentukan mengapa sistem tiba-tiba berjalan lambat?

Berikut ini adalah beberapa alasan umum mengapa sistem melambat:

- Kontensi sumber daya yang berlebihan di antara kueri yang bersamaan
- Jumlah kueri bersamaan yang aktif meningkat seiring waktu
- Tugas sistem internal seperti `"GARBAGE_COLLECTION"`

Untuk memantau penggunaan sistem dari waktu ke waktu, jalankan kueri `"currentOp"` berikut ini secara berkala dan keluarkan hasil ke penyimpanan eksternal. Kueri menghitung jumlah kueri dan operasi di setiap namespace dalam sistem. Anda kemudian dapat menganalisis hasil penggunaan sistem untuk memahami beban pada sistem dan mengambil tindakan yang tepat.

```
db.adminCommand({aggregate: 1,
                 pipeline: [{$currentOp: {allUsers: true, idleConnections: true}},
                            {$group: {_id: {desc: "$desc", ns: "$ns", WaitState:
 "$WaitState"}, count: {$sum: 1}}}],
                 cursor: {}
                });
```

Kueri ini mengembalikan agregat dari semua kueri yang berjalan di setiap namespace, semua tugas sistem internal, dan jumlah unik keadaan menunggu (jika ada) per namespace.

Keluaran dari operasi ini terlihat seperti berikut ini (format 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
}
```

Dalam keluaran sebelumnya, dua kueri pengguna dalam `"db.test"` namespace diblokir pada kunci koleksi: satu kueri di `"admin.$cmd"` namespace, dan satu tugas `"TTLMonitor"` internal.

Jika keluaran menunjukkan banyak kueri dengan memblokir keadaan tunggu, lihat Bagaimana cara menemukan dan menghentikan kueri yang berjalan lama atau diblokir?

Bagaimana cara menentukan penyebab pemanfaatan CPU yang tinggi pada satu atau lebih instance cluster?

Bagian berikut ini dapat membantu Anda mengidentifikasi penyebab pemanfaatan CPU instans yang tinggi. Hasil Anda dapat bervariasi bergantung pada beban kerja.

- Untuk menentukan mengapa instans tiba-tiba berjalan lambat, lihat Bagaimana cara menentukan mengapa sistem tiba-tiba berjalan lambat?

- Untuk mengidentifikasi dan mengakhiri kueri yang berjalan lamaa pada instans tertentu, lihat Bagaimana cara menemukan dan menghentikan kueri yang berjalan lama atau diblokir?
- Untuk memahami apakah kueri sedang berjalan, lihat Bagaimana saya tahu kapan kueri membuat kemajuan?
- Untuk menentukan mengapa kueri membutuhkan waktu yang lama untuk berjalan, lihat Bagaimana saya bisa melihat rencana kueri dan mengoptimalkan kueri?
- Untuk melacak kueri yang berjalan lama dari waktu ke waktu, lihat Membuat profil operasi Amazon DocumentDB.

Bergantung pada alasan untuk pemanfaatan CPU instans Anda yang tinggi, dengan melakukan satu atau lebih hal berikut ini dapat membantu.

- Jika instans primer menunjukkan pemanfaatan CPU yang tinggi, tetapi instans replika tidak menunjukkannya, pertimbangkan untuk mendistribusikan lalu lintas baca di replika melalui pengaturan preferensi baca klien (sebagai contoh, `secondaryPreferred`). Untuk informasi selengkapnya, lihat Menghubungkan ke Amazon DocumentDB sebagai set replika.

  Dengan menggunakan replika untuk pembacaan dapat membuat penggunaan klaster sumber daya yang lebih baik dengan mengizinkan instans primer untuk memproses lebih banyak lalu lintas tulis. Pembacaan dari replika pada akhirnya adalah konsisten.
- Jika pemanfaatan CPU yang tinggi adalah hasil dari beban kerja tulis Anda, mengubah ukuran instans klaster untuk jenis instans yang lebih besar meningkatkan jumlah inti CPU yang tersedia untuk layanan beban kerja. Untuk informasi selengkapnya, lihat Mengelola instans Amazon DocumentDB dan Spesifikasi kelas instans.
- Jika semua instans klaster menunjukkan pemanfaatan CPU yang tinggi, dan beban kerja menggunakan replika untuk pembacaan, akan menambahkan lebih banyak replika klaster akan meningkatkan sumber daya yang tersedia untuk lalu lintas baca. Untuk informasi selengkapnya, lihat Menambahkan instance Amazon DocumentDB ke klaster.

Bagaimana cara menentukan kursor terbuka pada sebuah instance?

Ketika terhubung ke instans Amazon DocumentDB, Anda dapat menggunakan perintah `db.runCommand("listCursors")` untuk mendaftar kursor terbuka pada instans tersebut. Terdapat batas hingga 4.560 kursor aktif terbuka pada waktu tertentu pada instans Amazon DocumentDB tertentu, bergantung pada tipe instans. Umumnya disarankan untuk menutup kursor

yang tidak lagi digunakan karena kursor memanfaatkan sumber daya pada instans dan memiliki batas atas. Lihat Kuota dan batas Amazon DocumentDB untuk batas spesifik.

```
db.runCommand("listCursors")
```

Bagaimana cara menentukan versi mesin Amazon DocumentDB saat ini?

Untuk menentukan versi mesin Amazon DocumentDB Anda saat ini, jalankan perintah berikut ini.

```
db.runCommand({getEngineVersion: 1})
```

Keluaran dari operasi ini terlihat seperti berikut ini (format JSON).

```
{ "engineVersion" : "2.x.x", "ok" : 1 }
```

> **Note**
>
> Versi mesin untuk Amazon DocumentDB 3.6 adalah 1.xx, versi mesin untuk Amazon DocumentDB 4.0 adalah 2.xx, versi mesin untuk Amazon DocumentDB 5.0 adalah 3.xx, dan versi mesin untuk Amazon DocumentDB 8.0 adalah 4.xx.

Bagaimana cara menganalisis penggunaan indeks dan mengidentifikasi indeks yang tidak digunakan?

Untuk mengidentifikasi indeks untuk koleksi tertentu, jalankan perintah berikut ini:

```
db.collection.getIndexes()
```

Untuk menganalisis berapa banyak indeks yang digunakan selama operasi yang dilakukan pada koleksi, `indexStats` perintah `collStats` dan dapat digunakan. Untuk melihat jumlah total pemindaian yang dilakukan menggunakan indeks (pemindaian indeks) dibandingkan dengan jumlah pemindaian yang dilakukan tanpa indeks (pemindaian koleksi), jalankan perintah berikut:

```
db.collection.stats()
```

Output untuk perintah ini mencakup nilai-nilai berikut:

- **idxScans**- Jumlah pemindaian yang dilakukan pada koleksi ini menggunakan indeks.
- **collScans**- Jumlah pemindaian yang dilakukan pada koleksi ini tanpa menggunakan indeks. Pemindaian ini akan melibatkan pemeriksaan dokumen dalam koleksi satu per satu.
- **lastReset**- Waktu penghitung ini telah diatur ulang terakhir. Statistik yang disediakan oleh perintah ini diatur ulang saat starting/stopping cluster atau penskalaan up/down instance.

Rincian berapa banyak setiap indeks digunakan dapat ditemukan dalam output dari perintah berikut. Ini adalah praktik terbaik untuk secara teratur mengidentifikasi dan menghapus indeks yang tidak digunakan untuk meningkatkan kinerja dan mengurangi biaya, karena menghilangkan komputasi, penyimpanan, dan I/Os penggunaan yang tidak perlu untuk mempertahankan indeks.

```
db.collection.aggregate([{$indexStats:{}}]).pretty()
```

Output dari perintah ini memberikan nilai-nilai berikut untuk setiap indeks yang dibuat pada koleksi:

- **ops**- Jumlah operasi yang menggunakan indeks. Jika beban kerja Anda telah berjalan selama waktu yang cukup lama dan Anda yakin bahwa beban kerja Anda dalam keadaan stabil, nilai ops sebesar nol akan menunjukkan bahwa indeks tidak digunakan sama sekali.
- **numDocsRead**- Jumlah dokumen yang dibaca selama operasi menggunakan indeks ini.
- **since**- Waktu sejak Amazon DocumentDB mulai mengumpulkan statistik tentang penggunaan indeks, yang biasanya merupakan nilai sejak restart database terakhir atau tindakan pemeliharaan.
- **size**- Ukuran indeks ini dalam byte.

Contoh berikut adalah contoh output dari menjalankan perintah di atas:

```
{
    "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" : ...
    }
}
```

Untuk menentukan ukuran indeks keseluruhan untuk koleksi, jalankan perintah berikut ini:

```
db.collection.stats()
```

Untuk menghapus indeks yang tidak digunakan, jalankan perintah berikut ini:

```
db.collection.dropIndex("indexName")
```

Bagaimana cara mengidentifikasi indeks yang hilang?

Anda dapat menggunakan profiler Amazon DocumentDB untuk log kueri yang lambat. Kueri yang muncul berulang kali di log kueri lambat mungkin menunjukkan bahwa indeks tambahan diperlukan untuk meningkatkan performa kueri tersebut.

Anda dapat mengidentifikasi peluang untuk indeks bermanfaat dengan mencari kueri berjalan lama yang memiliki satu atau lebih tahapan yang melakukan setidaknya satu `COLLSCAN` tahap, yang

berarti bahwa tahap kueri harus membaca setiap dokumen dalam koleksi untuk memberikan respons terhadap kueri.

Contoh berikut ini menunjukkan kueri pada koleksi perjalanan taksi yang berjalan pada koleksi besar.

```
db.rides.count({"fare.totalAmount":{$gt:10.0}}))
```

Untuk menjalankan contoh ini, kueri harus melakukan pemindaian koleksi (yaitu membaca setiap dokumen tunggal dalam koleksi) karena tidak ada indeks pada bidang `fare.totalAmount`. Keluaran dari profiler Amazon DocumentDB untuk kueri ini terlihat seperti berikut ini:

```
{
    ...
    "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"
}
```

Untuk mempercepat kueri dalam contoh ini, Anda ingin membuat indeks pada `fare.totalAmount`, sebagaimana ditunjukkan di bawah ini.

```
db.rides.createIndex( {"fare.totalAmount": 1}, {background: true} )
```

> **Note**
>
> Indeks yang dibuat di latar depan (artinya jika opsi `{background:true}` tidak disediakan saat membuat indeks) ambil kunci tulis eksklusif, yang mencegah aplikasi agar tidak menulis data ke koleksi hingga pembuatan indeks selesai. Waspadalah dampak potensial

ini saat membuat indeks pada klaster produksi. Saat membuat indeks, kami menyarankan pengaturan `{background:true}`.

Secara umum, Anda ingin membuat indeks pada bidang yang memiliki kardinalitas tinggi (sebagai contoh, sejumlah besar nilai unik). Membuat indeks di bidang dengan kardinalitas rendah dapat menghasilkan indeks besar yang tidak digunakan. Pengoptimal kueri Amazon DocumentDB menganggap ukuran keseluruhan koleksi dan selektivitas indeks saat membuat rencana kueri. Ada kalanya Anda akan melihat prosesor kueri memilih `COLLSCAN` bahkan ketika ada indeks. Hal ini terjadi ketika kueri prosesor memperkirakan bahwa pemanfaatan indeks tidak akan menghasilkan keuntungan performa pada saat memindai seluruh koleksi. Jika Anda ingin memaksa prosesor kueri untuk memanfaatkan indeks tertentu, Anda dapat menggunakan operator `hint()` sebagaimana ditunjukkan di bawah ini.

```
db.collection.find().hint("indexName")
```

Bagaimana cara menentukan pengumpulan database bloat?

Collection bloat terjadi ketika ukuran koleksi menjadi lebih besar karena akumulasi dokumen mati atau usang atau fragmentasi dalam halaman database. Persentase yang dilaporkan mewakili jumlah ruang dokumen yang dapat digunakan oleh dokumen future. Bloat ini menghabiskan ruang di cache buffer dan penyimpanan. Untuk menghapus bloat, koleksi harus dimuat ulang baik melalui dump/restore atau menggunakan migrasi loop-back dan switchover selama jendela pemeliharaan.

Example Contoh

Jalankan perintah berikut untuk menentukan penyimpanan yang tidak digunakan untuk koleksi Anda:

```
db.runCommand({collStats:'coll'})
```

Hasilnya terlihat mirip dengan ini:

```
{
        "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)
}
```

Ringkasan pertanyaan yang berguna

Kueri berikut ini dapat berguna untuk memantau peforma dan pemanfaatan sumber daya di Amazon DocumentDB.

- Gunakan perintah berikut untuk melihat statistik tentang koleksi tertentu, termasuk penghitung operasi, statistik cache, statistik akses, dan statistik ukuran:

```
db.collection.stats()
```

- Gunakan perintah berikut untuk melihat statistik tentang setiap indeks yang dibuat pada koleksi termasuk ukuran indeks, statistik cache khusus indeks, dan statistik penggunaan indeks:

```
db.collection.aggregate([{$indexStats:{}}]).pretty()
```

- Gunakan kueri berikut ini untuk mendaftar semua aktivitas.

```
db.adminCommand({currentOp: 1, $all: 1});
```

- Kode berikut ini mencantumkan semua kueri yang berjalan lama atau diblokir.

```
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: {}
                });
```

- Kode berikut ini mengakhiri kueri.

```
db.adminCommand({killOp: 1, op: <opid of running or blocked query>});
```

- Gunakan kode berikut ini untuk mendapatkan tampilan agregat dari keadaan sistem.

```
db.adminCommand({aggregate: 1,
                 pipeline: [{$currentOp: {allUsers: true, idleConnections: true}},
                            {$group: {_id: {desc: "$desc", ns: "$ns", WaitState:
 "$WaitState"}, count: {$sum: 1}}}],
                 cursor: {}
                });
```

Pengumpulan sampah di Amazon DocumentDB

Amazon DocumentDB mengimplementasikan arsitektur database kontrol konkurensi multi-versi (MVCC) yang membuat versi baru entri dokumen dan indeks untuk setiap operasi pembaruan. Arsitektur ini menyediakan memungkinkan isolasi transaksi, mencegah perubahan satu transaksi muncul di transaksi lain.

Topik

- Memahami pengumpulan Sampah di Amazon DocumentDB
- Proses pengumpulan sampah
- Arsitektur penyimpanan dan penyimpanan yang diperluas
- Memantau pengumpulan sampah
- Contoh keluaran CollStats
- Pertanyaan umum

Memahami pengumpulan Sampah di Amazon DocumentDB

Pengumpulan sampah (GC) adalah proses latar belakang otomatis yang mempertahankan kinerja dan ketersediaan sistem yang optimal di Amazon DocumentDB. Seperti banyak database modern, arsitektur MVCC Amazon DocumentDB membuat versi dokumen dan indeks baru dengan setiap pembaruan. Setiap operasi penulisan menggunakan ID MVCC unik dari penghitung terbatas. ID ini mengidentifikasi transaksi mana yang dimiliki versi dokumen dan apakah transaksi tersebut telah dilakukan atau dibatalkan. Seiring waktu, versi lama ini dan ID MVCC mereka menumpuk, membutuhkan pembersihan untuk mencegah penurunan kinerja.

Fungsi pengumpulan sampah

Pengumpul sampah melayani tiga fungsi penting:

- Mendapatkan kembali ruang penyimpanan - Ini menghapus dokumen usang dan versi indeks yang tidak lagi diperlukan oleh kueri aktif, membebaskan ruang untuk operasi penulisan di masa depan.
- Mencegah luapan ID MVCC — Ini mencegah luapan ID MVCC dengan mengelola penghitung terbatas ID MVCC. Tanpa manajemen ini, penghitung pada akhirnya akan mencapai batasnya, memaksa database ke mode read-only sementara hingga ID didaur ulang.

- Mempertahankan kinerja kueri — Ini mempertahankan kinerja kueri yang optimal dengan menghilangkan versi dokumen mati yang jika tidak akan menumpuk dan memperlambat pemrosesan kueri.

Proses pengumpulan sampah

Proses GC beroperasi per koleksi dan dapat memiliki beberapa proses yang berjalan bersamaan pada koleksi yang berbeda. Prosesnya terdiri dari empat fase berurutan:

1. Identifikasi — Sistem mengidentifikasi versi dokumen dan indeks yang tidak lagi direferensikan oleh transaksi atau kueri aktif.
2. Memory loading — Dokumen lama dan entri indeks dimuat ke dalam memori jika belum ada.
3. Penghapusan - Versi usang dihapus secara permanen untuk merebut kembali ruang penyimpanan.
4. Daur ulang ID MVCC — Sistem mendaur ulang ID MVCC dari versi yang dihapus untuk operasi baru.

Ketika pengumpulan sampah selesai memproses versi dokumen lama, ia menghapus ID MVCC tertua dari sistem. Pembersihan ini sangat penting untuk mencegah luapan ID MVCC dengan mendaur ulang ID MVCC, membuatnya tersedia untuk operasi penulisan baru di seluruh cluster. Tanpa proses daur ulang ini, sistem pada akhirnya akan menghabiskan penghitung ID MVCC yang terbatas dan memasuki status hanya-baca.

Penjadwalan pengumpulan sampah

Pengumpulan sampah berjalan secara otomatis di latar belakang pada interval berkala. Waktu dan frekuensi menyesuaikan secara dinamis berdasarkan beban sistem, sumber daya yang tersedia, volume tulis, dan tingkat konsumsi ID MVCC. Selama aktivitas tulis tinggi, proses GC mengeksekusi lebih sering untuk mengelola peningkatan jumlah versi dokumen.

Arsitektur penyimpanan dan penyimpanan yang diperluas

Amazon DocumentDB menggunakan arsitektur penyimpanan canggih yang memisahkan penyimpanan dokumen menjadi dua segmen berbeda:

Segmen penyimpanan dasar

Segmen penyimpanan dasar berisi data dokumen utama dan metadata. Segmen ini menyimpan:

- Konten dokumen yang sesuai dengan ukuran halaman standar (8 KB).
- Dokumen metadata dan informasi struktur.
- Indeks primer dan entri mereka.
- Collection-level statistik dan konfigurasi.

Segmen penyimpanan yang diperluas

Segmen penyimpanan yang diperluas menggunakan toko objek dokumen besar khusus yang dirancang untuk menangani dokumen yang melebihi ukuran halaman penyimpanan standar. Segmen ini menyediakan:

- Penanganan Dokumen Besar yang Efisien — Dokumen yang lebih besar dari ambang penyimpanan dasar secara otomatis dipindahkan ke segmen penyimpanan yang diperluas.
- Tata Letak Penyimpanan yang Dioptimalkan - Segmen ini menggunakan format penyimpanan berbeda yang dioptimalkan untuk objek besar, mengurangi fragmentasi dan meningkatkan pola akses.
- Pengumpulan Sampah Independen — Segmen penyimpanan yang diperluas memiliki proses pengumpulan sampah sendiri yang dapat berjalan secara independen dari pembersihan penyimpanan dasar.
- Akses Transparan — Aplikasi mengakses dokumen besar dengan mulus tanpa perlu mengetahui segmen penyimpanan mana yang berisi data.

Segmen penyimpanan yang diperluas sangat bermanfaat untuk:

- Koleksi dengan dokumen yang berisi array tertanam besar.
- Dokumen dengan struktur bersarang yang luas.
- Koleksi menyimpan data biner atau bidang teks besar.
- Aplikasi dengan ukuran dokumen campuran di mana beberapa dokumen secara signifikan melebihi ukuran rata-rata.

Memantau pengumpulan sampah

Metrik tingkat cluster

`AvailableMVCCIds`

- Lokasi - Amazon CloudWatch
- Deskripsi — Penghitung yang menunjukkan jumlah operasi tulis yang tersisa yang tersedia dari batas maksimum 1,8 miliar. Saat penghitung ini mencapai nol, klaster Anda memasuki mode hanya-baca hingga ID direklamasi dan didaur ulang. Penghitung berkurang dengan setiap operasi penulisan dan meningkat saat pengumpulan sampah mendaur ulang ID MVCC lama.
- Rekomendasi — Atur alarm saat nilainya turun di bawah 1,3 miliar. Peringatan dini ini memungkinkan Anda untuk mengambil langkah-langkah yang disarankan dibahas nanti.

`LongestActiveGCRuntime`

- Lokasi - Amazon CloudWatch
- Deskripsi — Durasi dalam hitungan detik proses pengumpulan sampah aktif terpanjang. Memperbarui setiap menit dan hanya melacak operasi aktif, tidak termasuk proses yang selesai dalam jendela satu menit.
- Rekomendasi — Bandingkan dengan data `gcRuntimeStats` historis untuk mengidentifikasi perilaku pengumpulan sampah yang tidak normal, seperti runtime yang diperpanjang selama penghapusan massal.

Metrik tingkat pengumpulan

`MVCCIDStats: MVCCIdScale`

- Lokasi - Perintah CollStats Database
- Deskripsi — Mengukur usia ID MVCC pada skala 0 hingga 1, di mana 1 menunjukkan usia maksimum sebelum klaster memasuki status hanya-baca. Gunakan metrik ini bersama `AvailableMVCCIds` untuk mengidentifikasi koleksi yang berisi ID MVCC tertua yang menua klaster.
- Rekomendasi - Pertahankan nilai di bawah 0,3 untuk setiap koleksi.

`gcRuntimeStats`

- Lokasi - Perintah CollStats Database
- Deskripsi — Memberikan riwayat metrik pengumpulan sampah selama dua bulan, termasuk total proses, durasi rata-rata, dan durasi maksimum. Hanya mencakup operasi pengumpulan sampah yang berlangsung lebih dari lima menit untuk memastikan statistik yang berarti.

`storageSizeStats`

- Lokasi - Perintah CollStats Database
- Deskripsi - Memberikan rincian rinci pemanfaatan penyimpanan di segmen penyimpanan yang berbeda:
  - `storageSegmentBase`— Penyimpanan yang digunakan oleh segmen penyimpanan dasar untuk dokumen standar
  - `storageSegmentExtended`— Penyimpanan yang digunakan oleh segmen penyimpanan yang diperluas untuk dokumen besar
- Penggunaan - Membantu mengidentifikasi koleksi dengan penyimpanan dokumen besar yang signifikan dan memahami pola distribusi penyimpanan.

`unusedStorageSize`(tingkat pengumpulan)

- Lokasi - Perintah CollStats Database
- Deskripsi — Memperkirakan ruang penyimpanan yang tidak digunakan dalam koleksi berdasarkan statistik sampel. Ini termasuk ruang dari dokumen yang dihapus dan segmen kosong. Metrik ini menyediakan total gabungan dan kerusakan per segmen:
  - Gabungan `unusedBytes` dan `unusedPercent` di semua segmen penyimpanan
  - `storageSegmentBase`— Ruang yang tidak terpakai khususnya di segmen penyimpanan dasar
  - `storageSegmentExtended`— Ruang yang tidak terpakai khususnya di segmen penyimpanan yang diperluas

`documentFragmentStats`

- Lokasi - Perintah CollStats Database
- Deskripsi — Memberikan informasi rinci tentang fragmen dokumen dan data mati dalam koleksi. Fragmen dokumen mewakili unit penyimpanan internal yang digunakan oleh mesin database, dan fragmen mati menunjukkan data yang tidak lagi dapat diakses tetapi belum direklamasi. Metrik ini meliputi:
  - `totalDocFragmentsCount`— Jumlah total fragmen dokumen dalam koleksi
  - `deadDocFragmentsCount`— Jumlah fragmen yang berisi data mati (tidak dapat diakses)
  - `deadDocFragmentsPercent`— Persentase fragmen yang berisi data mati
  - `deadDocFragmentBytes`— Perkiraan byte yang dikonsumsi oleh fragmen dokumen mati

  - Per-segment kerusakan untuk `storageSegmentBase` dan `storageSegmentExtended`
- Penggunaan — Pantau metrik ini untuk memahami efektivitas pengumpulan sampah dan mengidentifikasi koleksi yang mungkin mendapat manfaat dari operasi pemeliharaan. Persentase fragmen mati yang tinggi menunjukkan bahwa pengumpulan sampah mungkin tertinggal atau bahwa pengumpulan akan mendapat manfaat dari pengoptimalan.

Metrik tingkat indeks

`unusedStorageSize`(tingkat indeks)

- Lokasi - Perintah Database IndexStats
- Deskripsi — Memperkirakan ruang penyimpanan yang tidak digunakan dalam indeks berdasarkan statistik sampel. Ini termasuk ruang dari entri indeks usang dan segmen kosong.
- Rekomendasi — Gunakan `reIndex` perintah untuk membangun kembali indeks tanpa downtime dan merebut kembali ruang yang tidak digunakan. Lihat Mengelola Indeks untuk detail selengkapnya.

Contoh keluaran CollStats

Contoh berikut menunjukkan `collStats` output tipikal dengan pengumpulan sampah dan metrik penyimpanan:

```
{
    "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)
}
```

Pertanyaan umum

Bagaimana cara mengidentifikasi jika pengumpulan sampah tidak bekerja secara efisien?

Pantau tanda-tanda peringatan ini yang menunjukkan pengumpulan sampah yang tidak efisien:

- Excess Collection Bloat — `unusedStorageSize` Metrik yang terus meningkat selama penulisan berat atau penghapusan massal, terutama dengan indeks besar.
- Persentase Fragmen Mati Tinggi - `documentFragmentStats` menunjukkan `deadDocFragmentsPercent` nilai tinggi secara konsisten (di atas 10-15%).
- Latensi Kueri Terdegradasi — Peningkatan latensi kueri karena akumulasi dokumen mati.
- Durasi GC yang Diperpanjang — Operasi pengumpulan sampah memakan waktu lebih lama dari rata-rata historis di. `gcRuntimeStats`
- Elevated GC Processing — Tinggi `LongestActiveGCRuntime` menunjukkan bahwa pengumpul sampah tidak dapat memenuhi permintaan sistem.

Apakah pengumpulan sampah mempengaruhi kinerja database saya?

Dalam kondisi normal, pengumpulan sampah memiliki dampak kinerja minimal. Namun, ketika pengumpulan sampah tertinggal, Anda mungkin mengalami:

- Peningkatan biaya penyimpanan dari akumulasi dokumen mati.

- Kinerja kueri lebih lambat karena entri indeks usang.
- Mode hanya-baca sementara jika ID MVCC habis.
- Penggunaan sumber daya yang lebih tinggi selama proses pengumpulan intensif, terutama pada instance yang lebih kecil.
- Mengurangi efisiensi dalam operasi segmen penyimpanan yang diperluas untuk dokumen besar.

Bisakah saya memicu pengumpulan sampah secara manual?

Tidak, pengumpulan sampah di Amazon DocumentDB tidak dapat dipicu secara manual. Sistem mengelola pengumpulan sampah secara otomatis sebagai bagian dari operasi pemeliharaan internalnya.

Alarm apa yang harus saya tetapkan sebagai praktik terbaik operasional?

Sebaiknya siapkan pemantauan di tingkat cluster dan koleksi untuk memastikan kinerja optimal sistem Amazon DocumentDB Anda.

Untuk pemantauan tingkat cluster, mulailah dengan membuat CloudWatch alarm Amazon untuk `AvailableMVCCIds` metrik dengan ambang 1,3 miliar. Ini memberi Anda waktu yang cukup untuk mengambil tindakan sebelum metrik mencapai nol, di mana klaster Anda akan memasuki mode hanya-baca. Perlu diingat bahwa metrik ini dapat berfluktuasi berdasarkan pola penggunaan spesifik Anda - beberapa pelanggan melihatnya turun di bawah 1,3 miliar dan kemudian pulih di atas 1,5 miliar saat pengumpulan sampah menyelesaikan pekerjaannya.

Penting juga untuk memantau `LongestActiveGCRuntime` metrik melalui Amazon CloudWatch. Metrik ini, bersama dengan`gcRuntimeStats`, membantu Anda memahami seberapa efisien kinerja pengumpulan sampah di seluruh sistem Anda.

Untuk pemantauan tingkat pengumpulan, fokuslah pada metrik utama ini:

- `MVCCIdScale`Perhatikan peningkatan nilai yang menunjukkan ID MVCC menua dan mungkin perlu diperhatikan.
- `gcRuntimeStats`— Identifikasi proses pengumpulan sampah yang memakan waktu yang sangat lama atau diperpanjang selama beberapa hari.
- `documentFragmentStats`— `deadDocFragmentsPercent` Nilai monitor - persentase tinggi secara konsisten (di atas 10-15%) dapat mengindikasikan pengumpulan sampah tertinggal.
- `storageSizeStats`dan `unusedStorageSize` — Lacak pola pemanfaatan penyimpanan dan identifikasi koleksi dengan ruang signifikan yang tidak terpakai di kedua segmen penyimpanan.

Koleksi dengan operasi menulis yang sering membutuhkan perhatian ekstra, karena mereka menghasilkan lebih banyak pekerjaan untuk pengumpul sampah. Sebaiknya periksa metrik ini lebih sering untuk koleksi dengan aktivitas menulis berat untuk memastikan pengumpulan sampah mengikuti beban kerja Anda.

Perhatikan bahwa rekomendasi pemantauan ini berfungsi sebagai titik awal. Saat Anda menjadi lebih akrab dengan perilaku sistem Anda, Anda mungkin ingin menyesuaikan ambang batas ini agar lebih sesuai dengan pola dan persyaratan penggunaan spesifik Anda.

Apa yang harus saya lakukan jika **`AvailableMvccids`** saya turun di bawah 1,3 miliar?

Jika `AvailableMVCCIds` metrik Anda turun di bawah 1,3 miliar, sebaiknya segera ambil tindakan untuk mencegah klaster Anda memasuki mode hanya-baca. Kami sarankan terlebih dahulu meningkatkan ukuran instans Anda untuk menyediakan sumber daya komputasi yang lebih banyak kepada pengumpul sampah. Ini adalah rekomendasi utama kami karena memungkinkan aplikasi Anda untuk melanjutkan operasi normal sambil memberi pengumpul sampah daya tambahan yang dibutuhkan untuk mengejar ketinggalan.

Jika penskalaan saja tidak memperbaiki situasi, sebaiknya pertimbangkan pengurangan operasi penulisan Anda. Gunakan `MVCCIdScale` metrik untuk mengidentifikasi koleksi spesifik mana yang berisi ID MVCC lama yang perlu diperhatikan. Selain itu, pantau `documentFragmentStats` untuk mengidentifikasi koleksi dengan persentase fragmen mati tinggi yang mungkin berkontribusi terhadap inefisiensi pengumpulan sampah.

Setelah Anda mengidentifikasi koleksi ini, Anda mungkin perlu untuk sementara mengurangi operasi penulisan untuk memungkinkan pengumpulan sampah untuk mengejar ketinggalan. Selama periode pemulihan, kami sarankan untuk memantau `AvailableMVCCIds` metrik dengan cermat untuk memastikan tindakan Anda memiliki efek yang diinginkan. Cluster Anda dianggap sehat setelah `AvailableMVCCIds` nilainya kembali ke 1,5 miliar atau lebih tinggi.

Ingatlah bahwa langkah-langkah ini adalah tindakan pencegahan untuk membantu sistem Anda pulih sebelum mencapai keadaan kritis. Semakin cepat Anda mengambil tindakan setelah melihat penurunan metrik di bawah 1,3 miliar, semakin besar kemungkinan Anda untuk menghindari dampak apa pun pada operasi penulisan Anda.

Catatan rilis

Catatan rilis ini menjelaskan fitur Amazon DocumentDB, peningkatan, dan perbaikan bug yang diatur oleh versi mesin utama. Dalam setiap bagian versi, entri dicantumkan berdasarkan tanggal dalam format tabel.

Untuk detail tentang bagaimana versi engine dan versi patch diberi nomor (termasuk pemetaan antara awalan versi patch engine dan versi mesin Amazon DocumentDB), lihat. Penomoran versi mesin

Anda dapat menentukan versi patch mesin Amazon DocumentDB saat ini dengan menjalankan perintah berikut:

```
db.runCommand({getEngineVersion: 1})
```

Jika klaster Anda tidak menggunakan versi terbaru mesin, kemungkinan Anda memiliki pemeliharaan tersedia yang tertunda yang akan meningkatkan mesin Anda. Untuk informasi selengkapnya, lihat Memelihara Amazon DocumentDB di Panduan Pengembang.

Topik

- Catatan rilis versi mesin Amazon DocumentDB
- Catatan rilis umum Amazon DocumentDB
- Catatan rilis cluster elastis Amazon DocumentDB

Catatan rilis versi mesin Amazon DocumentDB

Catatan rilis untuk versi mesin Amazon DocumentDB yang diatur oleh versi utama.

Amazon DocumentDB 8.0

Catatan rilis untuk mesin Amazon DocumentDB versi 8.0 (kompatibilitas MongoDB 8.0).

Date	Catatan rilis
13 Mei 2026	Fitur baru

Date	Catatan rilis
	Amazon DocumentDB tanpa server sekarang tersedia untuk Amazon DocumentDB 8.0. Untuk informasi selengkapnya, lihat Menggunakan Amazon DocumentDB tanpa server.
6 Mei 2026	Perbaikan bug dan perubahan lainnya Amazon DocumentDB 8.0 (Versi Mesin: 4.0.6940) Pembaruan ini menyediakan patch mesin opsional dengan perbaikan berikut: 1. Memperbaiki bug dalam pengumpulan sampah yang dapat, dalam kasus tepi, menyebabkan pembaca mogok.
April 20, 2026	Fitur baru In-place upgrade versi utama (MVU) dari Amazon DocumentDB 5.0 ke Amazon DocumentDB 8.0 sekarang tersedia. Untuk informasi selengkapnya, lihat Peningkatan versi utama Amazon DocumentDB di tempat.

Date	Catatan rilis
April 20, 2026	Fitur dan peningkatan baru Amazon DocumentDB 8.0 (Versi Mesin: 4.0.5995, 4.0.6473) Ini adalah patch mesin yang diperlukan yang mencakup fitur, peningkatan, dan perbaikan baru berikut: 1. `$sortArray` operator di`$project`: • Support untuk `$sortArray` operator secara `$project` bertahap dalam pipeline agregasi untuk Planner versi 3. `$sortArray` Operator mengurutkan array berdasarkan elemennya (untuk nilai skalar) atau pada bidang tertentu (untuk array dokumen), mengembalikan array yang diurutkan dalam urutan ascending (1) atau descending (). `-1` 2. Support untuk pembaruan massal paralel yang tidak teratur di`bulkWrite()` . 3. Support untuk`$out`,`$collStats` ,`$indexStats` , `$geoNear` menggunakan Planner v3. Dengan Planner V3, `$out` panggung juga mendukung penulisan ke koleksi di berbagai database. 4. Pipa agregasi sekarang menyatu `$lookup` dan `$unwind` tahapan bahkan ketika tahap menyertakan parameter`$unwind`. `includeArrayIndex` Sebelumnya, optimasi ini dilewati saat `includeArrayIndex` hadir. 5. Peningkatan kinerja untuk meningkatkan throughput penulisan untuk db.r8g.8xlarge dan instance yang lebih besar yang berjalan pada konfigurasi. I/O-Optimized 6. Mengekspos metrik `Connections (active)` untuk memantau instance. Batasannya dapat ditemukan di halaman Kuota dan batas Amazon DocumentDB kami. 7. Meningkatkan ketahanan lintas wilayah Global Cluster dengan menghilangkan reboot cluster sekunder dengan Global Survivable Read Only. 8. Support untuk `$dateToParts` dan operator `$dateFromParts` agregasi.

Date	Catatan rilis
	9. Command-level Metrik kinerja sekarang tersedia CloudWatch untuk operasi CRUD (`find`,,,`insert`,`findAndModify` ,`update`,`delete`,`aggregate` ,`count`,`distinct`,`getMore`,`abortTransaction` ,, `commitTransaction` `AvgDuration` `P100Duration` , dan `MaxConcurrent` dimensi untuk pemantauan operasional yang lebih baik. 10. Memperbaiki bug dalam `$replaceRoot` perilaku di mana `newRoot` bidang yang berisi nilai boolean`true`/`false`atau nilai numerik`0`/`1`hilang dalam dokumen keluaran. 11. Memperbaiki bug di `$indexOfArray` mana kueri yang valid mengembalikan kesalahan yang menunjukkan bahwa mereka tidak valid. 12. Rencana penjelasan sekarang menyertakan `docsExamined` metrik dalam `executionStats` output. 13. Peningkatan kinerja otentikasi SASL. 14. Pembaruan dokumen yang dioptimalkan untuk mengurangi overhead penyimpanan saat nilai indeks tetap tidak berubah dan dokumen sesuai dengan halaman yang ada. 15. Support untuk `bypassDocumentValidation` di`$merge`. Semua cluster Amazon DocumentDB 8.0 akan secara otomatis ditingkatkan ke versi patch ini selama jendela pemeliharaan berikutny a.
Maret 26, 2026	Fitur baru Amazon DocumentDB sekarang mendukung generasi terbaru dari instans yang dioptimalkan memori, db.r8g, didukung Arm-based AWS oleh prosesor Graviton4, memberikan kinerja hingga 30% lebih baik dibandingkan instans R6G. Untuk informasi lebih lanjut, lihat Mengelola kelas instance di panduan ini

Date	Catatan rilis
November 11, 2025	Fitur baru Amazon DocumentDB sekarang menawarkan kompatibilitas protokol kawat penuh dengan MongoDB 8.0. Amazon DocumentDB 8.0 meningkatkan kinerja kueri yang memungkinkan Anda membangun aplikasi berkinerja tinggi dengan biaya lebih rendah. Amazon DocumentDB 8.0 (Versi Patch Mesin 4.0.4448) • Menambahkan dukungan untuk driver MongoDB 8.0 API. • Menambahkan dukungan untuk Planner Version3 yang memperluas peningkatan kinerja ke operator tahap agregasi, bersama dengan mendukung optimasi saluran agregasi dan perintah yang berbeda. • Menambahkan Support untuk kompresi berbasis kamus melalui algoritma kompresi Zstandard, meningkatkan rasio kompresi sebesar 5x. • Menambahkan kemampuan baru: Amazon DocumentDB 8.0 mendukung pemeriksaan dan tampilan. • Menambahkan dukungan untuk tahapan agregasi dan operator baru: $replaceWith, $vectorSearch, $merge, $set, $unset, $bucket, dan 3 operator agregasi baru $pow, $rand, $dateTrunc. • Menambahkan dukungan untuk versi baru indeks teks: Indeks teks v2 di Amazon DocumentDB 8.0 memperkenalkan token tambahan, meningkatkan kemampuan pencarian teks. • Termasuk peningkatan pencarian vektor melalui pembuatan indeks vektor paralel yang mengurangi waktu pembuatan indeks hingga 30x. • Dihapus dukungan untuk TLS 1.0 dan TLS 1.1. Versi TLS minimum sekarang TLS 1.2.

Amazon DocumentDB 5.0

Catatan rilis untuk Amazon DocumentDB engine versi 5.0 (kompatibilitas MongoDB 5.0).

Date	Versi minor	Catatan rilis
Juni 8, 2026	5.0.1	Versi minor baru Amazon DocumentDB 5.0.1 (Versi Mesin 3.0.21138) Amazon DocumentDB 5.0.1 adalah rilis versi minor pertama untuk Amazon DocumentDB 5.0 versi utama. Versi minor mencakup peningkatan seperti fitur baru, perbaikan keamanan, dan perbaikan bug. Anda dapat meningkatkan ke 5.0.1 dari 5.0.0 menggunakan upgrade versi minor. Untuk informasi selengkapnya, lihat Peningkatan versi minor Amazon DocumentDB. Versi minor ini mencakup fitur dan peningkatan baru berikut: 1. Ditambahkan `gcRuntimeStats` , `documentFragmentStats` , dan pemecahan metrik tingkat pengumpulan ke dalam `storageSegmentBase` dan ke `storageSegmentExtended` dalam `collstats` untuk memunculkan lebih banyak informasi ke dalam kesehatan keseluruhan dari proses pengumpulan sampah. Lihat Pengumpulan sampah di Amazon DocumentDB untuk detail. 2. Support untuk `$rand` dan operator `$pow` agregasi. 3. Mengekspos metrik `Connections (active)` untuk memantau instance. Batasannya dapat ditemukan di halaman Kuota dan batas Amazon DocumentDB kami. 4. Support untuk `$dateToParts` dan operator `$dateFromParts` agregasi. 5. Command-level Metrik kinerja sekarang tersedia CloudWatch untuk operasi CRUD (`find`,,,`insert`,`findAndModify` ,`update`,`delete`,`aggregate` ,`count`,`distinct`,`getMore`,`abortTran`

Date	Versi minor	Catatan rilis
		`saction` ,, `commitTransaction AvgDurati on P100Duration` , dan `MaxConcurrent` dimensi untuk pemantauan operasional yang lebih baik.
6 Mei 2026	5.0.0	Perbaikan bug dan perubahan lainnya Amazon DocumentDB 5.0 (Versi Mesin: 3.0.20421) Rilis ini untuk patch keamanan yang diperlukan.
April 20, 2026	5.0.0	Fitur baru In-place upgrade versi utama (MVU) dari Amazon DocumentDB 5.0 ke Amazon DocumentDB 8.0 sekarang tersedia. Untuk informasi selengkapnya, lihat Peningkatan versi utama Amazon DocumentDB di tempat.
April 13, 2026	5.0.0	Perbaikan bug dan perubahan lainnya Amazon DocumentDB 5.0 (Versi Mesin: 3.0.19917) Ini adalah patch mesin yang diperlukan yang mencakup fitur, peningkatan, dan perbaikan baru berikut: 1. Menambahkan DatabaseConnectionsActiveMax metrik untuk memberikan wawasan tentang “Koneksi (aktif)” untuk instance. Batasannya dapat ditemukan di halaman Kuota dan batas Amazon DocumentDB kami. 2. Peningkatan kinerja otentikasi SASL. Semua cluster Amazon DocumentDB 5.0 akan secara otomatis ditingkatkan ke versi patch ini selama jendela pemeliharaan berikutnya.

Date	Versi minor	Catatan rilis
Oktober 22, 2025	5.0.0	Fitur baru Amazon DocumentDB 5.0 (Versi Patch Mesin 3.0.17983) Amazon DocumentDB sekarang mendukung generasi terbaru dari instans yang dioptimalkan memori, db.r8g, didukung Arm-based AWS oleh prosesor Graviton4, memberikan kinerja hingga 30% lebih baik dibanding kan instans R6G. Untuk informasi lebih lanjut, lihat Mengelola kelas instance di panduan ini dan posting blog ini. Fitur baru Amazon DocumentDB 5.0 (Versi Patch Mesin 3.0.15902) • Menambahkan dukungan untuk nama koleksi yang lebih panjang (panjang namespace hingga 255 karakter). • Memperkenalkan perencana kueri baru (perencan a versi 2.0). Untuk informasi selengkapnya, lihat Perencana kueri v2 dalam panduan ini. • Menambahkan bidang baru `collstats` untuk “MVCcidStats” dan “gcRuntimeStats” yang memberikan informasi tentang kesehatan pengumpul an secara keseluruhan sehubungan dengan pengumpulan sampah dan pengumpulan penuaan.

Date	Versi minor	Catatan rilis
Oktober 16, 2025	5.0.0	Fitur baru Amazon DocumentDB 5.0 (Versi Patch Mesin 3.0.15902) Amazon DocumentDB sekarang mendukung mode dual-stack IPv4/IPv6 () saat cluster Anda berada di cloud pribadi virtual (VPC). Untuk informasi lebih lanjut, lihat posting blog ini dan Amazon VPC dan Amazon DocumentDB di Panduan Pengembang Amazon DocumentDB.
September 15, 2025	5.0.0	Perbaikan bug dan perubahan lainnya Amazon DocumentDB 5.0 (Versi Patch Mesin 3.0.17983) Memperbaiki kerusakan mesin saat kueri berisi tahap berikutnya `$replaceRoot` dan `$lookup` tahap.

Date	Versi minor	Catatan rilis
Juli 29, 2025	5.0.0	Fitur baru Amazon DocumentDB 5.0 (Versi Patch Mesin 3.0.15902) • Ditambahkan dukungan dari operator `$regexFindAll` agregasi. • Indeks bloat dan metrik bloat berbasis koleksi diluncurkan • CloudWatch Metrik yang ditambahkan `AvailableMVCCIds` dan `LongestActiveGCRuntime` yang memberikan informasi tentang kesehatan klaster secara keseluruhan sehubungan dengan pengumpulan sampah dan pengumpulan penuaan. • `serverStatus` Perintah sekarang menyertakan bidang “nvme_cache” yang menunjukkan jumlah halaman yang ditulis dan tidak ditulis ke cache NVMe pada instance. NVMe-backed • Batasan yang dihapus pada jumlah bidang untuk `$group` dan `$project` • Support untuk operator String berikut:`$trim`,`$rtrim`,`$ltrim`,`$regexFindAll` ,`$replaceOne` , dan `$replaceAll` • Support untuk operator Array berikut: `$first` dan `$last` • Support untuk operator Stage berikut: `$collStats` Perbaikan bug dan perubahan lainnya Amazon DocumentDB 5.0 (Versi Patch Mesin 3.0.15902) Memperbaiki masalah selama pembuatan indeks yang memengaruhi koleksi besar.

Date	Versi minor	Catatan rilis
Juli 28, 2025	5.0.0	Fitur baru Amazon DocumentDB 5.0 (Versi Patch Mesin 3.0.14228) Cluster berbasis instans Amazon DocumentDB sekarang mendukung Amazon DocumentDB Tanpa Server, konfigurasi auto-scaling sesuai permintaan. Untuk informasi lebih lanjut, lihat Menggunakan Amazon DocumentDB tanpa server dan ini apa posting blog baru.
Maret 24, 2025	5.0.0	Fitur baru Amazon DocumentDB 5.0 (Versi Patch Mesin 3.0.11800) Menambahkan dukungan untuk `postBatch ResumeToken` aliran perubahan. Untuk informasi selengkapnya, lihat `Melanjutkan aliran perubahan dengan posting BatchResumeToken` .

Date	Versi minor	Catatan rilis
Januari 15, 2025	5.0.0	Fitur baru Amazon DocumentDB 5.0 (Versi Patch Mesin 3.0.11800) Amazon DocumentDB sekarang mempertahankan ketersediaan baca melalui restart instance penulis. Instance pembaca sekarang akan terus melayani permintaan baca selama restart instance penulis. Perbaikan bug dan perubahan lainnya Amazon DocumentDB 5.0 (Versi Patch Mesin 3.0.11800) • `killOp`Perintah tetap untuk menangani kasus khusus sisipan massal. • Peningkatan penggunaan I/O-related memori jaringan pada instans Amazon DocumentDB. • `count`Perintah tetap untuk kueri `$text` filter.
Desember 18, 2024	5.0.0	Fitur baru Amazon DocumentDB 5.0 (Versi Patch Mesin 3.0.5721) Cluster berbasis instans Amazon DocumentDB sekarang mendukung instans R6GD. NVMe-backed Untuk informasi lebih lanjut, lihat NVMe-backed contoh dan ini apa posting blog baru.

Date	Versi minor	Catatan rilis
6 November 2024	5.0.0	Fitur baru Amazon DocumentDB 5.0 (Versi Patch Mesin 3.0.11051) • Menambahkan dukungan untuk versi TLS minimum `tls1.2+` dan `tls1.3+` parameter `tls` cluster. • Diaktifkan dukungan untuk karakter dolar ($) dan dot (.) dalam nama bidang. Untuk perbedaan fungsional, lihatDolar ($) dan titik (.) dalam nama bidang.

Date	Versi minor	Catatan rilis
September 17, 2024	5.0.0	Fitur baru Amazon DocumentDB 5.0 (Versi Patch Mesin 3.0.10696) Amazon DocumentDB sekarang mendukung pembangunan kembali indeks dengan di. `reIndex` `runCommand` Untuk informasi selengkapnya, lihat Pemeliharaan indeks menggunakan `reIndex`. **Note** `reIndex`hanya didukung di Amazon DocumentDB 5.0 (Engine Patch Version 3.0.10696 dan lebih tinggi). Perbaikan bug dan perubahan lainnya Amazon DocumentDB 5.0 (Versi Patch Mesin 3.0.10696) • `$setOnInsert` sekarang mendukung `_id` bidang selama penyisipan yang dihasilkan dari operasi pembaruan. • Memperbaiki masalah yang mencegah reklamasi ruang penyimpanan setelah upgrade versi utama ke Amazon DocumentDB 5.0.

Date	Versi minor	Catatan rilis
Agustus 22, 2024	5.0.0	Fitur baru Amazon DocumentDB 5.0 (Semua versi patch mesin) Cluster global Amazon DocumentDB sekarang mendukung pengalihan klaster dan failover cluster terkelola. Untuk informasi selengkapnya, lihat Melakukan peralihan untuk klaster global Amazon DocumentDB dan Melakukan failover terkelola untuk klaster global Amazon DocumentDB. **Note** Switchover dan failover cluster global hanya didukung di Amazon DocumentDB 4.0 dan 5.0.

<table>
<tr><th>Date</th><th>Versi minor</th><th>Catatan rilis</th></tr>
<tr><td>Juli 23, 2024</td><td>5.0.0</td><td>Fitur baru

Amazon DocumentDB 5.0 (Versi Patch Mesin 3.0.8126)

• Menambahkan dukungan untuk nama indeks yang lebih panjang (hingga 255 karakter). Untuk informasi selengkapnya, lihat Kendala penamaan
• Meningkatkan batas koneksi maksimum dua kali lipat:

<table>
<tr><th>Tipe instans</th><th>Batas asli</th><th>Batas baru</th></tr>
<tr><td>t3.medium</td><td>500</td><td>1000</td></tr>
<tr><td>t4g.medium</td><td>500</td><td>1000</td></tr>
<tr><td>r5.large</td><td>1700</td><td>3400</td></tr>
<tr><td>r5.xlarge</td><td>3500</td><td>7000</td></tr>
<tr><td>r5.2xlarge</td><td>7100</td><td>14200</td></tr>
<tr><td>r5.4xlarge</td><td>14200</td><td>28400</td></tr>
<tr><td>r5.8xlarge</td><td>28400</td><td>60000</td></tr>
<tr><td>r5.12xlarge</td><td>30000</td><td>60000</td></tr>
<tr><td>r5.16xlarge</td><td>30000</td><td>60000</td></tr>
<tr><td>r5.24xlarge</td><td>30000</td><td>60000</td></tr>
<tr><td>r6g.large</td><td>1700</td><td>3400</td></tr>
<tr><td>r6g.xlarge</td><td>3500</td><td>7000</td></tr>
<tr><td>r6g.2xlarge</td><td>7100</td><td>14200</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Date</th><th>Versi minor</th><th>Catatan rilis</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Tipe instans</th><th>Batas asli</th><th>Batas baru</th></tr>
<tr><td>r6g.4xlarge</td><td>14200</td><td>28400</td></tr>
<tr><td>r6g.8xlarge</td><td>28400</td><td>60000</td></tr>
<tr><td>r6g.12xlarge</td><td>30000</td><td>60000</td></tr>
<tr><td>r6g.16xlarge</td><td>30000</td><td>60000</td></tr>
</table>
Untuk informasi selengkapnya, lihat Batas instans.
Perbaikan bug dan perubahan lainnya
Amazon DocumentDB 5.0 (Versi Patch Mesin 3.0.8126)
Meningkatkan logika untuk menyinkronkan <code>CurrentTime</code> dan <code>ResumeToken</code> untuk aliran perubahan pada pembaca.
</td></tr>
</table>

Date	Versi minor	Catatan rilis

Juli 22, 2024	5.0.0	Fitur baru
		
		Amazon DocumentDB 5.0 (Versi Patch Mesin 3.0.6742)
		
		• Ditambahkan dukungan DMLAudit penyaringan. Anda sekarang dapat mengatur kondisi filter untuk menyaring log Audit DMLberdasarkan persyarat an spesifiknya alih-alih mencatat setiap kueri DML. Untuk informasi selengkapnya, lihat Memfilter acara audit DML..
		• Ditambahkan dukungan dalam kompresi dokumen untuk yang berikut:
		  • Menetapkan ambang kompresi minimum
		  • Mengaktifkan kompresi untuk koleksi yang ada (berlaku untuk dokumen baru)
		  • Izinkan pengaturan kompresi default di tingkat cluster
		
		  Untuk informasi selengkapnya, lihat Mengelola kompresi dokumen tingkat koleksi.
		• Menambahkan dukungan untuk mengkonsumsi aliran perubahan pada instance pembaca. Untuk informasi selengkapnya, lihat Menggunakan aliran perubahan pada instance sekunder.

9 Juli 2024	5.0.0	Fitur baru Untuk Amazon DocumentDB 5.0 (versi patch mesin minimum 3.0.4780), pembaruan sertifikat TLS CA tidak lagi memerlukan reboot sistem. Untuk informasi selengkapnya, lihat Memperbarui sertifikat Amazon DocumentDB TLS Anda.

Date	Versi minor	Catatan rilis
29 Mei 2024	5.0.0	Fitur baru Amazon DocumentDB 5.0 (Versi Patch Mesin 3.0.6742) • Ditambahkan dukungan untuk `$regexMatch` dan `$regexFind` operator. • Menambahkan dukungan untuk memastikan presisi penuh dalam log audit saat menangani bilangan bulat besar. Log audit sekarang mempertahankan representasi numerik yang tepat untuk semua angka, mencegah hilangnya presisi.
April 3, 2024	5.0.0	Fitur baru Amazon DocumentDB 5.0 (Versi Patch Mesin 3.0.5721) • Ditambahkan dukungan untuk `bypassDocumentValidation` dan pesan kesalahan granular untuk`$jsonSchema`. Untuk informasi selengkapnya tentang `bypassDocumentValidation`, lihat memintas DocumentValidation. • Menambahkan dukungan dari`$expr`. • Menambahkan dukungan untuk Uncorrelated Joins in. `$lookup` • Menambahkan dukungan untuk mempertahankan aturan validasi dalam tahap `$out` agregasi.

Date	Versi minor	Catatan rilis
10 Januari 2024	5.0.0	Fitur baru Amazon DocumentDB 5.0 (Versi Patch Mesin 3.0.4574, 3.0.4780, 3.0.4960) • Ditambahkan dukungan untuk indeks vektor HNSW. Untuk informasi selengkapnya, lihat Pencarian vektor untuk Amazon DocumentDB. • Menambahkan dukungan untuk indeks paral. Untuk informasi selengkapnya, lihat Indeks sebagian. • Menambahkan dukungan untuk runtime GC pada koleksi dalam `currentOp` perintah. • Menambahkan dukungan indeks teks untuk pencarian teks asli di Amazon DocumentDB. Untuk informasi selengkapnya, lihat Melakukan pencarian teks dengan Amazon DocumentDB. • Ditambahkan dukungan untuk kata kunci `$jsonSchema` skematype,`allOf`,`oneOf`,`anyOf`,,`not`,`maxItems`,`minItems` `ties` ,`minProperties` ,`pattern`,`patternPr` `operties` ,, `multipleOf` `dependencies` , danuniqueItems . Untuk mengetahui informasi selengkapnya, lihat Menggunakan validasi skema JSON. • Ditambahkan dukungan untuk operator aritmatik a`$ceil`,,,`$floor`,`$ln`, `$log`$log10, `$sqrt` dan. `$exp` Untuk mengetahui informasi selengkapnya, lihat Operator aritmatika. • Ditambahkan dukungan untuk operator `$switch` ekspresi kondisional.

Date	Versi minor	Catatan rilis
		• Menambahkan dukungan untuk build indeks `IVFFLAT` vektor paralel. Dokumentasi diperbarui dengan menghapus batasan build indeks `IVFFLAT` vektor paralel dari panduan pengembang. Perbaikan bug dan perubahan lainnya • Menambahkan kembali fungsionalitas case-insensitive untuk pemanggilan. `db.runCommand("dbstats")` Amazon DocumentDB 5.0pelanggan pada versi patch mesin sebelum 3.0.4960 atau 2.0.10221 harus menerapkan tambalan mesin terbaru ini. • Memperbaiki kesalahan saat memanggil `db.coll.stats()` mongo shell versi 1.7 dan yang lebih baru. Dokumentasi diperbarui dengan menghapus tip `db.coll.stats()` pemecahan masalah shell mongo dari panduan pengembang.
November 29, 2023	5.0.0	Fitur baru Amazon DocumentDB 5.0 (Versi Patch Mesin 3.0.3727) Ditambahkan dukungan untuk pencarian vektor. Untuk informasi lebih lanjut, lihat posting blog ini dan kunjungi Pencarian vektor untuk Amazon DocumentDB di Panduan Pengembang Amazon DocumentDB.

Date	Versi minor	Catatan rilis
21 November 2023	5.0.0	Fitur baru Amazon DocumentDB 5.0 (Versi Patch Mesin 3.0.3727) Menambahkan dukungan untuk I/O-optimized penyimpanan. Untuk informasi selengkapnya, lihat Konfigurasi penyimpanan cluster Amazon DocumentDB di Panduan Pengembang Amazon DocumentDB. Menambahkan integrasi untuk pembelajaran mesin tanpa kode dengan SageMaker Canvas. Untuk informasi selengkapnya, lihat No-code pembelajaran mesin dengan Amazon SageMaker AI Canvas di Panduan Pengembang Amazon DocumentDB.

Date	Versi minor	Catatan rilis
6 November 2023	5.0.0	Amazon DocumentDB 5.0 (Versi Patch Mesin 3.0.3727) Fitur baru • Ditambahkan dukungan untuk kata kunci `$jsonSchema` skema`maxLength` ,`minLength` ,`maximum`,`minimum`,,`exclusiveMaximum` , `exclusiveMinimum` `items`, dan`additionalItems` . Perhatikan bahwa validasi skema JSON hanya didukung di cluster berbasis instance. • Menambahkan dukungan untuk operator pipa `$convert` agregasi dan operator turunan singkatan `$toBool`,,,`$toInt`,`$toLong`, `$toDouble` `$toString` `$toDecimal` , `$toObjectId` dan. `$toDate` • Menambahkan dukungan untuk operator ekspresi set`$setDifference` ,`$anyElementTrue` , dan`$allElementTrue` . Perbaikan bug dan perubahan lainnya Memperbaiki masalah saat pembaruan aliran perubahan dari `-NaN` ke `NaN` tidak ditampilkan.
15 September 2023	5.0.0	Fitur baru Amazon DocumentDB 5.0 (Versi Patch Mesin 3.0.3140) Menambahkan dukungan untuk validator skema $JSONSchema hanya dalam cluster berbasis instance. Untuk mengetahui informasi selengkapnya, lihat Menggunakan validasi skema JSON.

Date	Versi minor	Catatan rilis
13 Juli 2023	5.0.0	Fitur baru Amazon DocumentDB 5.0 (Versi Patch Mesin 3.0.1948) • Menambahkan dukungan untuk kompresi dokumen. • Menambahkan dukungan untuk build indeks paralel. • Menambahkan dukungan untuk status build indeks. Perbaikan bug dan perubahan lainnya Amazon DocumentDB 5.0 (Versi Patch Mesin 3.0.1948) • Memperbaiki masalah otentikasi `createCollection` untuk klaster elastis Amazon DocumentDB saat pengguna tidak memiliki akses ke koleksi sistem. • Memperbaiki masalah saat instance wilayah sekunder tidak dapat menggunakan nama instans wilayah primer yang sama.
Juni 7, 2023	5.0.0	Perbaikan bug dan perubahan lainnya Amazon DocumentDB 5.0 • instans r5 dan t3.medium sekarang didukung di Amazon DocumentDB 5.0. • `engineVersion` opsi default ada `5.0.0` di AWS SDK, AWS CLI, dan CloudFormation.

Date	Versi minor	Catatan rilis
10 Mei 2023	5.0.0	Perbaikan bug dan perubahan lainnya Amazon DocumentDB 5.0 (Versi Patch Mesin 3.0.1361) • Ditambahkan dukungan untuk `ignoreunknownindexoptions` dalam createIndex perintah. • Berhenti menambahkan kueri pemantauan internal ke log audit. • Nama variabel lokal yang ditentukan pengguna sekarang mendukung “_” (garis bawah) untuk operator proyeksi seperti `$let` dan. `$filter`

Date	Versi minor	Catatan rilis

1 Maret 2023	5.0.0	Fitur baru
		
		Amazon DocumentDB 5.0 (Versi Patch Mesin 3.0.775)
		
		• Memperkenalkan Amazon DocumentDB 5.0
		  • Kompatibilitas MongoDB 5.0 (dukungan untuk driver MongoDB 5.0 API)
		  • Support untuk Client-side Field Level Encryption (FLE). Anda sekarang dapat mengenkripsi bidang di sisi klien sebelum menulis data ke cluster Amazon DocumentDB. Untuk informasi selengkapnya, lihat enkripsi tingkat Client-side bidang
		  • Operator agregasi baru:`$dateAdd`, `$dateSubtract`
		• Peningkatan batas penyimpanan menjadi 128 TiB untuk semua cluster Amazon DocumentDB berbasis instans dan cluster elastis berbasis shard.
		• Amazon DocumentDB 5.0 sekarang mendukung pemindaian indeks dengan operator `$elemMatch` di tingkat bersarang pertama. Pemindaian indeks didukung ketika kueri hanya memiliki satu tingkat `$elemMatch` filter dan `$elemMatch` kueri bersarang tidak mendukung pemindaian indeks.
		
		Bentuk kueri yang mendukung pemindaian indeks:
		
		`db.foo.find( { "a": {$elemMatch: { "b": "xyz", "c": "abc"} } })`
		
		Bentuk kueri yang tidak mendukung pemindaian indeks:

Date	Versi minor	Catatan rilis
		`db.foo.find( { "a": {$elemMatch: { "b": {$elemMatch: { "d": "xyz", "e": "abc"} }} } })`

Amazon DocumentDB 4.0

Catatan rilis untuk Amazon DocumentDB engine versi 4.0 (kompatibilitas MongoDB 4.0).

Date	Catatan rilis
April 27, 2026	Perbaikan bug dan perubahan lainnya Amazon DocumentDB 4.0 (Versi Mesin: 2.0.14278) Ini adalah patch mesin yang diperlukan yang mengatasi masalah berikut: 1. Memperbaiki masalah yang dapat, dalam kasus yang jarang terjadi, menyebabkan pemulihan point-in-time (PITR) dan peningkatan versi utama (MVU) menjadi tidak responsif. Semua cluster Amazon DocumentDB 4.0 akan secara otomatis ditingkatkan ke versi patch ini selama jendela pemeliharaan berikutnya.
9 Maret 2026	Perbaikan bug dan perubahan lainnya Amazon DocumentDB 4.0 (Versi Mesin: 2.0.14165) Pembaruan ini menyediakan patch mesin opsional dengan perbaikan berikut: 1. Menambahkan DatabaseConnectionsActiveMax metrik untuk memberikan wawasan tentang “Koneksi (aktif)” untuk instance. Batasannya dapat ditemukan di halaman Kuota dan batas Amazon DocumentDB kami.

Date	Catatan rilis
	2. Peningkatan kinerja otentikasi SASL.
Oktober 22, 2025	Fitur baru Amazon DocumentDB 4.0 (Versi Patch Mesin 2.0.11747) Menambahkan bidang baru `collstats` untuk “MVCcidStats” dan “gcRuntimeStats” yang memberikan informasi tentang kesehatan pengumpulan secara keseluruhan sehubungan dengan pengumpulan sampah dan pengumpulan penuaan.
Oktober 16, 2025	Fitur baru Amazon DocumentDB 4.0 (Versi Patch Mesin 2.0.11747) Amazon DocumentDB sekarang mendukung mode dual-stack IPv4/IPv6 () saat cluster Anda berada di cloud pribadi virtual (VPC). Untuk informasi lebih lanjut, lihat posting blog ini dan Amazon VPC dan Amazon DocumentDB di Panduan Pengembang Amazon DocumentDB.

Date	Catatan rilis
Juli 29, 2025	Fitur baru Amazon DocumentDB 4.0 (Versi Patch Mesin 2.0.11747) • Indeks bloat dan metrik bloat berbasis koleksi diluncurkan • CloudWatch Metrik yang ditambahkan `AvailableMVCCIds` dan `LongestActiveGCRuntime` yang memberikan informasi tentang kesehatan klaster secara keseluruhan sehubungan dengan pengumpulan sampah dan pengumpulan penuaan. • Support untuk operator String berikut:`$trim`,`$rtrim`, dan `$ltrim` • Support untuk operator Stage berikut: `$collStats` Perbaikan bug dan perubahan lainnya Amazon DocumentDB 4.0 (Versi Patch Mesin 2.0.11747) Memperbaiki masalah selama pembuatan indeks yang memengaruhi koleksi besar.
Januari 15, 2025	Perbaikan bug dan perubahan lainnya Amazon DocumentDB 4.0 (Versi Patch Mesin 2.0.11096) • `killOp`Perintah tetap untuk menangani kasus khusus sisipan massal. • Peningkatan penggunaan I/O-related memori jaringan pada instans Amazon DocumentDB.
6 November 2024	Fitur baru Amazon DocumentDB 4.0 (Versi Patch Mesin 2.0.10980) • Menambahkan dukungan untuk versi TLS minimum `tls1.2+` dan `tls1.3+` parameter `tls` cluster.

Date	Catatan rilis
September 17, 2024	Perbaikan bug dan perubahan lainnya Amazon DocumentDB 4.0 (Versi Patch Mesin 2.0.10898) • `$setOnInsert` sekarang mendukung `_id` bidang selama penyisipan yang dihasilkan dari operasi pembaruan. • Memperbaiki masalah yang mencegah reklamasi ruang penyimpanan setelah upgrade versi utama ke Amazon DocumentDB 5.0.
Agustus 22, 2024	Fitur baru Amazon DocumentDB 4.0 (Versi Patch Mesin 2.0.5704) Cluster global Amazon DocumentDB sekarang mendukung pengalihan klaster dan failover cluster terkelola. Untuk informasi selengkapnya, lihat Melakukan peralihan untuk klaster global Amazon DocumentDB dan Melakukan failover terkelola untuk klaster global Amazon DocumentDB. **Note** Switchover dan failover cluster global hanya didukung di Amazon DocumentDB 4.0 dan 5.0.

<table>
<tr><th>Date</th><th>Catatan rilis</th></tr>
<tr><td>Juli 23, 2024</td><td>
Fitur baru

Amazon DocumentDB 4.0 (Versi Patch Mesin 2.0.10709)

• Menambahkan dukungan untuk nama indeks yang lebih panjang (hingga 255 karakter). Untuk informasi selengkapnya, lihat Kendala penamaan
• Meningkatkan batas koneksi maksimum dua kali lipat:
<table>
<tr><th>Tipe instans</th><th>Batas asli</th><th>Batas baru</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>Date</th><th>Catatan rilis</th></tr>
<tr><td></td><td>
<table>
<tr><th>Tipe instans</th><th>Batas asli</th><th>Batas baru</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>
Untuk informasi selengkapnya, lihat Batas instans.
</td></tr>
<tr><td>Juli 22, 2024</td><td>Fitur baru

Amazon DocumentDB 4.0 (Versi Patch Mesin 2.0.10593)

• Ditambahkan dukungan DMLAudit penyaringan. Anda sekarang dapat mengatur kondisi filter untuk menyaring log Audit DMLberdas arkan persyaratan spesifiknya alih-alih mencatat setiap kueri DML. Untuk informasi selengkapnya, lihat Memfilter acara audit DML..</td></tr>
<tr><td>9 Juli 2024</td><td>Fitur baru

Untuk Amazon DocumentDB 4.0 (versi patch mesin minimum 2.0.10179), pembaruan sertifikat TLS CA tidak lagi memerlukan reboot sistem. Untuk informasi selengkapnya, lihat Memperbarui sertifikat Amazon DocumentDB TLS Anda.</td></tr>
<tr><td>29 Mei 2024</td><td>Fitur baru

Amazon DocumentDB 4.0 (Versi Patch Mesin 2.0.10593)

• Menambahkan dukungan untuk memastikan presisi penuh dalam log audit saat menangani bilangan bulat besar. Log audit sekarang mempertahankan representasi numerik yang tepat untuk semua angka, mencegah hilangnya presisi.</td></tr>
</table>

Date	Catatan rilis
April 3, 2024	Fitur baru Amazon DocumentDB 4.0 (Versi Patch Mesin 2.0.10392) • Menambahkan dukungan untuk `bypassDocumentValidation` untuk`$jsonSchema` . Untuk informasi selengkapnya tentang `bypassDocumentValidation` , lihat memintas DocumentValidation. • Menambahkan dukungan dari`$expr`. • Menambahkan dukungan untuk Uncorrelated Joins in. `$lookup` • Menambahkan dukungan untuk mempertahankan aturan validasi dalam tahap `$out` agregasi.
10 Januari 2024	Fitur baru Amazon DocumentDB 4.0 (Versi Patch Mesin 2.0.10124, 2.0.10179, 2.0.10221) • Menambahkan dukungan untuk runtime GC pada koleksi dalam `currentOp` perintah. • Ditambahkan dukungan untuk kata kunci `$jsonSchema` skematype,`allOf`,`oneOf`,`anyOf`,,`not`,`maxItems`,`minItems`,`maxProperties` ,`minProperties` ,`pattern`,`patternProperties` ,, `multipleOf` `dependencies` , danuniqueItems . Untuk mengetahui informasi selengkapnya, lihat Menggunakan validasi skema JSON. • Ditambahkan dukungan untuk operator aritmatik a`$ceil`,,,`$floor`,`$ln`, `$log`$log10, `$sqrt` dan. `$exp` Untuk mengetahui informasi selengkapnya, lihat Operator aritmatika. • Ditambahkan dukungan untuk operator `$switch` ekspresi kondisional.

Date	Catatan rilis
20 Desember 2023	Perubahan lainnya Dukungan yang diaktifkan untuk peningkatan versi utama di tempat di Amazon DocumentDB 4.0. Untuk informasi selengkapnya, lihat Peningkatan versi utama Amazon DocumentDB di tempat.
6 November 2023	Amazon DocumentDB 4.0 (Versi Patch Mesin 2.0.9876) Fitur baru • Ditambahkan dukungan untuk kata kunci `$jsonSchema` skema`maxLength` ,`minLength` ,`maximum`,`minimum`,,`exclusiveMaximum` , `exclusiveMinimum` `items`, dan`additionalItems` . Perhatikan bahwa validasi skema JSON hanya didukung di cluster berbasis instance. • Menambahkan dukungan untuk operator pipa `$convert` agregasi dan operator turunan singkatan`$toBool`,,,`$toInt`,`$toLong`, `$toDouble` `$toString` `$toDecimal` , `$toObjectId` dan. `$toDate` • Menambahkan dukungan untuk operator ekspresi set`$setDifference` ,`$anyElementTrue` , dan`$allElementTrue` . Perbaikan bug dan perubahan lainnya Memperbaiki masalah saat pembaruan aliran perubahan dari `-NaN` ke `NaN` tidak ditampilkan.
20 September 2023	Fitur baru Menambahkan dukungan untuk peningkatan versi utama di tempat di Amazon DocumentDB 4.0. Untuk mengetahui informasi selengkapnya, lihat Peningkatan versi utama Amazon DocumentDB di tempat.

Date	Catatan rilis
15 September 2023	Fitur baru Amazon DocumentDB 4.0 (Versi Patch Mesin 2.0.9686) Menambahkan dukungan untuk validator skema $JSONSchema hanya dalam cluster berbasis instance. Untuk mengetahui informasi selengkapnya, lihat Menggunakan validasi skema JSON.
13 Juli 2023	Fitur baru Amazon DocumentDB 4.0 (Versi Patch Mesin 2.0.9259) • Menambahkan dukungan untuk build indeks paralel. Perbaikan bug dan perubahan lainnya Amazon DocumentDB 4.0 (Versi Patch Mesin 2.0.9259) • Berhenti menambahkan kueri pemantauan internal ke log audit.

Date	Catatan rilis
4 April 2023	Perbaikan bug dan perubahan lainnya Amazon DocumentDB 4.0 (Versi Patch Mesin 2.0.8934) • Memperbaiki masalah dengan audit DHTML saat diaktifkan selama beban kerja yang sedang berlangsung. • Memperbaiki masalah dengan audit DHTML saat perintah agregat dengan petunjuk diteruskan nilai string. • Memperbaiki masalah dengan `listCollections` perintah yang tidak berfungsi saat pengguna dengan peran readwriteanydatabase memiliki opsi AuthorizedCollections dan NameOnly disetel ke true. • Memperbaiki masalah untuk mengurai string numerik dengan benar dalam nama bidang. • Batalkan kursor yang berjalan lama saat berdampak pada pengumpulan sampah. • Nama variabel lokal yang ditentukan pengguna sekarang mendukung “_” (garis bawah) untuk operator proyeksi seperti `$let` dan. `$filter`
27 Februari 2023	Perbaikan bug dan perubahan lainnya Amazon DocumentDB 4.0 Menambahkan dukungan untuk AWS Lambda. Untuk informasi selengkapnya, lihat Menggunakan AWS Lambda dengan Mengubah Aliran.

Date	Catatan rilis
Agustus 9, 2022	Fitur baru 4.0 • Ditambahkan dukungan untuk tipe data Decimal128. Decimal128 adalah tipe data BSON yang didukung di semua wilayah di mana DocumentDB tersedia. Untuk informasi selengkapnya, lihat jenis data DynamoDB. • Menambahkan dukungan untuk audit kueri DHTML dengan Amazon CloudWatch Logs. Sekarang Amazon DocumentDB dapat merekam peristiwa Data Manipulation Language (DHTML) dan peristiwa Data Definition Language (DDL) ke Amazon Logs. CloudWatch Untuk informasi lebih lanjut, lihat posting blog ini. Perbaikan bug dan perubahan lainnya 4.0 • Anda sekarang dapat mengubah passwoprd Anda sendiri dengan kata sandi sendiri dengan hak istimewa. `changeOwnPassword`
25 Juli 2022	Fitur baru Amazon DocumentDB 4.0 Anda sekarang dapat membuat cluster lebih cepat dengan kemampuan untuk membuat klon yang menggunakan volume cluster DocumentDB yang sama dan memiliki data yang sama dengan cluster asli. Untuk detailnya, lihat Mengelola Cluster Amazon DocumentDB.

Date	Catatan rilis
Juni 27, 2022	Fitur baru Amazon DocumentDB 4.0 (Versi Patch Mesin 2.0.7509) Amazon DocumentDB secara dinamis mengubah ukuran database Anda berdasarkan pola penggunaan. Menambahkan lebih banyak data meningkatkan ruang hingga 64 Tebibytes (TiB) dan menghapus data mengurangi ruang yang dialokasikan.
7 April 2022	Fitur baru 4.0 (Versi Patch Mesin 2.0.6142 dan 2.0.6948) Amazon DocumentDB Performance Insights sekarang dalam pratinjau . Anda sekarang dapat menyimpan tujuh hari riwayat kinerja di jendela bergulir tanpa biaya tambahan. Untuk informasi selengkapnya, lihat Monitoring with Performance Insights.

24 Januari 2022	Fitur baru
	
	4.0 (Versi Patch Mesin 2.0.5170)
	
	• Amazon DocumentDB sekarang menawarkan uji coba gratis. Untuk detailnya, lihat halaman uji coba gratis Amazon DocumentDB.
	• Anda sekarang dapat menggunakan fitur yang disempurnakan dengan kueri Geospasial, termasuk API berikut:
	  • `$geoWithin`
	  • `$geoIntersects`
	• Ditambahkan dukungan untuk operator MongoDB berikut:
	  • `$mergeObjects`
	  • `$reduce`
	
	Untuk informasi selengkapnya, lihat Menanyakan data Geospasial dengan Amazon DocumentDB.

Date	Catatan rilis

Januari 21, 2022	Fitur baru
	
	Amazon DocumentDB 4.0 (Versi Patch Mesin 2.0.5706)
	
	• Amazon DocumentDB Graviton2 (r6g.large, r6g.2xlarge, r6g.4xlarge, r6g.8xlarge, r6g.12xlarge, r6g.16xlarge dan t4g.medium) instans sekarang didukung.
	
	Fitur baru
	
	4.0 (Versi Patch Mesin 2.0.5706)
	
	• Menambahkan dukungan untuk API MongoDB berikut:
	  • `$reduce`
	  • `$mergeObjects`
	  • `$geoWithin`
	  • `$geoIntersects`

Date	Catatan rilis

25 Oktober 2021	Fitur baru
	
	4.0 (Versi Patch Mesin 2.0.5704)
	
	• Menambahkan dukungan untuk API MongoDB berikut
	  • `$literal`
	  • `$map`
	  • `$$ROOT`
	• Support untuk kemampuan GeoSpatial Query. Lihat posting blog ini untuk lebih jelasnya
	• Support untuk kontrol akses dengan peran yang ditentukan pengguna. Lihat posting blog ini untuk lebih jelasnya
	• Driver Amazon DocumentDB JDBC untuk mengaktifkan konektivitas dari alat BI seperti Tableau dan alat kueri seperti SQL Workbench
	
	Perbaikan bug dan perubahan lainnya
	
	4.0 (Versi Patch Mesin 2.0.5704)
	
	• Perbaikan bug `$natural` untuk mengurutkan dengan benar ketika eksplisit `.sort()` hadir bersama dengan `$natural`
	• Perbaikan bug untuk aliran perubahan agar berfungsi `$redact`
	• Perbaikan bug `$ifNull` untuk bekerja dengan array kosong
	• Perbaikan bug untuk consumption/server kerusakan sumber daya yang berlebihan saat pengguna yang saat ini masuk dihapus atau hak istimewa pengguna untuk aktivitas yang sedang berlangsung dicabut
	• Perbaikan bug `listDatabase` dan pemeriksaan `listCollection` hak istimewa
	• Bug Fix dedupe logika untuk elemen multi-kunci

Date	Catatan rilis
24 Juni 2021	Fitur baru 4.0 (Versi Patch Mesin 2.0.3371) • instans r5.8xlarge dan r5.16xlarge sekarang didukung. Pelajari lebih lanjut di posting blog Amazon DocumentDB Sekarang Mendukung Instans r5.8xlarge dan r5.16xlarge. • Cluster global sekarang didukung untuk memberikan pemulihan bencana dari pemadaman di seluruh wilayah dan memungkin kan pembacaan global latensi rendah dengan memungkinkan pembacaan dari cluster Amazon DocumentDB terdekat. Perhatika n bahwa klaster global saat ini tidak didukung di Wilayah Amerika Selatan (São Paulo), Eropa (Milan), Tiongkok (Beijing), dan Tiongkok (Ningxia).

Date	Catatan rilis
4 Mei 2021	Fitur baru 4.0 (Versi Patch Mesin 2.0.3371) • `renameCollection` • `$zip` • `$indexOfArray` • `$reverseArray` • `$natural` • `$hint`dukungan untuk pembaruan • Pemindaian indeks untuk `distinct` Perbaikan bug dan perubahan lainnya 4.0 (Versi Patch Mesin 2.0.3371) • Mengurangi penggunaan memori untuk `$in` kueri • Memperbaiki kebocoran memori dalam indeks multikey • Memperbaiki rencana penjelasan dan keluaran profiler untuk `$out` • Menambahkan batas waktu untuk operasi dari sistem pemantauan internal untuk meningkatkan keandalan • Memperbaiki cacat yang memengaruhi predikat kueri yang diteruska n ke indeks multikey
15 Januari 2021	Fitur baru Amazon DocumentDB 4.0 (Versi Patch Mesin 2.0.722) • Tidak ada

Date	Catatan rilis
9 November 2020	Fitur baru Amazon DocumentDB 4.0 (Versi Patch Mesin 2.0.722) • Kompatibilitas MongoDB 4.0 • Transaksi ACID • Support untuk `cluster(client.watch()` atau `mongo.watch())` dan aliran `(db.watch())` perubahan tingkat database • Kemampuan untuk memulai atau melanjutkan aliran perubahan menggunakan `startAtOperationTime` • Perpanjang periode retensi aliran perubahan Anda menjadi 7 hari (sebelumnya 24 jam) • AWS DMS target untuk Amazon DocumentDB 4.0 • CloudWatch metrik:`TransactionsOpen` ,,`TransactionsOpenMax` , `TransactionsAborted` `TransactionsStarted` , dan `TransactionsCommitted` • Bidang baru untuk transaksi di`currentOp` ,`ServerStatus` , dan`profiler`. • Kemampuan untuk menggunakan indeks dengan tahap `$lookup` agregasi • `find()`kueri dengan proyeksi dapat disajikan arah dari indeks (kueri tertutup) • Kemampuan untuk digunakan `hint()` dengan `findAndModify` • Optimalisasi kinerja untuk operator `$addToSet` • Perbaikan untuk mengurangi ukuran indeks secara keseluruhan. • Operator agregasi baru:`$ifNull`,`$replaceRoot` ,,`$setIsSubset` , `$setIntersection` `$setUnion` , dan `$setEquals` • Dengan `ListDatabase` perintah `ListCollection` dan, Anda sekarang dapat secara opsional menggunakan `authorizedDatabases` parameter `authorizedCollections` dan untuk memungkinkan pengguna untuk membuat daftar koleksi dan database yang mereka memiliki izin untuk mengakses tanpa

Date	Catatan rilis
	memerlukan `listDatabase` peran `listCollections` dan, masing-masing • Pengguna juga dapat mengakhiri kursor mereka sendiri tanpa memerlukan peran `KillCursor` • Membandingkan jenis numerik subdokumen sekarang konsisten dengan membandingkan jenis numerik dokumen tingkat pertama. Perilaku di Amazon DocumentDB 4.0 sekarang kompatibel dengan MongoDB. Perbaikan bug dan perubahan lainnya Amazon DocumentDB 4.0 (Versi Patch Mesin 2.0.722) • `$setOnInsert` tidak lagi mengizinkan pembaruan saat menggunakan operator $ posisi. Perilaku di Amazon DocumentDB 4.0 sekarang kompatibel dengan MongoDB. • Memperbaiki masalah dengan `$createCollection` dan mengatur `autoIndexId` • Proyeksi untuk dokumen bersarang • Pengaturan default yang diubah untuk memori kerja agar diskalakan dengan ukuran memori instans • Perbaikan pengumpulan sampah • Pencarian dengan kunci kosong di jalur, perbedaan perilaku dengan mongo • Memperbaiki `dateToString` bug dalam perilaku zona waktu • Tetap `$push` (agregasi) untuk menghormati urutan pengurutan • Memperbaiki bug `$currentOp` dengan agregat • Memperbaiki masalah dengan `readPreference` di sekunder • Masalah tetap dengan memvalidasi `$createIndex` adalah database yang sama dengan perintah yang dikeluarkan • Memperbaiki perilaku yang tidak konsisten untuk`minKey`, `maxKey` pencarian gagal

Date	Catatan rilis
	• Memperbaiki masalah dengan `$size` operator yang tidak bekerja dengan array komposit • Memperbaiki masalah dengan negasi `$in` dengan regex • Memperbaiki masalah dengan `$distinct` perintah yang dijalankan terhadap tampilan • Memperbaiki masalah dengan agregasi dan menemukan perintah yang menyortir bidang yang hilang secara berbeda • Tetap `$eq` ke ekspresi reguler yang tidak memeriksa jenis • Memperbaiki `$currentDate` bug dalam perilaku posisi ordinal stempel waktu • Granularitas milidetik tetap untuk `$currentDate`

Amazon DocumentDB 3.6

Catatan rilis untuk Amazon DocumentDB engine versi 3.6 (MongoDB 3.6 kompatibilitas).

Date	Catatan rilis
April 27, 2026	Fitur baru Amazon DocumentDB 3.6 (Versi Mesin: 1.0.208692) [sedang berlangsung] Pembaruan ini menyediakan patch mesin opsional dengan perbaikan berikut: 1. Menambahkan dukungan untuk `startAtOperationTime` aliran perubahan.
Februari 12, 2026	Perbaikan bug dan perubahan lainnya Amazon DocumentDB 3.6 (Versi Mesin: 1.0.208675) 1. Menambahkan DatabaseConnectionsActiveMax metrik untuk memberikan wawasan tentang “Koneksi (aktif)” untuk instance.

Date	Catatan rilis
	Batasannya dapat ditemukan di halaman Kuota dan batas Amazon DocumentDB kami. 2. Peningkatan kinerja otentikasi SASL. 3. Perbaikan bug untuk perintah $indexStats.
Agustus 20, 2024	Fitur baru Untuk Amazon DocumentDB 3.6 (versi patch mesin minimum 1.0.208662), pembaruan sertifikat TLS CA tidak lagi memerlukan reboot sistem. Untuk informasi selengkapnya, lihat Memperbarui sertifikat Amazon DocumentDB TLS Anda.
Juli 22, 2024	Fitur baru Amazon DocumentDB 3.6 (Versi Patch Mesin 1.0.208662) Menghapus batasan indeks pada instance db.r5.* dan db.r6.* di Amazon DocumentDB MVU. Untuk informasi selengkapnya, lihat Prasyarat. Perbaikan bug dan perubahan lainnya Amazon DocumentDB 3.6 (Versi Patch Mesin 1.0.208662) Amazon DocumentDB sekarang mengenali -NaN sebagai token JSON yang valid.
20 Desember 2023	Perubahan lainnya Dukungan yang diaktifkan untuk peningkatan versi utama di tempat di Amazon DocumentDB 3.6. Untuk informasi selengkapnya, lihat Peningkatan versi utama Amazon DocumentDB di tempat.

Date	Catatan rilis
17 November 2023	Perbaikan bug dan perubahan lainnya Amazon DocumentDB 3.6 (Versi Patch Mesin 1.0.208570) Nama variabel lokal yang ditentukan pengguna sekarang mendukung "_" (garis bawah) untuk operator proyeksi seperti `$let` dan. `$filter`
20 September 2023	Fitur baru Menambahkan dukungan untuk upgrade versi utama di tempat di Amazon DocumentDB 3.6. Untuk mengetahui informasi selengkapnya, lihat Peningkatan versi utama Amazon DocumentDB di tempat.
2 Februari 2023	Perbaikan bug dan perubahan lainnya Amazon DocumentDB 3.6 (Versi Patch Mesin 1.0.208432) • Memperbaiki masalah dengan audit DHTML saat diaktifkan selama beban kerja yang sedang berlangsung. • Memperbaiki masalah dengan audit DHTML saat perintah agregat dengan petunjuk diteruskan nilai string. • Memperbaiki masalah dengan `listCollections` perintah yang tidak berfungsi saat pengguna dengan peran readwriteanydatabase memiliki opsi AuthorizedCollections dan NameOnly disetel ke true. • Memperbaiki masalah untuk mengurai string numerik dengan benar dalam nama bidang. • Batalkan kursor yang berjalan lama saat berdampak pada pengumpulan sampah.

Date	Catatan rilis
Agustus 9, 2022	Fitur baru Amazon DocumentDB 3.6 (Versi Patch Mesin 1.0.208152) • Ditambahkan dukungan untuk tipe data Decimal128. Decimal128 adalah tipe data BSON yang didukung di semua wilayah di mana DocumentDB tersedia. Untuk informasi selengkapnya, lihat jenis data DynamoDB. • Menambahkan dukungan untuk audit kueri DHTML dengan Amazon CloudWatch Logs. Sekarang Amazon DocumentDB dapat merekam peristiwa Data Manipulation Language (DHTML) dan peristiwa Data Definition Language (DDL) ke Amazon Logs. CloudWatch Untuk informasi lebih lanjut, lihat posting blog ini. Perbaikan bug dan perubahan lainnya Amazon DocumentDB 3.6 (Versi Patch Mesin 1.0.208152) • Anda sekarang dapat mengubah passwoprd Anda sendiri dengan kata sandi sendiri dengan hak istimewa. `changeOwnPassword`
7 April 2022	Fitur baru Amazon DocumentDB 3.6 (Versi Patch Mesin 1.0.207836 dan 1.0.208015) Amazon DocumentDB Performance Insights sekarang dalam pratinjau . Anda sekarang dapat menyimpan tujuh hari riwayat kinerja di jendela bergulir tanpa biaya tambahan. Untuk informasi selengkapnya, lihat Monitoring with Performance Insights.

Date	Catatan rilis

24 Januari 2022	Fitur baru
	
	Amazon DocumentDB 3.6 (Versi Patch Mesin 1.0.207684)
	
	• Amazon DocumentDB sekarang menawarkan uji coba gratis. Untuk detailnya, lihat halaman uji coba gratis Amazon DocumentDB.
	• Anda sekarang dapat menggunakan fitur yang disempurnakan dengan kueri Geospasial, termasuk API berikut:
	  • `$geoWithin`
	  • `$geoIntersects`
	• Ditambahkan dukungan untuk operator MongoDB berikut:
	  • `$mergeObjects`
	  • `$reduce`
	
	Untuk informasi selengkapnya, lihat Menanyakan data Geospasial dengan Amazon DocumentDB.

Januari 21, 2022	Fitur baru
	
	Amazon DocumentDB 3.6 (Versi Patch Mesin 1.0.207781)
	
	• Menambahkan dukungan untuk API MongoDB berikut:
	  • `$reduce`
	  • `$mergeObjects`
	  • `$geoWithin`
	  • `$geoIntersects`

Date	Catatan rilis

25 Oktober 2021	Fitur baru
	
	Amazon DocumentDB 3.6 (Versi Patch Mesin 1.0.207780)
	
	• Menambahkan dukungan untuk API MongoDB berikut
	  • `$literal`
	  • `$map`
	  • `$$ROOT`
	• Support untuk kemampuan GeoSpatial Query. Lihat posting blog ini untuk lebih jelasnya
	• Support untuk kontrol akses dengan peran yang ditentukan pengguna. Lihat posting blog ini untuk lebih jelasnya
	• Driver Amazon DocumentDB JDBC untuk mengaktifkan konektivitas dari alat BI seperti Tableau dan alat kueri seperti SQL Workbench
	
	Perbaikan bug dan perubahan lainnya
	
	Amazon DocumentDB 3.6 (Versi Patch Mesin 1.0.207780)
	
	• Perbaikan bug `$natural` untuk mengurutkan dengan benar ketika eksplisit `.sort()` hadir bersama dengan `$natural`
	• Perbaikan bug untuk aliran perubahan agar berfungsi `$redact`
	• Perbaikan bug `$ifNull` untuk bekerja dengan array kosong
	• Perbaikan bug untuk consumption/server kerusakan sumber daya yang berlebihan saat pengguna yang saat ini masuk dihapus atau hak istimewa pengguna untuk aktivitas yang sedang berlangsung dicabut
	• Perbaikan bug `listDatabase` dan pemeriksaan `listCollection` hak istimewa
	• Bug Fix dedupe logika untuk elemen multi-kunci

Date	Catatan rilis
24 Juni 2021	Fitur baru Amazon DocumentDB 3.6 (Versi Patch Mesin 1.0.207117) • instans r5.8xlarge dan r5.16xlarge sekarang didukung. Pelajari lebih lanjut di posting blog Amazon DocumentDB Sekarang Mendukung Instans r5.8xlarge dan r5.16xlarge. • Cluster global sekarang didukung untuk memberikan pemulihan bencana dari pemadaman di seluruh wilayah dan memungkin kan pembacaan global latensi rendah dengan memungkinkan pembacaan dari cluster Amazon DocumentDB terdekat. Perhatika n bahwa klaster global saat ini tidak didukung di Wilayah Amerika Selatan (São Paulo), Eropa (Milan), Tiongkok (Beijing), dan Tiongkok (Ningxia).

Date	Catatan rilis
4 Mei 2021	Fitur baru Amazon DocumentDB 3.6 (Versi Patch Mesin 1.0.207117) • `renameCollection` • `$zip` • `$indexOfArray` • `$reverseArray` • `$natural` • `$hint`dukungan untuk pembaruan • Pemindaian indeks untuk `distinct` Perbaikan bug dan perubahan lainnya Amazon DocumentDB 3.6 (Versi Patch Mesin 1.0.207117) • Mengurangi penggunaan memori untuk `$in` kueri • Memperbaiki kebocoran memori dalam indeks multikey • Memperbaiki rencana penjelasan dan keluaran profiler untuk `$out` • Menambahkan batas waktu untuk operasi dari sistem pemantauan internal untuk meningkatkan keandalan • Memperbaiki cacat yang memengaruhi predikat kueri yang diteruska n ke indeks multikey

Date	Catatan rilis
15 Januari 2021	Fitur baru Amazon DocumentDB 3.6 (Versi Patch Mesin 1.0.206295) • Kemampuan untuk menggunakan indeks dengan tahap `$lookup` agregasi • `find()`kueri dengan proyeksi dapat disajikan arah dari indeks (kueri tertutup) • Kemampuan untuk digunakan `hint()` dengan `findAndModify` • Optimalisasi kinerja untuk operator `$addToSet` • Perbaikan untuk mengurangi ukuran indeks keseluruhan • Operator agregasi baru:`$ifNull`,`$replaceRoot` ,,`$setIsSubset` , `$setInstersection` `$setUnion` , dan `$setEquals` • Pengguna juga dapat mengakhiri kursor mereka sendiri tanpa memerlukan peran `KillCursor`
9 November 2020	Fitur baru Amazon DocumentDB 3.6 (Versi Patch Mesin 1.0.206295) • Tidak ada Perbaikan bug dan perubahan lainnya Amazon DocumentDB 3.6 (Versi Patch Mesin 1.0.206295) • Tidak ada

Date	Catatan rilis
30 Oktober 2020	Fitur baru Amazon DocumentDB 3.6 (Versi Patch Mesin 1.0.206295) • Ditambahkan kemampuan untuk membuka kursor aliran perubahan di tingkat cluster (`client.watch()` atau `mongo.watch()`) dan database (`db.watch()`) • Kemampuan untuk meningkatkan periode retensi aliran perubahan menjadi 7 hari (sebelumnya 24 jam) Perbaikan bug dan perubahan lainnya Amazon DocumentDB 3.6 (Versi Patch Mesin 1.0.206295) • Berbagai peningkatan kinerja kasus umum • Peningkatan keamanan yang ditargetkan • Memperbaiki masalah dengan skip sort pada bidang kedua dari indeks gabungan • Aktifkan indeks reguler untuk kesetaraan pada bidang tunggal indeks multi-kunci (bukan gabungan) • Kondisi balapan otentikasi tetap • Memperbaiki masalah yang menyebabkan kecelakaan pengumpul an sampah yang jarang terjadi • Peningkatan keamanan RBAC • Ditambahkan `databaseConnectionsMax` metrik • Peningkatan kinerja untuk beban kerja tertentu pada instans `r5.24xlarge`

Date	Catatan rilis
22 September 2020	Fitur baru Amazon DocumentDB 3.6 (Versi Patch Mesin 1.0.206295) • `$outtahap` agregasi • Meningkatkan jumlah maksimum koneksi dan kursor per instance sebanyak 10x Perbaikan bug dan perubahan lainnya Amazon DocumentDB 3.6 (Versi Patch Mesin 1.0.206295) • Tidak ada
10 Juli 2020	Fitur baru Amazon DocumentDB 3.6 (Versi Patch Mesin 1.0.206295) • Salinan Cuplikan Lintas Wilayah Perbaikan bug dan perubahan lainnya Amazon DocumentDB 3.6 (Versi Patch Mesin 1.0.206295) • Tidak ada

Date	Catatan rilis
30 Juni 2020	Fitur baru Amazon DocumentDB 3.6 (Versi Patch Mesin 1.0.206295) • Contoh menengah T3 Perbaikan bug dan perubahan lainnya Amazon DocumentDB 3.6 (Versi Patch Mesin 1.0.206295) • Reklamasi memori idle untuk instance t3 • Peningkatan otentikasi • Peningkatan kinerja otentikasi SASL • Memperbaiki `currentOp` masalah saat melebihi operasi maksimum yang mungkin • `killOps`Masalah tetap untuk pembaruan massal dan hapus • Perbaikan `$sample` kinerja dengan `$match` • Dukungan tetap untuk $$ kasus cond dalam tahap redact • Memperbaiki berbagai penyebab crash root yang berulang • Perbaikan penyapuan TTL untuk mengurangi iOS dan latensi • Pemanfaatan memori yang dioptimalkan untuk `$unwind` • Kondisi balapan statistik koleksi tetap dengan indeks drop • Kondisi balapan tetap selama pembuatan indeks bersamaan • Memperbaiki crash yang jarang terjadi `hash_search` dalam indeks

Catatan rilis umum Amazon DocumentDB

Pembaruan layanan umum termasuk ketersediaan wilayah, integrasi perkakas, dan fitur lintas versi.

Date	Catatan rilis
April 29, 2026	Fitur baru Amazon DocumentDB sekarang tersedia di wilayah Kanada Barat (Calgary). Untuk informasi lebih lanjut, lihat posting blog ini.
April 17, 2026	Fitur baru Penskalaan yang lebih cepat pada DocumentDB tanpa server sekarang tersedia.
Februari 11, 2026	Fitur baru Amazon DocumentDB sekarang tersedia di wilayah Asia Pasifik (Melbourne). Untuk informasi lebih lanjut, lihat posting blog ini. Amazon DocumentDB sekarang tersedia di wilayah Eropa (Zurich). Untuk informasi lebih lanjut, lihat posting blog ini.
Januari 08, 2026	Fitur baru Amazon DocumentDB sekarang tersedia di wilayah Asia Pasifik (Jakarta) . Untuk informasi lebih lanjut, lihat posting blog ini.
Oktober 13, 2025	Fitur baru Amazon DocumentDB sekarang mendukung instans R6G dan T4G di wilayah Asia Pasifik (Hyderabad). Untuk informasi selengkapnya, lihat Kelas instance yang didukung berdasarkan wilayah.
Oktober 7, 2025	Fitur baru Amazon DocumentDB sekarang tersedia di wilayah berikut: • Asia Pasifik (Malaysia) • Asia Pasifik (Osaka) • Asia Pasifik (Thailand)

Date	Catatan rilis
	• Meksiko (Tengah) Untuk informasi lebih lanjut, lihat posting blog ini.
September 26, 2025	Fitur baru Amazon DocumentDB sekarang mendukung salinan snapshot lintas wilayah dan lintas akun dan berbagi snapshot. Untuk informasi selengkapnya, lihat Menyalin snapshot di seluruh Wilayah AWS dan Berbagi snapshot cluster Amazon DocumentDB.
Juni 18, 2025	Fitur baru Amazon DocumentDB sekarang tersedia di wilayah Israel (Tel Aviv).
8 Mei 2025	Fitur baru Amazon DocumentDB sekarang tersedia di wilayah Eropa (Stockholm). Untuk informasi lebih lanjut, lihat posting blog ini.
April 2, 2025	Perbaikan bug dan perubahan lainnya Memperbaiki bug dalam pembuatan indeks vektor dengan indeks latar belakang.
Februari 6, 2025	Fitur baru Amazon DocumentDB sekarang terintegrasi dengan. AWS Toolkit for Visual Studio CodeUntuk informasi selengkapnya, lihat posting blog Apa yang Baru ini, dan lihat Bekerja dengan Amazon DocumentDB di Toolkit di Panduan Pengguna.AWS Toolkit for Visual Studio Code

Date	Catatan rilis
Oktober 22, 2024	Fitur baru Amazon DocumentDB sekarang mendukung metrik throughput `StorageNetworkReceiveThroughput` jaringan penyimpan an,, dan. `StorageNetworkTransmitThroughput` `StorageNe tworkThroughput` Untuk informasi selengkapnya, lihat Mengevalu asi penggunaan instans Amazon DocumentDB dengan metrik CloudWatch.
September 18, 2024	Fitur baru Amazon DocumentDB sekarang tersedia di wilayah Afrika (Cape Town). Untuk informasi lebih lanjut, lihat posting blog ini. Amazon DocumentDB sekarang tersedia di wilayah Eropa (Spanyol). Untuk informasi lebih lanjut, lihat posting blog ini.
Juni 25, 2024	Fitur baru Otentikasi dengan ARN AWS IAM tersedia di klaster 5.0 berbasis instans Amazon DocumentDB di semua wilayah yang didukung. Untuk informasi selengkapnya, lihat Otentikasi menggunakan identitas IAM.
April 3, 2024	Fitur baru Amazon DocumentDB sekarang tersedia di wilayah Timur Tengah (UEA). Untuk informasi lebih lanjut, lihat posting blog ini. Perbaikan bug dan perubahan lainnya • Memperbaiki kesalahan saat memanggil `db.coll.stats()` mongo shell versi 1.7 dan yang lebih baru. • Memperbaiki masalah kebocoran memori untuk kueri aliran perubahan yang berisi `$regex` sebagai bagian dari pipeline agregasi yang sama.

Date	Catatan rilis
Januari 30, 2024	Fitur baru Cluster global Amazon DocumentDB Cluster global sekarang tersedia di kedua AWS GovCloud (US) wilayah: AWS GovCloud (US-East) dan AWS GovCloud (US-West).
13 Desember 2023	Fitur baru Menambahkan dukungan untuk konektivitas EC2 1-klik. Untuk informasi selengkapnya, lihat Connect menggunakan Amazon EC2.
17 November 2023	Fitur baru Amazon DocumentDB sekarang tersedia di wilayah () AWS GovCloud . US-East Untuk informasi lebih lanjut, lihat posting blog ini.
25 September 2023	Fitur baru Amazon DocumentDB sekarang tersedia di wilayah Asia Pasifik (Hong Kong). Untuk informasi lebih lanjut, lihat posting blog ini.
11 September 2023	Fitur baru Amazon DocumentDB sekarang tersedia di wilayah Asia Pasifik (Hyderabad). Untuk informasi lebih lanjut, lihat posting blog ini.
29 April 2022	Fitur baru Amazon DocumentDB sekarang tersedia di wilayah Tiongkok (Beijing). Untuk informasi lebih lanjut, lihat posting blog ini.
16 Maret 2022	Fitur baru Amazon DocumentDB sekarang tersedia di wilayah Eropa (Milan). Untuk informasi lebih lanjut, lihat posting blog ini.

Date	Catatan rilis
8 Februari 2022	Fitur baru Instans Amazon DocumentDB R6G dan T4G sekarang tersedia di Asia Pasifik, Amerika Selatan, dan Eropa. Untuk informasi lebih lanjut, lihat posting blog ini.
4 Mei 2021	Fitur baru Lihat semua fitur baru di posting blog ini.
9 November 2020	Fitur baru Lihat semua fitur baru di posting blog ini.
30 Oktober 2020	Fitur baru Lihat semua fitur baru di posting blog ini.
22 September 2020	Fitur baru Lihat semua fitur baru di posting blog ini.
10 Juli 2020	Fitur baru Lihat semua fitur baru di posting blog ini.
30 Juni 2020	Fitur baru Lihat semua fitur baru di posting blog ini.

Catatan rilis cluster elastis Amazon DocumentDB

Catatan rilis untuk cluster elastis Amazon DocumentDB.

Date	Catatan rilis
Januari 28, 2025	Fitur baru

Date	Catatan rilis
	Untuk konektivitas sekali klik, Amazon DocumentDB sekarang AWS CloudShell terintegrasi dengan cluster berbasis instance dan elastis. Untuk informasi lebih lanjut, lihat salah satu atau semua hal berikut: • Memulai Amazon DocumentDB • Memulai dengan cluster elastis Amazon DocumentDB • Apa posting blog baru • Posting blog cara teknis
November 12, 2024	Fitur baru Cluster elastis Amazon DocumentDB sekarang mendukung Indeks Latar Belakang.
November 1, 2024	Fitur baru Amazon DocumentDB sekarang mendukung tindakan pemeliharaan klaster elastis. Untuk informasi selengkapnya, lihat Mempertahankan cluster elastis Amazon DocumentDB.
Agustus 8, 2024	Fitur baru Cluster elastis Amazon DocumentDB sekarang tersedia di wilayah Asia Pasifik (Hong Kong), Kanada (Tengah), dan Eropa (Paris). Untuk informasi selengkapnya, lihat Ketersediaan wilayah untuk cluster elastis.
Juli 8, 2024	Fitur baru Cluster elastis Amazon DocumentDB sekarang tersedia di wilayah Eropa (Milan). Untuk informasi selengkapnya, lihat Ketersediaan wilayah untuk cluster elastis.

Date	Catatan rilis
Februari 22, 2024	Fitur baru Cluster elastis Amazon DocumentDB Cluster elastis Amazon DocumentDB sekarang mendukung fitur-fitur berikut: • Replika instance pecahan sekunder yang dapat dibaca - untuk informasi lebih lanjut, lihat langkah 5b dari. Langkah 1: Buat cluster elastis • Start/stop cluster - untuk informasi lebih lanjut, lihatMenghentikan dan memulai cluster elastis Amazon DocumentDB. • Instans pecahan yang dapat dikonfigurasi - untuk informasi lebih lanjut, lihat langkah 5b dari. Langkah 1: Buat cluster elastis • Pencadangan otomatis untuk snapshot - untuk informasi lebih lanjut, lihat. Mengelola cadangan otomatis snapshot cluster elastis • Salin snapshot - untuk informasi lebih lanjut, lihatMenyalin snapshot cluster elastis.
Januari 30, 2024	Fitur baru Cluster elastis Amazon DocumentDB Cluster elastis Amazon DocumentDB sekarang tersedia di wilayah berikut: • Asia Pasifik (Mumbai) • Asia Pasifik (Seoul) • Amerika Selatan (Sao Paulo) • Eropa (London) Untuk informasi selengkapnya, lihat Wilayah cluster elastis dan ketersediaan versi.

Date	Catatan rilis

3 Agustus 2023	Fitur baru
	
	Amazon DocumentDB Cluster elastis
	
	• Amazon DocumentDB Elastic cluster sekarang mendukung operasi berikut:
	  • `top`
	  • `collStats`
	  • `hint`
	  • `dataSize`
	
	  Lihat API MongoDB yang didukung, operasi, dan tipe data di Amazon DocumentDB daftar lengkap perintah dan operasi yang didukung.
	• Indeks Time to Live (TTL) sekarang didukung.
	• Indeks `hints` sekarang didukung dengan ekspresi indeks.

22 Maret 2023	Fitur baru Cluster elastis Amazon DocumentDB sekarang tersedia di wilayah Asia Pasifik (Singapura), Asia Pasifik (Sydney), dan Asia Pasifik (Tokyo). Untuk informasi selengkapnya, lihat Wilayah cluster elastis dan ketersediaan versi.
30 November 2022	Fitur baru Amazon DocumentDB Cluster elastis Amazon DocumentDB elastis cluster adalah jenis baru dari Amazon DocumentDB cluster yang memungkinkan pengguna untuk memanfaatkan API sharding MongoDB untuk skala cluster mereka. Cluster elastis menangani hampir semua jumlah pembacaan dan penulisan dengan kapasitas penyimpanan petabyte dengan mendistribusikan data dan menghitung di beberapa instance dan volume komputasi yang mendasarinya. Untuk mempelajari selengkapnya, lihat Menggunakan cluster elastis Amazon DocumentDB.

Riwayat dokumen untuk Panduan Pengembang Amazon DocumentDB

- Versi API: 2014-10-31
- Pembaruan dokumentasi terbaru: 27 Mei 2026

Tabel berikut menguraikan dokumentasi untuk rilis Panduan Developer Amazon DocumentDB ini.

Perubahan	Deskripsi	Tanggal
Panduan Pengembang Amazon DocumentDB Baru	Panduan Pengembang Amazon DocumentDB sekarang diterbitkan di URL baru. Referensi API tersedia secara terpisah di Referensi API Amazon DocumentDB.	27 Mei 2026
AWS pembaruan kebijakan terkelola - perubahan kebijakan	Amazon DocumentDB memperbarui kebijakan akses penuh untuk cluster elastis.	Februari 11, 2025
AWS pembaruan kebijakan terkelola - perubahan kebijakan	Amazon DocumentDB memperbarui kebijakan akses penuh untuk cluster elastis.	Februari 21, 2024
AWS pembaruan kebijakan terkelola - perubahan kebijakan	Amazon DocumentDB memperbarui kebijakan read-only dan akses penuh untuk cluster elastis.	Juni 21, 2023
AWS pembaruan kebijakan terkelola - kebijakan baru	Amazon DocumentDB memperkenalkan kebijakan hanya-baca baru untuk cluster elastis.	8 Juni 2023

AWS pembaruan kebijakan terkelola - kebijakan baru	Amazon DocumentDB memperkenalkan kebijakan akses penuh baru untuk cluster elastis.	Juni 5, 2023
MongoDB 5.0 Kompatibilitas	Amazon DocumentDB sekarang kompatibel dengan versi 5.0 dari MongoDB.	1 Maret 2023
Pembaruan Kebijakan	Untuk mendukung fitur cluster elastis Amazon DocumentDB AmazonDocDBConsoleFullAccess , kebijakan diperbarui, dan AmazonDocDB-ElasticServiceRolePolicy diperkenalkan.	30 November 2022
Cluster Elastis	Menambahkan fitur Elastic Cluster baru yang mendukung partisi berbasis hash (sharding) data di seluruh sistem penyimpanan terdistribusi Amazon DocumentDB.	30 November 2022
Cluster Global	Menambahkan dokumentasi tentang cara menggunakan Klaster Global.	2 Juni 2021
Langganan Acara	Menambahkan dokumentasi langganan peristiwa.	26 Maret 2021
Versi 3.6 Upgrade	Perbaikan terdokumentasi ke versi 3.6 dalam kontrol akses berbasis peran, operator agregasi, dan performa.	15 Januari 2021

MongoDB 4.0 Kompatibilitas	Amazon DocumentDB kini kompatibel dengan MongoDB versi 4.0.	9 November 2020
Panduan Memulai	Panduan Memulai Baru untuk memulai menggunakan Amazon DocumentDB, AWS Cloud9 Amazon EC2, Robo3T atau Studio3T.	15 Agustus 2020
Zona Ketersediaan Tambahan didukung	Amazon DocumentDB menambahkan dukungan untuk Availability Zone tambahan di Asia Pacific (Seoul) (ap-northeast-2).	14 Juli 2020
Menambahkan dukungan untuk menyalin snapshot di seluruh Wilayah.	Amazon DocumentDB menambahkan dukungan untuk menyalin snapshot klaster di seluruh Wilayah AWS. Untuk informasi lebih lanjut, lihat Menyalin Snapshot di Seluruh Wilayah.	10 Juli 2020
Ditambahkan dukungan untuk kelas contoh T3.	Menambahkan dukungan untuk tipe instans T3 di semua Wilayah yang mendukung Amazon DocumentDB. Untuk informasi selengkapnya, lihat Kelas Instans yang Didukung berdasarkan Wilayah dan Spesifikasi Kelas Instans.	30 Juni 2020
Menambahkan dukungan untuk AWS GovCloud (US).	Amazon DocumentDB sekarang tersedia di Wilayah (AWS GovCloud (US) us-gov-west-1).	29 Juni 2020

Menambahkan 16 CloudWatch metrik baru.	Amazon DocumentDB menambahkan dukungan untuk 16 metrik Amazon baru. CloudWatch Untuk informasi selengkapnya, lihat Memantau Amazon CloudWatch DocumentDB dengan.	23 Juni 2020
Ditambahkan dukungan untuk karakter null dan $ regex operator.	Amazon DocumentDB menambahkan dukungan untuk karakter null dalam string dan kemampuan untuk menggunakan sebuah indeks untuk $regex. Untuk melihat API MongoDB dan kemampuan alur agregasi yang didukung untuk Amazon DocumentDB, lihat Perbedaan Fungsional dengan MongoDB.	22 Juni 2020
Menambahkan dukungan untuk meningkatkan kemampuan pengindeksan multi-kunci.	Amazon DocumentDB menambahkan dukungan untuk kemampuan pengindeksan multi-kunci yang ditingkatkan yang mencakup pengindeksan array yang lebih besar dari 2.048 byte dan kemampuan untuk membuat indeks multi-kunci campuran dengan beberapa kunci dalam array yang sama. Untuk informasi selengkapnya, lihat Perbedaan Fungsional dengan MongoDB.	23 April 2020

Menambahkan dukungan untuk perlindungan penghapusan untuk tumpukan Amazon DocumentDB. CloudFormation	Amazon DocumentDB menambahkan dukungan untuk mengaktifkan perlindungan penghapusan saat membuat tumpukan Amazon DocumentDB. CloudFormation	20 April 2020
Menambahkan dukungan untuk kontrol akses berbasis peran.	Amazon DocumentDB menambahkan dukungan untuk kontrol akses berbasis peran menggunakan peran bawaan.	26 Maret 2020
Menambahkan dukungan untuk Availability Zone tambahan di Kanada (Tengah) (ca-central-1).	Amazon DocumentDB kini tersedia di Wilayah Canada (Central) (ca-central-1) dengan instans kelas R5 dan 3 Availability Zone.	26 Maret 2020
Menambahkan dukungan untuk dua API MongoDB tambahan.	Amazon DocumentDB menambahkan dukungan untuk API MongoDB `$dateFromString` dan `executionStats` .	23 Maret 2020
Menambahkan dukungan untuk lima API MongoDB tambahan.	Amazon DocumentDB menambahkan dukungan untuk API MongoDB `$objectToArray` , `$arrayToObject` , `$slice`, `$mod`, dan `$range`.	6 Februari 2020
Menambahkan dukungan untuk Kanada (Tengah).	Amazon DocumentDB kini tersedia di Wilayah Canada (Central) (ca-central-1) dengan instans kelas R5.	11 Desember 2019

Menambahkan dukungan untuk ChangeStreamLogSize.	Amazon DocumentDB menambahkan dukungan untuk `ChangeStreamLogSize` untuk metrik Cloudwatch.	22 November 2019
Menambahkan dukungan untuk wilayah Eropa (Paris)	Amazon DocumentDB kini tersedia di wilayah Europe (Paris) (eu-west-3) dengan instans kelas R5.	30 Oktober 2019
Menambahkan dukungan untuk wilayah Asia Pasifik (Mumbai)	Amazon DocumentDB kini tersedia di wilayah Asia Pacific (Mumbai) (ap-south-1) dengan instans kelas R5.	17 Oktober 2019
Menambahkan dukungan untuk tiga API MongoDB tambahan	Amazon DocumentDB menambahkan dukungan untuk API MongoDB `$addFields` , `$concatArrays` , dan `$lookup`.	16 Oktober 2019
Menambahkan dukungan untuk wilayah Asia Pasifik (Singapura)	Amazon DocumentDB kini tersedia di wilayah Asia Pacific (Singapore) (ap-southeast-1) dengan instans kelas R5.	14 Oktober 2019
Menambahkan dokumen baru untuk memperbarui sertifikat TLS	Menambahkan instruksi untuk memperbarui sertifikat CA guna menggunakan sertifikat CA baru untuk membuat koneksi TLS.	2 Oktober 2019
Menambahkan dukungan API untuk sertifikat	Amazon DocumentDB tipe data Sertifikat baru untuk instans. Untuk informasi selengkapnya, lihat DBInstance.	1 Oktober 2019

Support untuk pembuatan profil kueri	Amazon DocumentDB menambahkan kemampuan untuk membuat profil operasi yang didukung pada instans dan basis data klaster Anda.	19 Agustus 2019
Menambahkan AZ ketiga di Asia Pasifik (Tokyo)	Amazon DocumentDB menambahkan Availability Zone (AZ) ketiga untuk instans komputasi Anda di Asia Pacific (Tokyo).	9 Agustus 2019
Support untuk API Mongo tambahan	Menambahkan dukungan untuk kemampuan alur agregasi tambahan yang mencakup operator agregasi `$in`, `$isoWeek`, `$isoWeekYear` , `$isoDayOfWeek` , dan `$dateToString` serta tahap agregasi `$addToSet` . Amazon DocumentDB juga menambahkan dukungan untuk perintah `top()` guna diagnostik tingkat koleksi dan kemampuan untuk memodifikasi parameter `expireAfterSeconds` untuk pengindeksan TTL menggunakan perintah `collMod()` .	31 Juli 2019
Menambahkan dukungan untuk Eropa (London)	Amazon DocumentDB kini tersedia di Europe (London) (eu-west-2) dengan instans kelas R5.	18 Juli 2019

Contoh kode yang ditambahkan	Menambahkan contoh kode di R dan Ruby untuk menghubungkan secara terprogram ke Amazon DocumentDB.	17 Juli 2019
Menambahkan praktik terbaik	Menambahkan Praktik Terbaik untuk membantu Anda mengelola biaya Amazon DocumentDB Anda.	17 Juli 2019
Support untuk menghentikan dan memulai cluster	Amazon DocumentDB menambahkan dukungan untuk menghentikan dan memulai klaster guna mengelola biaya pengembangan dan lingkungan pengujian.	1 Juli 2019
Support untuk perlindungan penghapusan klaster	Untuk melindungi klaster Anda dari penghapusan yang tidak disengaja, Amazon DocumentDB menambahkan perlindungan penghapusan. Untuk informasi selengkapnya, lihat topik berikut: Membuat Klaster Amazon DocumentDB, Memodifikasi Klaster Amazon DocumentDB, Menghapus Klaster Amazon DocumentDB, dan `DeletionProtection` dalam topik API DBCluster.	1 Juli 2019
Pembaruan perbedaan fungsional	Menambahkan Transaksi Implisit ke Perbedaan Fungsional.	26 Juni 2019

Penambahan perbedaan fungsional	Menambahkan catatan mengenai penyimpanan dan kompresi indeks di Amazon DocumentDB.	13 Juni 2019
Wilayah tambahan didukung	Amazon DocumentDB kini tersedia di wilayah Asia Pacific (Sydney) (ap-southeast-2) dengan instans kelas R5.	5 Juni 2019
Kelas instans R5 didukung di wilayah tambahan	Menambahkan dukungan kelas instans R5 untuk 4 wilayah tambahan: US East (Ohio), US East (N. Virginia), US West (Oregon), dan EU (Ireland). Dengan perubahan ini, instans R5 didukung di semua wilayah yang mendukung Amazon DocumentDB.	17 Mei 2019
Wilayah tambahan yang didukung	Menambahkan dukungan untuk 2 wilayah tambahan, Asia Pacific (Tokyo) (ap-northeast-1) dan Asia Pacific (Seoul) (ap-northeast-2) dengan kelas instans R5. Untuk informasi selengkap nya, lihat Kelas Instans yang Didukung berdasarkan Wilayah dan Spesifikasi Kelas Instans.	8 Mei 2019
Menambahkan lebih banyak contoh kode koneksi	Menambahkan contoh kode di Java dan C# untuk menghubungkan ke Amazon DocumentDB.	24 April 2019

Dukungan Mongo API tambahan	Menambahkan dukungan untuk tujuh operator string agregasi (`$indexOfBytes`, `$indexOfCP`, `$strLenBytes`, `$strLenCP`, `$toLower`, `$toUpper`, dan `$split`), sembilan operator tanggal-waktu (`$dayOfYear`, `$dayOfMonth`, `$dayOfWeek`, `$year`, `$month`, `$hour`, `$minute`, `$second`, dan `$millisecond`), dan tahap alur agregasi `$sample`.	4 April 2019
Ditambahkan contoh kode koneksi	Menambahkan contoh kode di Python, PHP Node.js, dan Go untuk menghubungkan ke Amazon DocumentDB.	21 Maret 2019
Support untuk Wilayah Frankfurt dan instans R5	Dukungan tambahan untuk Wilayah Europe (Frankfurt) (eu-central-1) dengan kelas instans R5. Untuk informasi selengkapnya, lihat Kelas Instans yang Didukung berdasarkan Wilayah dan Spesifikasi Kelas Instans.	13 Maret 2019

Dukungan operator pipa agregasi	Menambahkan dukungan untuk operator string agregasi baru (`$concat`, `$substr`, `$substrBytes` , `$substrCP` , `$strcasecmp`), sebuah operator agregasi array (`$size`), sebuah operator akumulator grup agregasi (`$push`), dan tahap agregasi (`$redact` dan `$indexStats`). Kami juga menambahkan dukungan untuk operator array posisional (`$[]` dan `$[<identifier>]`) dan `hint()`.	28 Februari 2019
Upgrade mesin	Menambahkan dokumentasi untuk menentukan modifikasi klaster yang tertunda dan meningkatkan versi mesin klaster Anda.	15 Februari 2019
Acara audit	Menambahkan dukungan untuk mengaudit peristiwa database dengan Amazon CloudWatch Logs.	12 Februari 2019
Mulai Cepat	Menambahkan topik Mulai Cepat untuk membantu Anda memulai dengan mudah menggunakan Amazon CloudFormation DocumentDB.	11 Januari 2019

Rilis Publik	Ini adalah rilis publik awal Amazon DocumentDB (dengan kompatibilitas MongoDB). Rilis ini mencakup Panduan Developer dan Referensi API Manajemen Sumber Daya terintegrasi.	9 Januari 2019

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.