
Amazon DocumentDB

Developer Guide



Amazon DocumentDB: Developer Guide

Copyright © 2022 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What Is Amazon DocumentDB	1
Overview	1
Clusters	2
Instances	2
Regions and AZs	4
Regions	4
Availability Zones	4
Pricing	5
Monitoring	6
Interfaces	6
AWS Management Console	6
AWS CLI	6
The mongo Shell	6
MongoDB Drivers	6
What's Next?	6
How It Works	7
Amazon DocumentDB Endpoints	8
TLS Support	10
Amazon DocumentDB Storage	10
Amazon DocumentDB Replication	11
Amazon DocumentDB Reliability	11
Read Preference Options	12
TTL Deletes	15
Billable Resources	15
What is a Document Database?	17
Use Cases	17
Understanding Documents	18
Working with Documents	22
Get Started Guide	31
Prerequisites	31
Step 1: Create an AWS Cloud9 environment	32
Step 2: Create a security group	35
Step 3: Create an Amazon DocumentDB cluster	37
Step 4: Install the mongo shell	40
Step 5: Connect to your Amazon DocumentDB cluster	41
Step 6: Insert and query data	41
Step 7: Explore	43
Quick Start using AWS CloudFormation	44
Prerequisites	44
Required IAM Permissions	44
Amazon EC2 Key Pair	46
Launching an Amazon DocumentDB AWS CloudFormation Stack	46
Accessing the Amazon DocumentDB Cluster	49
Termination Protection and Deletion Protection	49
MongoDB 4.0 Compatibility	51
What's new in Amazon DocumentDB 4.0	51
Get Started with Amazon DocumentDB 4.0	52
Upgrade or Migrate to Amazon DocumentDB 4.0	52
Functional Differences	52
Functional Differences Between Amazon DocumentDB 3.6 and 4.0	52
Functional Differences Between Amazon DocumentDB 4.0 and MongoDB 4.0	53
Transactions	54
Requirements	54
Best Practices	54

Limitations	54
Monitoring and Diagnostics	55
Transaction Isolation Level	56
Use Cases	56
Multi-Statement Transactions	56
Multi-Collection Transactions	57
Transaction API Examples for Callback API	58
Transaction API Examples for Core API	58
Supported Commands	81
Unsupported Capabilities	81
Sessions	82
Causal consistency	82
Retryable writes	83
Transaction Errors	83
Best Practices	84
Basic Operational Guidelines	84
Instance Sizing	85
Working with Indexes	86
Building Indexes	86
Index Selectivity	86
Impact of Indexes on Writing Data	86
Identifying Missing Indexes	87
Identifying Unused Indexes	87
Security Best Practices	87
Cost Optimization	87
Using Metrics to Identify Performance Issues	88
Viewing Performance Metrics	88
Setting a CloudWatch Alarm	88
Evaluating Performance Metrics	88
Tuning Queries	89
TTL and Timeseries Workloads	90
Migrations	90
Working with Cluster Parameter Groups	90
Aggregation Pipeline Queries	90
batchInsert and batchUpdate	91
Functional Differences with MongoDB	92
Functional Benefits of Amazon DocumentDB	92
Implicit Transactions	92
Updated Functional Differences	93
Array Indexing	93
Multi-key Indexes	94
Null Characters in Strings	94
Role-Based Access Control	94
\$regex Indexing	95
Projection for Nested Documents	95
Functional Differences with MongoDB	95
Admin Databases and Collections	96
cursormaxTimeMS	96
explain()	96
Field Name Restrictions	96
Index Builds	97
Lookup with empty key in path	97
MongoDB APIs, Operations, and Data Types	97
mongodump and mongorestore Utilities	97
Result Ordering	97
Retryable Writes	98
Sparse Index	98

Storage Compression	98
Using \$elemMatch Within an \$all Expression	98
\$ne, \$nin, \$nor, \$not, \$exists, and \$elemMatch Indexing	99
\$lookup	99
Supported MongoDB APIs, Operations, and Data Types	102
Database Commands	102
Administrative Commands	103
Aggregation	103
Authentication	104
Diagnostic Commands	104
Query and Write Operations	104
Role Management Commands	105
Sessions Commands	105
User Management	106
Query and Projection Operators	106
Array Operators	106
Bitwise Operators	107
Comment Operator	107
Comparison Operators	107
Element Operators	107
Evaluation Query Operators	108
Logical Operators	108
Projection Operators	108
Update Operators	108
Array Operators	109
Bitwise Operators	109
Field Operators	109
Update Modifiers	110
Geospatial	110
Geometry Specifiers	110
Query Selectors	110
Cursor Methods	111
Aggregation Pipeline Operators	112
Accumulator Expressions	112
Arithmetic Operators	113
Array Operators	113
Boolean Operators	114
Comparison Operators	114
Conditional Expression Operators	114
Data Type Operator	115
Date Operators	115
Literal Operator	115
Merge Operator	116
Natural Operator	116
Set Operators	116
Stage Operators	116
String Operators	117
System Variables	118
Text Search Operator	118
Type Conversion Operators	118
Variable Operators	119
Data Types	119
Indexes and Index Properties	120
Indexes	120
Index Properties	120
Migrating to Amazon DocumentDB	121
Migrating Between Versions	121

Step 1: Enable Change Streams	122
Step 2: Modify the Change Streams Retention Duration	122
Step 3: Migrate Your Indexes	122
Step 4: Create a AWS DMS Replication Instance	123
Step 5: Create an AWS DMS Source Endpoint	124
Step 6: Create an AWS DMS Target Endpoint	126
Step 7: Create and run a migration task	127
Step 8: Changing the application endpoint to the Amazon DocumentDB cluster 4.0	128
Migration Tools	128
AWS Database Migration Service	129
Command Line Utilities	129
Discovery	129
Planning: Amazon DocumentDB Cluster Requirements	132
Migration Approaches	134
Offline	134
Online	135
Hybrid	136
Migration Sources	137
Migration Connectivity	138
Testing	140
Migration Plan Testing Considerations	140
Performance Testing	142
Failover Testing	142
Additional Resources	142
Security	143
Data Protection	143
Encrypting Data at Rest	144
Encrypting Data in Transit	147
Key Management	153
Identity and Access Management	153
Authentication	154
Overview of Managing Access	155
Managing Access Using Policies	158
Using Identity-Based Policies (IAM Policies)	158
Amazon DocumentDB API Permissions Reference	161
Managing Amazon DocumentDB Users	166
Master and serviceadmin User	166
Creating Additional Users	167
Automatically Rotating Passwords	168
Role-Based Access Control	169
RBAC Concepts	169
Getting Started with RBAC built-in roles	170
Getting Started with RBAC user-defined roles	173
Connecting to Amazon DocumentDB as a User	176
Common Commands	177
Functional Differences	181
Limits	181
Restricting Database Access Using Role-Based Access Control	181
Logging and Monitoring	186
Updating Certificates	187
Updating Your Application and Amazon DocumentDB Cluster	187
Troubleshooting	189
Frequently Asked Questions	190
Updating Certificates — GovCloud (US-West)	195
Updating Your Application and Amazon DocumentDB Cluster	187
Troubleshooting	189
Frequently Asked Questions	190

Compliance Validation	202
Resilience	203
Infrastructure Security	204
Security Best Practices	204
Auditing Events	205
Supported Events	205
Enabling Auditing	208
Disabling Auditing	212
Accessing Your Audit Events	214
Backing Up and Restoring	215
Back Up and Restore: Concepts	215
Understanding Backup Storage Usage	217
Dumping, Restoring, Importing, and Exporting Data	218
mongodump	218
mongorestore	218
mongoexport	219
mongoimport	219
Tutorial	219
Cluster Snapshot Considerations	221
Backup Storage	222
Backup Window	222
Backup Retention Period	222
Comparing Automatic and Manual Snapshots	223
Creating a Manual Cluster Snapshot	224
Create a Cluster Snapshot Using the AWS Management Console	224
Create a Cluster Snapshot Using the AWS CLI	225
Copying a Cluster Snapshot	226
Copying Shared Snapshots	227
Copying Snapshots Across AWS Regions	227
Limitations	227
Handling Encryption	227
Parameter Group Considerations	227
Copying a Cluster Snapshot	228
Sharing a Cluster Snapshot	232
Sharing an Encrypted Snapshot	233
Sharing a Snapshot	235
Restoring from a Cluster Snapshot	236
Restore from a Cluster Snapshot Using the AWS Management Console	237
Restore from a Cluster Snapshot Using the AWS CLI	238
Restoring to a Point in Time	241
Restore to a Point in Time Using the AWS Management Console	241
Restore To a Point in Time Using the AWS CLI	243
Deleting a Cluster Snapshot	245
Delete a Cluster Snapshot Using the AWS Management Console	245
Delete a Cluster Snapshot Using the AWS CLI	245
Managing Amazon DocumentDB	247
Operational Tasks Overview	247
Adding a Replica to an Amazon DocumentDB Cluster	247
Describing Clusters and Instances	248
Creating a Cluster Snapshot	249
Restoring from a Snapshot	250
Removing an Instance from a Cluster	251
Deleting a Cluster	251
Global Clusters	252
What is a global cluster?	252
How are global clusters useful?	252
What are the current limitations of global clusters?	252

Quick Start Guide	253
Managing Global Clusters	262
Connecting Global Clusters	267
Monitoring Global Clusters	267
Disaster Recovery	268
Managing Clusters	269
Understanding Clusters	270
Cluster Settings	271
Determining a Cluster's Status	272
Cluster Lifecycle	273
Scaling Amazon DocumentDB Clusters	300
Cloning a volume for an Amazon DocumentDB cluster	302
Understanding Fault Tolerance	312
Managing Instances	313
Managing Instance Classes	313
Determining an Instance's Status	319
Instance Lifecycle	319
Managing Subnet Groups	334
Creating a Subnet Group	335
Describing a Subnet Group	338
Modifying a Subnet Group	340
Deleting a Subnet Group	342
High Availability and Replication	343
Read Scaling	344
High Availability	344
Adding Replicas	345
Failover	345
Replication Lag	348
Managing Events	349
Viewing Event Categories	349
Viewing Amazon DocumentDB Events	351
Choosing Regions and Availability Zones	353
Region Availability	354
Managing Cluster Parameter Groups	355
Describing Cluster Parameter Groups	355
Creating Cluster Parameter Groups	360
Modifying Cluster Parameter Groups	362
Modifying Clusters to Use Customized Cluster Parameter Groups	365
Copying Cluster Parameter Groups	366
Resetting Cluster Parameter Groups	367
Deleting Cluster Parameter Groups	369
Cluster Parameters Reference	371
Understanding Endpoints	378
Finding a Cluster's Endpoints	379
Finding an Instance's Endpoint	380
Connecting to Endpoints	383
Understanding Amazon DocumentDB ARNs	384
Constructing an ARN	384
Finding an ARN	386
Tagging Resources	387
Overview of Resource Tags	388
Tag Constraints	388
Adding or Updating Tags	389
Listing Tags	390
Removing Tags	391
Maintaining Amazon DocumentDB	392
Determining Pending Maintenance Actions	392

Applying Updates	393
User-Initiated Updates	396
Managing Your Maintenance Windows	397
Understanding Service-Linked Roles	398
Service-Linked Role Permissions	398
Creating a Service-Linked Role	399
Modifying a Service-Linked Role	399
Deleting a Service-Linked Role	400
Supported Regions for Amazon DocumentDB Service-Linked Roles	400
Monitoring Amazon DocumentDB	401
Monitoring a Cluster's Status	402
Cluster Status Values	402
Monitoring a Cluster's Status Using the AWS Management Console	403
Monitoring a Cluster's Status Using the AWS CLI	404
Monitoring an Instance's Status	404
Instance Status Values	405
Monitoring an Instance's Status Using the AWS Management Console	406
Monitoring an Instance's Status Using the AWS CLI	407
Event Subscriptions	407
Subscribing to Events	408
Manage Subscriptions	409
Categories and Messages	412
Monitoring Amazon DocumentDB with CloudWatch	414
Amazon DocumentDB Metrics	414
Viewing CloudWatch Data	422
Amazon DocumentDB Dimensions	426
Monitoring Opcounters	426
Monitoring Database Connections	426
Logging Amazon DocumentDB API Calls with CloudTrail	426
Amazon DocumentDB Information in CloudTrail	427
Profiling Operations	427
Supported Operations	428
Limitations	428
Enabling the Profiler	428
Disabling the Profiler	431
Disabling Profiler Logs Export	432
Accessing Your Profiler Logs	434
Common Queries	434
Monitoring with Performance Insights	434
Performance Insights concepts	435
Enabling and disabling Performance Insights	437
Configuring access policies for Performance Insights	439
Analyzing metrics with the Performance Insights dashboard	443
Retrieving metrics with the Performance Insights API	456
Amazon CloudWatch metrics for Performance Insights	466
Performance Insights for counter metrics	467
Developing with Amazon DocumentDB	469
Connecting Programmatically	469
Determining the <code>tls</code> Value	469
Connecting with TLS Enabled	471
Connecting with TLS Disabled	480
Using Change Streams	486
Supported Operations	487
Billing	487
Limitations	487
Enabling Change Streams	487
Example	489

Full Document Lookup	491
Resuming a Change Stream	491
Resuming a Change Stream with <code>startAtOperationTime</code>	492
Transactions in change streams	494
Modifying the Change Stream Log Retention Duration	494
Connecting as a Replica Set	496
Using Cluster Connections	498
Multiple Connection Pools	499
Summary	499
Connecting from Outside an Amazon VPC	499
Connect Using Robo 3T	500
Prerequisites	500
Connect with Robo 3T	501
Connect Using Studio 3T	503
Prerequisites	500
Connect with Studio 3T	503
Connect Using Amazon EC2	509
Prerequisites	509
Step 1: Create an Amazon EC2 Instance	510
Step 2: Create a security group	514
Step 3: Create an Amazon DocumentDB Cluster	516
Step 4: Connect to your Amazon EC2 instance	518
Step 5: Install the mongo shell	519
Step 6: Manage Amazon DocumentDB TLS	520
Step 7: Connect to your Amazon DocumentDB cluster	520
Step 8: Insert and query data	41
Step 9: Explore	523
Connect Using JDBC Driver	523
Getting Started	523
Connect from Tableau Desktop	524
Connect from DbVisualizer	527
Automatic schema generation	528
SQL Support and Limitations	534
Troubleshooting	534
Quotas and Limits	535
Supported Instance Types	535
Supported Regions	536
Regional Quotas	536
Aggregation Limits	538
Cluster Limits	538
Instance Limits	539
Naming Constraints	541
TTL Constraints	542
Querying	543
Querying Documents	543
Retrieving All Documents	543
Matching Field Values	544
Embedded Documents	544
Field Values in Embedded Documents	544
Matching an Array	544
Matching Values in an Array	545
Using Operators	545
Geospatial Data	545
Overview	1
Indexing and Storing Geospatial Data	545
Querying Geospatial Data	547
Limitations	548

Query Plan	548
Query Plan	548
Query Plan Cache	550
Explain Results	550
Scan and Filter Stage	551
Index Intersection	551
Index Union	552
Multiple Index Intersection/Union	552
Compound Index	553
Sort Stage	553
Group Stage	553
Troubleshooting	554
Connection Issues	554
Cannot Connect to an Amazon DocumentDB Endpoint	554
Testing a Connection to an Amazon DocumentDB Instance	556
Connecting to an Invalid Endpoint	557
Index Creation	557
Index Build Fails	557
Background Index Build Latency Issues and Fails	558
Performance and Resource Utilization	558
Find and Terminate Long Running or Blocked Queries	558
See a Query Plan and Optimize a Query	560
List All Running Operations on an Instance	561
Know When a Query Is Making Progress	562
Determine Why a System Suddenly Runs Slowly	564
Determine the Cause of High CPU Utilization	565
How Do I Determine the Open Cursors on an Instance?	566
How do I Determine the Current Amazon DocumentDB Engine Version?	566
How Do I Identify Unused Indexes?	567
How Do I Identify Missing Indexes?	567
Summary of Useful Queries	568
Resource Management API Reference	570
Actions	570
AddSourceIdentifierToSubscription	572
AddTagsToResource	574
ApplyPendingMaintenanceAction	576
CopyDBClusterParameterGroup	578
CopyDBClusterSnapshot	580
CreateDBCluster	584
CreateDBClusterParameterGroup	590
CreateDBClusterSnapshot	592
CreateDBInstance	594
CreateDBSubnetGroup	599
CreateEventSubscription	601
CreateGlobalCluster	604
DeleteDBCluster	607
DeleteDBClusterParameterGroup	609
DeleteDBClusterSnapshot	611
DeleteDBInstance	613
DeleteDBSubnetGroup	615
DeleteEventSubscription	617
DeleteGlobalCluster	619
DescribeCertificates	621
DescribeDBClusterParameterGroups	623
DescribeDBClusterParameters	625
DescribeDBClusters	627
DescribeDBClusterSnapshotAttributes	629

DescribeDBClusterSnapshots	631
DescribeDBEngineVersions	634
DescribeDBInstances	637
DescribeDBSubnetGroups	639
DescribeEngineDefaultClusterParameters	641
DescribeEventCategories	643
DescribeEvents	645
DescribeEventSubscriptions	648
DescribeGlobalClusters	650
DescribeOrderableDBInstanceOptions	652
DescribePendingMaintenanceActions	654
FailoverDBCluster	656
ListTagsForResource	658
ModifyDBCluster	660
ModifyDBClusterParameterGroup	665
ModifyDBClusterSnapshotAttribute	667
ModifyDBInstance	669
ModifyDBSubnetGroup	674
ModifyEventSubscription	676
ModifyGlobalCluster	678
RebootDBInstance	680
RemoveFromGlobalCluster	682
RemoveSourceIdentifierFromSubscription	684
RemoveTagsFromResource	686
ResetDBClusterParameterGroup	688
RestoreDBClusterFromSnapshot	690
RestoreDBClusterToPointInTime	695
StartDBCluster	700
StopDBCluster	702
Data Types	703
AvailabilityZone	704
Certificate	705
CloudwatchLogsExportConfiguration	707
DBCluster	708
DBClusterMember	713
DBClusterParameterGroup	714
DBClusterRole	715
DBClusterSnapshot	716
DBClusterSnapshotAttribute	719
DBClusterSnapshotAttributesResult	720
DBEngineVersion	721
DBInstance	723
DBInstanceStateInfo	727
DBSubnetGroup	728
Endpoint	730
EngineDefaults	731
Event	732
EventCategoriesMap	734
EventSubscription	735
Filter	737
GlobalCluster	738
GlobalClusterMember	740
OrderableDBInstanceOption	741
Parameter	743
PendingCloudwatchLogsExports	745
PendingMaintenanceAction	746
PendingModifiedValues	748

ResourcePendingMaintenanceActions	751
Subnet	752
Tag	753
UpgradeTarget	754
VpcSecurityGroupMembership	755
Common Errors	755
Common Parameters	757
Release Notes	759
January 21, 2022	759
New Features	759
October 25, 2021	760
New Features	760
Bug fixes and other changes	760
June 24, 2021	760
New Features	760
May 4, 2021	760
New Features	760
Bug fixes and other changes	761
January 15, 2021	761
New Features	761
November 9, 2020	761
New Features	761
Bug Fixes and Other Changes	762
October 30, 2020	763
New Features	763
Bug Fixes and Other Changes	763
September 22, 2020	763
New Features	763
Bug Fixes and Other Changes	764
July 10, 2020	764
New Features	764
Bug Fixes and Other Changes	764
June 30, 2020	764
New Features	764
Bug Fixes and Other Changes	764
Document History	766

What Is Amazon DocumentDB (with MongoDB Compatibility)

Amazon DocumentDB (with MongoDB compatibility) is a fast, reliable, and fully managed database service. Amazon DocumentDB makes it easy to set up, operate, and scale MongoDB-compatible databases in the cloud. With Amazon DocumentDB, you can run the same application code and use the same drivers and tools that you use with MongoDB.

Before using Amazon DocumentDB, you should review the concepts and features described in [How It Works \(p. 7\)](#). After that, complete the steps in [Get Started Guide \(p. 31\)](#).

Topics

- [Overview of Amazon DocumentDB \(p. 1\)](#)
- [Clusters \(p. 2\)](#)
- [Instances \(p. 2\)](#)
- [Regions and Availability Zones \(p. 4\)](#)
- [Amazon DocumentDB Pricing \(p. 5\)](#)
- [Monitoring \(p. 6\)](#)
- [Interfaces \(p. 6\)](#)
- [What's Next? \(p. 6\)](#)
- [Amazon DocumentDB: How It Works \(p. 7\)](#)
- [What is a Document Database? \(p. 17\)](#)

Overview of Amazon DocumentDB

The following are some high-level features of Amazon DocumentDB:

- Amazon DocumentDB automatically grows the size of your storage volume as your database storage needs grow. Your storage volume grows in increments of 10 GB, up to a maximum of 64 TiB. You don't need to provision any excess storage for your cluster to handle future growth.
- With Amazon DocumentDB, you can increase read throughput to support high-volume application requests by creating up to 15 replica instances. Amazon DocumentDB replicas share the same underlying storage, lowering costs and avoiding the need to perform writes at the replica nodes. This capability frees up more processing power to serve read requests and reduces the replica lag time—often down to single digit milliseconds. You can add replicas in minutes regardless of the storage volume size. Amazon DocumentDB also provides a reader endpoint, so the application can connect without having to track replicas as they are added and removed.
- Amazon DocumentDB lets you scale the compute and memory resources for each of your instances up or down. Compute scaling operations typically complete in a few minutes.
- Amazon DocumentDB runs in Amazon Virtual Private Cloud (Amazon VPC), so you can isolate your database in your own virtual network. You can also configure firewall settings to control network access to your cluster.
- Amazon DocumentDB continuously monitors the health of your cluster. On an instance failure, Amazon DocumentDB automatically restarts the instance and associated processes. Amazon DocumentDB doesn't require a crash recovery replay of database redo logs, which greatly reduces restart times. Amazon DocumentDB also isolates the database cache from the database process, enabling the cache to survive an instance restart.

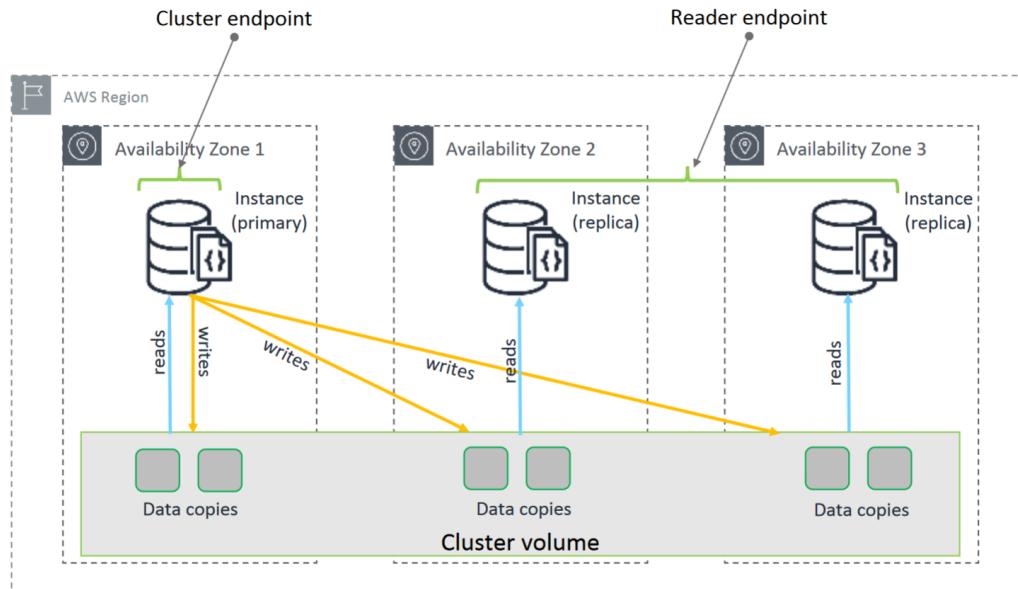
- On instance failure, Amazon DocumentDB automates failover to one of up to 15 Amazon DocumentDB replicas that you create in other Availability Zones. If no replicas have been provisioned and a failure occurs, Amazon DocumentDB tries to create a new Amazon DocumentDB instance automatically.
- The backup capability in Amazon DocumentDB enables point-in-time recovery for your cluster. This feature allows you to restore your cluster to any second during your retention period, up to the last 5 minutes. You can configure your automatic backup retention period up to 35 days. Automated backups are stored in Amazon Simple Storage Service (Amazon S3), which is designed for 99.999999999% durability. Amazon DocumentDB backups are automatic, incremental, and continuous, and they have no impact on your cluster performance.
- With Amazon DocumentDB, you can encrypt your databases using keys that you create and control through AWS Key Management Service (AWS KMS). On a database cluster running with Amazon DocumentDB encryption, data stored at rest in the underlying storage is encrypted. The automated backups, snapshots, and replicas in the same cluster are also encrypted.

If you are new to AWS services, use the following resources to learn more:

- AWS offers services for computing, databases, storage, analytics, and other functionality. For an overview of all AWS services, see [Cloud Computing with Amazon Web Services](#).
- AWS provides a number of database services. For guidance on which service is best for your environment, see [Databases on AWS](#).

Clusters

A *cluster* consists of 0 to 16 instances and a cluster storage volume that manages the data for those instances. All writes are done through the primary instance. All instances (primary and replicas) support reads. The cluster's data is stored in the cluster volume with copies in three different Availability Zones.



Instances

An Amazon DocumentDB instance is an isolated database environment in the cloud. An instance can contain multiple user-created databases. You can create and modify an instance using the AWS Management Console or the AWS CLI.

The computation and memory capacity of an instance are determined by its *instance class*. You can select the instance that best meets your needs. If your needs change over time, you can choose a different instance class. For instance class specifications, see [Instance Class Specifications \(p. 317\)](#).

Amazon DocumentDB instances run only in the Amazon VPC environment. Amazon VPC gives you control of your virtual networking environment: You can choose your own IP address range, create subnets, and configure routing and access control lists (ACLs).

Before you can create Amazon DocumentDB instances, you must create a cluster to contain the instances.

Not all instance classes are supported in every region. The following table shows which instance classes are supported in each region.

Supported instance classes by Region

Region	R6G	R5	R4	T4G	T3
US East (Ohio)	Supported	Supported	Supported	Supported	Supported
US East (N. Virginia)	Supported	Supported	Supported	Supported	Supported
US West (Oregon)	Supported	Supported	Supported	Supported	Supported
South America (São Paulo)	Supported	Supported		Supported	Supported
Asia Pacific (Mumbai)	Supported	Supported		Supported	Supported
Asia Pacific (Seoul)	Supported	Supported		Supported	Supported
Asia Pacific (Sydney)	Supported	Supported		Supported	Supported
Asia Pacific (Singapore)	Supported	Supported		Supported	Supported
Asia Pacific (Tokyo)	Supported	Supported		Supported	Supported
Canada (Central)	Supported	Supported		Supported	Supported
Europe (Frankfurt)	Supported	Supported		Supported	Supported
Europe (Ireland)	Supported	Supported	Supported	Supported	Supported
Europe (London)	Supported	Supported		Supported	Supported
Europe (Milan)	Supported	Supported		Supported	Supported
Europe (Paris)	Supported	Supported		Supported	Supported
China (Beijing) Region	Supported	Supported		Supported	Supported
China (Ningxia)	Supported	Supported		Supported	Supported

Region	R6G	R5	R4	T4G	T3
AWS GovCloud (US)	Supported	Supported			Supported

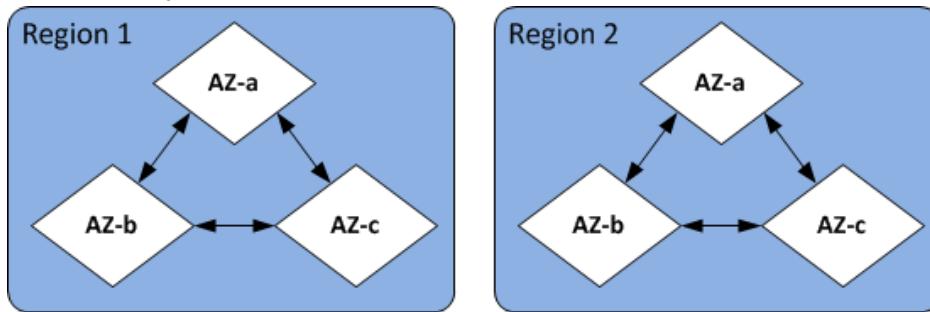
Regions and Availability Zones

Regions and Availability Zones define the physical locations of your cluster and instances.

Regions

AWS Cloud computing resources are housed in highly available data center facilities in different areas of the world (for example, North America, Europe, or Asia). Each data center location is called a *Region*.

Each AWS Region is designed to be completely isolated from the other AWS Regions. Within each are multiple Availability Zones. By launching your nodes in different Availability Zones, you can achieve the greatest possible fault tolerance. The following diagram shows a high-level view of how AWS Regions and Availability Zones work.



Availability Zones

Each AWS Region contains multiple distinct locations called *Availability Zones*. Each Availability Zone is engineered to be isolated from failures in other Availability Zones, and to provide inexpensive, low-latency network connectivity to other Availability Zones in the same Region. By launching instances for a given cluster in multiple Availability Zones, you can protect your applications from the unlikely event of an Availability Zone failing.

The Amazon DocumentDB architecture separates storage and compute. For the storage layer, Amazon DocumentDB replicates six copies of your data across three AWS Availability Zones. As an example, if you are launching an Amazon DocumentDB cluster in a Region that only supports two Availability Zones, your data storage will be replicated six ways across three Availability Zones but your compute instances will only be available in two Availability Zones.

The following table lists the number of Availability Zones that you can use in a given AWS Region to provision compute instances for your cluster.

Region Name	Region	Availability Zones (compute)
US East (Ohio)	us-east-2	3
US East (N. Virginia)	us-east-1	6

Region Name	Region	Availability Zones (compute)
US West (Oregon)	us-west-2	4
South America (São Paulo)	sa-east-1	3
Asia Pacific (Mumbai)	ap-south-1	3
Asia Pacific (Seoul)	ap-northeast-2	4
Asia Pacific (Singapore)	ap-southeast-1	3
Asia Pacific (Sydney)	ap-southeast-2	3
Asia Pacific (Tokyo)	ap-northeast-1	3
Canada (Central)	ca-central-1	3
China (Beijing) Region	cn-north-1	3
China (Ningxia)	cn-northwest-1	3
Europe (Frankfurt)	eu-central-1	3
Europe (Ireland)	eu-west-1	3
Europe (London)	eu-west-2	3
Europe (Milan)	eu-south-1	3
Europe (Paris)	eu-west-3	3
AWS GovCloud (US)	us-gov-west-1	3

Amazon DocumentDB Pricing

Amazon DocumentDB clusters are billed based on the following components. Amazon DocumentDB does not currently have a free tier so creating a cluster will incur costs.

- **Instance hours (per hour)**—Based on the instance class of the instance (for example, db.r5.xlarge). Pricing is listed on a per-hour basis, but bills are calculated down to the second and show times in decimal form. Amazon DocumentDB usage is billed in one second increments, with a minimum of 10 minutes. For more information, see [Managing Instance Classes \(p. 313\)](#).
- **I/O requests (per 1 million requests per month)** — Total number of storage I/O requests that you make in a billing cycle.
- **Backup storage (per GiB per month)** — Backup storage is the storage that is associated with automated database backups and any active database snapshots that you have taken. Increasing your backup retention period or taking additional database snapshots increases the backup storage consumed by your database. Backup storage is metered in GB-months and per second does not apply. For more information, see [Backing Up and Restoring in Amazon DocumentDB \(p. 215\)](#).
- **Data transfer (per GB)** — Data transfer in and out of your instance from or to the internet or other AWS Regions.

For detailed information, see [Amazon DocumentDB \(with MongoDB compatibility\) pricing](#).

Monitoring

There are several ways that you can track the performance and health of an instance. You can use the free Amazon CloudWatch service to monitor the performance and health of an instance. You can find performance charts on the Amazon DocumentDB console. You can subscribe to Amazon DocumentDB events to be notified when changes occur with an instance, snapshot, parameter group, or security group.

For more information, see the following:

- [Monitoring Amazon DocumentDB with CloudWatch \(p. 414\)](#)
- [Logging Amazon DocumentDB API Calls with AWS CloudTrail \(p. 426\)](#)

Interfaces

There are multiple ways for you to interact with Amazon DocumentDB, including the AWS Management Console and the AWS CLI.

AWS Management Console

The AWS Management Console is a simple web-based user interface. You can manage your clusters and instances from the console with no programming required. To access the Amazon DocumentDB console, sign in to the AWS Management Console and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.

AWS CLI

You can use the AWS Command Line Interface (AWS CLI) to manage your Amazon DocumentDB clusters and instances. With minimal configuration, you can start using all of the functionality provided by the Amazon DocumentDB console from your favorite terminal program.

- To install the AWS CLI, see [Installing the AWS Command Line Interface](#).
- To begin using the AWS CLI for Amazon DocumentDB, see [AWS Command Line Interface Reference for Amazon DocumentDB](#).

The mongo Shell

To connect to your cluster to create, read, update, delete documents in your databases, you can use the mongo shell with Amazon DocumentDB. To download and install the mongo 4.0 shell, see [Step 4: Install the mongo shell \(p. 40\)](#).

MongoDB Drivers

For developing and writing applications against an Amazon DocumentDB cluster, you can also use the MongoDB drivers with Amazon DocumentDB.

What's Next?

The preceding section introduced you to the basic infrastructure components that Amazon DocumentDB offers. What should you do next? Depending upon your circumstances, see one of the following topics to get started.

- Get started with Amazon DocumentDB by creating a cluster and instance using AWS CloudFormation [Amazon DocumentDB Quick Start Using AWS CloudFormation \(p. 44\)](#).
- Get started with Amazon DocumentDB by creating a cluster and instance using the instructions in our [Get Started Guide \(p. 31\)](#).
- Migrate your MongoDB implementation to Amazon DocumentDB using the guidance at [Migrating to Amazon DocumentDB \(p. 121\)](#)

Amazon DocumentDB: How It Works

Amazon DocumentDB (with MongoDB compatibility) is a fully managed, MongoDB-compatible database service. With Amazon DocumentDB, you can run the same application code and use the same drivers and tools that you use with MongoDB. Amazon DocumentDB is compatible with MongoDB 3.6 and 4.0.

Topics

- [Amazon DocumentDB Endpoints \(p. 8\)](#)
- [TLS Support \(p. 10\)](#)
- [Amazon DocumentDB Storage \(p. 10\)](#)
- [Amazon DocumentDB Replication \(p. 11\)](#)
- [Amazon DocumentDB Reliability \(p. 11\)](#)
- [Read Preference Options \(p. 12\)](#)
- [TTL Deletes \(p. 15\)](#)
- [Billable Resources \(p. 15\)](#)

When you use Amazon DocumentDB, you begin by creating a *cluster*. A cluster consists of zero or more database instances and a cluster volume that manages the data for those instances. An Amazon DocumentDB *cluster volume* is a virtual database storage volume that spans multiple Availability Zones. Each Availability Zone has a copy of the cluster data.

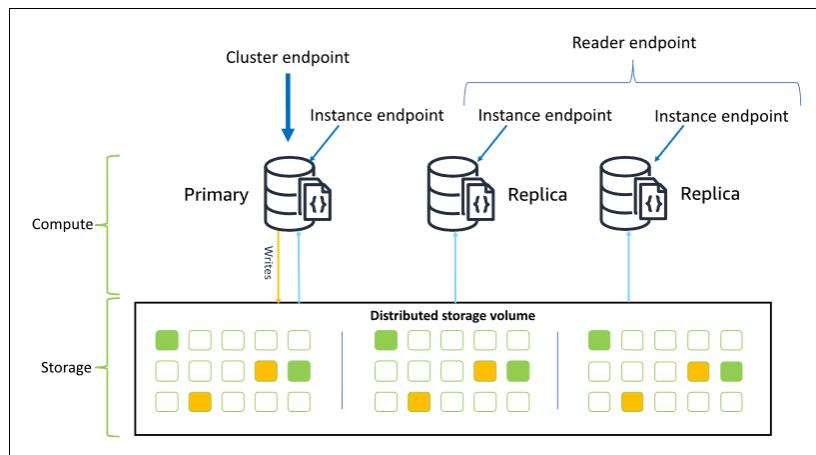
An Amazon DocumentDB cluster consists of two components:

- **Cluster volume**—Uses a cloud-native storage service to replicate data six ways across three Availability Zones, providing highly durable and available storage. An Amazon DocumentDB cluster has exactly one cluster volume, which can store up to 64 TiB of data.
- **Instances**—Provide the processing power for the database, writing data to, and reading data from, the cluster storage volume. An Amazon DocumentDB cluster can have 0–16 instances.

Instances serve one of two roles:

- **Primary instance**—Supports read and write operations, and performs all the data modifications to the cluster volume. Each Amazon DocumentDB cluster has one primary instance.
- **Replica instance**—Supports only read operations. An Amazon DocumentDB cluster can have up to 15 replicas in addition to the primary instance. Having multiple replicas enables you to distribute read workloads. In addition, by placing replicas in separate Availability Zones, you also increase your cluster availability.

The following diagram illustrates the relationship between the cluster volume, the primary instance, and replicas in an Amazon DocumentDB cluster:



Cluster instances do not need to be of the same instance class, and they can be provisioned and terminated as desired. This architecture lets you scale your cluster's compute capacity independently of its storage.

When your application writes data to the primary instance, the primary executes a durable write to the cluster volume. It then replicates the state of that write (not the data) to each active replica. Amazon DocumentDB replicas do not participate in processing writes, and thus Amazon DocumentDB replicas are advantageous for read scaling. Reads from Amazon DocumentDB replicas are eventually consistent with minimal replica lag—usually less than 100 milliseconds after the primary instance writes the data. Reads from the replicas are guaranteed to be read in the order in which they were written to the primary. Replica lag varies depending on the rate of data change, and periods of high write activity might increase the replica lag. For more information, see the [ReplicationLag metrics at Amazon DocumentDB Metrics \(p. 414\)](#).

Amazon DocumentDB Endpoints

Amazon DocumentDB provides multiple connection options to serve a wide range of use cases. To connect to an instance in an Amazon DocumentDB cluster, you specify the instance's endpoint. An *endpoint* is a host address and a port number, separated by a colon.

We recommend that you connect to your cluster using the cluster endpoint and in replica set mode (see [Connecting to Amazon DocumentDB as a Replica Set \(p. 496\)](#)) unless you have a specific use case for connecting to the reader endpoint or an instance endpoint. To route requests to your replicas, choose a driver read preference setting that maximizes read scaling while meeting your application's read consistency requirements. The `secondaryPreferred` read preference enables replica reads and frees up the primary instance to do more work.

The following endpoints are available from an Amazon DocumentDB cluster.

Cluster Endpoint

The *cluster endpoint* connects to your cluster's current primary instance. The cluster endpoint can be used for read and write operations. An Amazon DocumentDB cluster has exactly one cluster endpoint.

The cluster endpoint provides failover support for read and write connections to the cluster. If your cluster's current primary instance fails, and your cluster has at least one active read replica, the cluster endpoint automatically redirects connection requests to a new primary instance. When connecting to your Amazon DocumentDB cluster, we recommend that you connect to your cluster using the cluster endpoint and in replica set mode (see [Connecting to Amazon DocumentDB as a Replica Set \(p. 496\)](#)).

The following is an example Amazon DocumentDB cluster endpoint:

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

The following is an example connection string using this cluster endpoint:

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

For information about finding a cluster's endpoints, see [Finding a Cluster's Endpoints \(p. 379\)](#).

Reader Endpoint

The *reader endpoint* load balances read-only connections across all available replicas in your cluster. Attempting to perform a write operation over a connection to the reader endpoint results in an error. An Amazon DocumentDB cluster has exactly one reader endpoint.

If the cluster contains only one (primary) instance, the reader endpoint connects to the primary instance. When you add a replica instance to your Amazon DocumentDB cluster, the reader endpoint opens read-only connections to the new replica after it is active.

The following is an example reader endpoint for an Amazon DocumentDB cluster:

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

The following is an example connection string using a reader endpoint:

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

The reader endpoint load balances read-only connections, not read requests. If some reader endpoint connections are more heavily used than others, your read requests might not be equally balanced among instances in the cluster. It is recommended to distribute requests by connecting to the cluster endpoint as a replica set and utilizing the secondaryPreferred read preference option.

For information about finding a cluster's endpoints, see [Finding a Cluster's Endpoints \(p. 379\)](#).

Instance Endpoint

An *instance endpoint* connects to a specific instance within your cluster. The instance endpoint for the current primary instance can be used for read and write operations. However, attempting to perform write operations to an instance endpoint for a read replica results in an error. An Amazon DocumentDB cluster has one instance endpoint per active instance.

An instance endpoint provides direct control over connections to a specific instance for scenarios in which the cluster endpoint or reader endpoint might not be appropriate. An example use case is provisioning for a periodic read-only analytics workload. You can provision a larger-than-normal replica instance, connect directly to the new larger instance with its instance endpoint, run the analytics queries, and then terminate the instance. Using the instance endpoint keeps the analytics traffic from impacting other cluster instances.

The following is an example instance endpoint for a single instance in an Amazon DocumentDB cluster:

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

The following is an example connection string using this instance endpoint:

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

Note

An instance's role as primary or replica can change due to a failover event. Your applications should never assume that a particular instance endpoint is the primary instance. We do not recommend connecting to instance endpoints for production applications. Instead, we recommend that you connect to your cluster using the cluster endpoint and in replica set mode (see [Connecting to Amazon DocumentDB as a Replica Set \(p. 496\)](#)). For more advanced control of instance failover priority, see [Understanding Amazon DocumentDB Cluster Fault Tolerance \(p. 312\)](#).

For information about finding a cluster's endpoints, see [Finding an Instance's Endpoint \(p. 380\)](#).

Replica Set Mode

You can connect to your Amazon DocumentDB cluster endpoint in replica set mode by specifying the replica set name `rs0`. Connecting in replica set mode provides the ability to specify the Read Concern, Write Concern, and Read Preference options. For more information, see [Read Consistency \(p. 13\)](#).

The following is an example connection string connecting in replica set mode:

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

When you connect in replica set mode, your Amazon DocumentDB cluster appears to your drivers and clients as a replica set. Instances added and removed from your Amazon DocumentDB cluster are reflected automatically in the replica set configuration.

Each Amazon DocumentDB cluster consists of a single replica set with the default name `rs0`. The replica set name cannot be modified.

Connecting to the cluster endpoint in replica set mode is the recommended method for general use.

Note

All instances in an Amazon DocumentDB cluster listen on the same TCP port for connections.

TLS Support

For more details on connecting to Amazon DocumentDB using Transport Layer Security (TLS), see [Encrypting Data in Transit \(p. 147\)](#).

Amazon DocumentDB Storage

Amazon DocumentDB data is stored in a *cluster volume*, which is a single, virtual volume that uses solid state drives (SSDs). A cluster volume consists of six copies of your data, which are replicated automatically across multiple Availability Zones in a single AWS Region. This replication helps ensure that your data is highly durable, with less possibility of data loss. It also helps ensure that your cluster is more available during a failover because copies of your data already exist in other Availability Zones. These copies can continue to serve data requests to the instances in your Amazon DocumentDB cluster.

How Data Storage is Billed

Amazon DocumentDB automatically increases the size of a cluster volume as the amount of data increases. An Amazon DocumentDB cluster volume can grow to a maximum size of 64 TiB; however, you

are only charged for the space that you use in an Amazon DocumentDB cluster volume. Starting with Amazon DocumentDB 4.0, when data is removed, such as by dropping a collection or index, the overall allocated space decreases by a comparable amount. Thus, you can reduce storage charges by deleting collections, indexes, and databases that you no longer need. With Amazon DocumentDB 3.6, when data is removed, such as by dropping a collection or index, the overall allocated space remains the same. The free space is reused automatically when the data volume increases in the future.

Note

With Amazon DocumentDB 3.6, storage costs are based on the storage "high water mark" (the maximum amount that was allocated for the Amazon DocumentDB cluster at any point in time). You can manage costs by avoiding ETL practices that create large volumes of temporary information, or that load large volumes of new data prior to removing unneeded older data. If removing data from an Amazon DocumentDB cluster results in a substantial amount of allocated but unused space, resetting the high water mark requires doing a logical data dump and restore to a new cluster, using a tool such as `mongodump` or `mongorestore`. Creating and restoring a snapshot does not reduce the allocated storage because the physical layout of the underlying storage remains the same in the restored snapshot.

Note

Using utilities like `mongodump` and `mongorestore` incur I/O charges based on the sizes of the data that is being read and written to the storage volume.

For information about Amazon DocumentDB data storage and I/O pricing, see [Amazon DocumentDB \(with MongoDB compatibility\) pricing](#) and [Pricing FAQs](#).

Amazon DocumentDB Replication

In an Amazon DocumentDB cluster, each replica instance exposes an independent endpoint. These replica endpoints provide read-only access to the data in the cluster volume. They enable you to scale the read workload for your data over multiple replicated instances. They also help improve the performance of data reads and increase the availability of the data in your Amazon DocumentDB cluster. Amazon DocumentDB replicas are also failover targets and are quickly promoted if the primary instance for your Amazon DocumentDB cluster fails.

Amazon DocumentDB Reliability

Amazon DocumentDB is designed to be reliable, durable, and fault tolerant. (To improve availability, you should configure your Amazon DocumentDB cluster so that it has multiple replica instances in different Availability Zones.) Amazon DocumentDB includes several automatic features that make it a reliable database solution.

Storage Auto-Repair

Amazon DocumentDB maintains multiple copies of your data in three Availability Zones, greatly reducing the chance of losing data due to a storage failure. Amazon DocumentDB automatically detects failures in the cluster volume. When a segment of a cluster volume fails, Amazon DocumentDB immediately repairs the segment. It uses the data from the other volumes that make up the cluster volume to help ensure that the data in the repaired segment is current. As a result, Amazon DocumentDB avoids data loss and reduces the need to perform a point-in-time restore to recover from an instance failure.

Survivable Cache Warming

Amazon DocumentDB manages its page cache in a separate process from the database so that the page cache can survive independently of the database. In the unlikely event of a database failure, the page cache remains in memory. This ensures that the buffer pool is warmed with the most current state when the database restarts.

Crash Recovery

Amazon DocumentDB is designed to recover from a crash almost instantaneously, and to continue serving your application data. Amazon DocumentDB performs crash recovery asynchronously on parallel threads so that your database is open and available almost immediately after a crash.

Resource Governance

Amazon DocumentDB safeguards resources that are needed to run critical processes in the service, such as health checks. To do this, and when an instance is experiencing high memory pressure, Amazon DocumentDB will throttle requests. As a result, some operations may be queued to wait for the memory pressure to subside. If memory pressure continues, queued operations may timeout. You can monitor whether or not the service throttling operations due to low memory with the following CloudWatch metrics: `LowMemThrottleQueueDepth`, `LowMemThrottleMaxQueueDepth`, `LowMemNumOperationsThrottled`, `LowMemNumOperationsTimedOut`. For more information, see Monitoring Amazon DocumentDB with CloudWatch. If you see sustained memory pressure on your instance as a result of the LowMem CloudWatch metrics, we advise that you scale-up your instance to provide additional memory for your workload.

Read Preference Options

Amazon DocumentDB uses a cloud-native shared storage service that replicates data six times across three Availability Zones to provide high levels of durability. Amazon DocumentDB does not rely on replicating data to multiple instances to achieve durability. Your cluster's data is durable whether it contains a single instance or 15 instances.

Write Durability

Amazon DocumentDB uses a unique, distributed, fault-tolerant, self-healing storage system. This system replicates six copies ($V=6$) of your data across three AWS Availability Zones to provide high availability and durability. When writing data, Amazon DocumentDB ensures that all writes are durably recorded on a majority of nodes before acknowledging the write to the client. If you are running a three-node MongoDB replica set, using a write concern of `{w:3, j:true}` would yield the best possible configuration when comparing with Amazon DocumentDB.

Writes to an Amazon DocumentDB cluster must be processed by the cluster's writer instance. Attempting to write to a reader results in an error. An acknowledged write from an Amazon DocumentDB primary instance is durable, and can't be rolled back. Amazon DocumentDB is highly durable by default and doesn't support a non-durable write option. You can't modify the durability level (that is, write concern). Amazon DocumentDB ignores `w=anything` and is effectively `w:3` and `j: true`. You cannot reduce it.

Because storage and compute are separated in the Amazon DocumentDB architecture, a cluster with a single instance is highly durable. Durability is handled at the storage layer. As a result, an Amazon DocumentDB cluster with a single instance and one with three instances achieve the same level of durability. You can configure your cluster to your specific use case while still providing high durability for your data.

Writes to an Amazon DocumentDB cluster are atomic within a single document.

Amazon DocumentDB does not support the `wtimeout` option and will not return an error if a value is specified. Writes to the primary Amazon DocumentDB instance are guaranteed not to block indefinitely.

Read Isolation

Reads from an Amazon DocumentDB instance only return data that is durable before the query begins. Reads never return data modified after the query begins execution nor are dirty reads possible under any circumstances.

Read Consistency

Data read from an Amazon DocumentDB cluster is durable and will not be rolled back. You can modify the read consistency for Amazon DocumentDB reads by specifying the read preference for the request or connection. Amazon DocumentDB does not support a non-durable read option.

Reads from an Amazon DocumentDB cluster's primary instance are strongly consistent under normal operating conditions and have read-after-write consistency. If a failover event occurs between the write and subsequent read, the system can briefly return a read that is not strongly consistent. All reads from a read replica are eventually consistent and return the data in the same order, and often with less than 100 ms replica lag.

Amazon DocumentDB Read Preferences

Amazon DocumentDB supports setting a read preference option only when reading data from the cluster endpoint in replica set mode. Setting a read preference option affects how your MongoDB client or driver routes read requests to instances in your Amazon DocumentDB cluster. You can set read preference options for a specific query, or as a general option in your MongoDB driver. (Consult your client or driver's documentation for instructions on how to set a read preference option.)

If your client or driver is not connecting to an Amazon DocumentDB cluster endpoint in replica set mode, the result of specifying a read preference is undefined.

Amazon DocumentDB does not support setting *tag sets* as a read preference.

Supported Read Preference Options

- **primary**—Specifying a primary read preference helps ensure that all reads are routed to the cluster's primary instance. If the primary instance is unavailable, the read operation fails. A primary read preference yields read-after-write consistency and is appropriate for use cases that prioritize read-after-write consistency over high availability and read scaling.

The following example specifies a primary read preference:

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

- **primaryPreferred**—Specifying a primaryPreferred read preference routes reads to the primary instance under normal operation. If there is a primary failover, the client routes requests to a replica. A primaryPreferred read preference yields read-after-write consistency during normal operation, and eventually consistent reads during a failover event. A primaryPreferred read preference is appropriate for use cases that prioritize read-after-write consistency over read scaling, but still require high availability.

The following example specifies a primaryPreferred read preference:

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

- **secondary**—Specifying a secondary read preference ensures that reads are only routed to a replica, never the primary instance. If there are no replica instances in a cluster, the read request fails. A secondary read preference yields eventually consistent reads and is appropriate for use cases that prioritize primary instance write throughput over high availability and read-after-write consistency.

The following example specifies a secondary read preference:

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

- **secondaryPreferred**—Specifying a secondaryPreferred read preference ensures that reads are routed to a read replica when one or more replicas are active. If there are no active replica instances in a cluster, the read request is routed to the primary instance. A secondaryPreferred read preference yields eventually consistent reads when the read is serviced by a read replica. It yields read-after-write consistency when the read is serviced by the primary instance (barring failover events). A secondaryPreferred read preference is appropriate for use cases that prioritize read scaling and high availability over read-after-write consistency.

The following example specifies a secondaryPreferred read preference:

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

- **nearest**—Specifying a nearest read preference routes reads based solely on the measured latency between the client and all instances in the Amazon DocumentDB cluster. A nearest read preference yields eventually consistent reads when the read is serviced by a read replica. It yields read-after-write consistency when the read is serviced by the primary instance (barring failover events). A nearest read preference is appropriate for use cases that prioritize achieving the lowest possible read latency and high availability over read-after-write consistency and read scaling.

The following example specifies a nearest read preference:

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

High Availability

Amazon DocumentDB supports highly available cluster configurations by using replicas as failover targets for the primary instance. If the primary instance fails, an Amazon DocumentDB replica is promoted as the new primary, with a brief interruption during which read and write requests made to the primary instance fail with an exception.

If your Amazon DocumentDB cluster doesn't include any replicas, the primary instance is re-created during a failure. However, promoting an Amazon DocumentDB replica is much faster than re-creating the primary instance. So we recommend that you create one or more Amazon DocumentDB replicas as failover targets.

Replicas that are intended for use as failover targets should be of the same instance class as the primary instance. They should be provisioned in different Availability Zones from the primary. You can control which replicas are preferred as failover targets. For best practices on configuring Amazon DocumentDB for high availability, see [Understanding Amazon DocumentDB Cluster Fault Tolerance \(p. 312\)](#).

Scaling Reads

Amazon DocumentDB replicas are ideal for read scaling. They are fully dedicated to read operations on your cluster volume, that is, replicas do not process writes. Data replication happens within the cluster volume and not between instances. So each replica's resources are dedicated to processing your queries, not replicating and writing data.

If your application needs more read capacity, you can add a replica to your cluster quickly (usually in less than ten minutes). If your read capacity requirements diminish, you can remove unneeded replicas. With Amazon DocumentDB replicas, you pay only for the read capacity that you need.

Amazon DocumentDB supports client-side read scaling through the use of Read Preference options. For more information, see [Amazon DocumentDB Read Preferences \(p. 13\)](#).

TTL Deletes

Deletes from a TTL index area achieved via a background process are best effort and are not guaranteed within a specific timeframe. Factors like instance size, instance resource utilization, document size, and overall throughput can affect the timing of a TTL delete.

When the TTL monitor deletes your documents, each deletion incurs IO costs, which will increase your bill. If throughput and TTL delete rates increase, you should expect an increase in your bill due to increase IO usage.

When you create a TTL index on an existing collection, you must delete all expired documents before creating the index. The current TTL implementation is optimized for deleting a small fraction of documents in the collection, which is typical if TTL was enabled on the collection from the start, and may result in higher IOPS than necessary if a large number of documents need to be deleted at one go.

If you do not want to create a TTL index to delete documents, you can instead segment documents into collections based on time, and simply drop those collections when the documents are no longer needed. For example: you can create one collection per week and drop it without incurring IO costs. This can be significantly more cost effective than using a TTL index.

Billable Resources

Identifying Billable Amazon DocumentDB Resources

As a fully managed database service, Amazon DocumentDB charges for instances, storage, I/Os, backups, and data transfer. For more information, see [Amazon DocumentDB \(with MongoDB compatibility\) pricing](#).

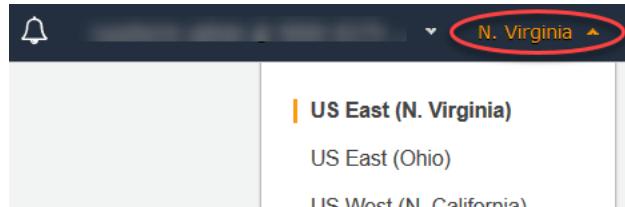
To discover billable resources in your account and potentially delete the resources, you can use the AWS Management Console or AWS CLI.

Using the AWS Management Console

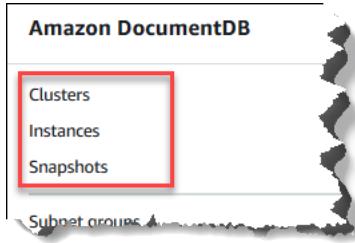
Using the AWS Management Console, you can discover the Amazon DocumentDB clusters, instances, and snapshots that you have provisioned for a given AWS Region.

To discover clusters, instances, and snapshots

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. To discover billable resources in a Region other than your default Region, in the upper-right corner of the screen, choose the AWS Region that you want to search.



3. In the navigation pane, choose the type of billable resource that you're interested in: **Clusters**, **Instances**, or **Snapshots**.



4. All your provisioned clusters, instances, or snapshots for the Region are listed in the right pane. You will be charged for clusters, instances, and snapshots.

Using the AWS CLI

Using the AWS CLI, you can discover the Amazon DocumentDB clusters, instances, and snapshots that you have provisioned for a given AWS Region.

To discover clusters and instances

The following code lists all your clusters and instances for the specified Region. If you want to search for clusters and instances in your default Region, you can omit the `--region` parameter.

Example

For Linux, macOS, or Unix:

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

For Windows:

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

Output from this operation looks something like the following.

```
"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",
```

To discover snapshots

The following code lists all your snapshots for the specified Region. If you want to search for snapshots in your default Region, you can omit the `--region` parameter.

For Linux, macOS, or Unix:

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

For Windows:

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

Output from this operation looks something like the following.

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

You only need to delete manual snapshots. Automated snapshots are deleted when you delete the cluster.

Deleting Unwanted Billable Resources

To delete a cluster, you must first delete all the instances in the cluster.

- To delete instances, see [Deleting an Amazon DocumentDB Instance \(p. 331\)](#).

Important

Even if you delete the instances in a cluster, you are still billed for the storage and backup usage associated with that cluster. To stop all charges, you must also delete your cluster and manual snapshots.

- To delete clusters, see [Deleting an Amazon DocumentDB Cluster \(p. 296\)](#).
- To delete manual snapshots, see [Deleting a Cluster Snapshot \(p. 245\)](#).

What is a Document Database?

Some developers don't think of their data model in terms of normalized rows and columns. Typically, in the application tier, data is represented as a JSON document because it is more intuitive for developers to think of their data model as a document.

The popularity of document databases has grown because they let you persist data in a database by using the same document model format that you use in your application code. Document databases provide powerful and intuitive APIs for flexible and agile development.

Topics

- [Document Database Use Cases \(p. 17\)](#)
- [Understanding Documents \(p. 18\)](#)
- [Working with Documents \(p. 22\)](#)

Document Database Use Cases

Your use case drives whether you need a document database or some other type of database for managing your data. Document databases are useful for workloads that require a flexible schema

for fast, iterative development. The following are some examples of use cases for which document databases can provide significant advantages:

Topics

- [User Profiles \(p. 18\)](#)
- [Real-Time Big Data \(p. 18\)](#)
- [Content Management \(p. 18\)](#)

User Profiles

Because document databases have a flexible schema, they can store documents that have different attributes and data values. Document databases are a practical solution to online profiles in which different users provide different types of information. Using a document database, you can store each user's profile efficiently by storing only the attributes that are specific to each user.

Suppose that a user elects to add or remove information from their profile. In this case, their document could be easily replaced with an updated version that contains any recently added attributes and data or omits any newly omitted attributes and data. Document databases easily manage this level of individuality and fluidity.

Real-Time Big Data

Historically, the ability to extract information from operational data was hampered by the fact that operational databases and analytical databases were maintained in different environments—operational and business/reporting respectively. Being able to extract operational information in real time is critical in a highly competitive business environment. By using document databases, a business can store and manage operational data from any source and concurrently feed the data to the BI engine of choice for analysis. There is no requirement to have two environments.

Content Management

To effectively manage content, you must be able to collect and aggregate content from a variety of sources, and then deliver it to the customer. Due to their flexible schema, document databases are perfect for collecting and storing any type of data. You can use them to create and incorporate new types of content, including user-generated content, such as images, comments, and videos.

Understanding Documents

Document databases are used for storing semistructured data as a document—rather than normalizing data across multiple tables, each with a unique and fixed structure, as in a relational database. Documents stored in a document database use nested key-value pairs to provide the document's structure or schema. However, different types of documents can be stored in the same document database, thus meeting the requirement for processing similar data that is in different formats. For example, because each document is self-describing, the JSON-encoded documents for an online store that are described in the topic [Example Documents in a Document Database \(p. 20\)](#) can be stored in the same document database.

Topics

- [SQL vs. Nonrelational Terminology \(p. 19\)](#)
- [Simple Documents \(p. 19\)](#)
- [Embedded Documents \(p. 19\)](#)
- [Example Documents in a Document Database \(p. 20\)](#)

- [Understanding Normalization in a Document Database \(p. 21\)](#)

SQL vs. Nonrelational Terminology

The following table compares terminology used by document databases (MongoDB) with terminology used by SQL databases.

SQL	MongoDB
Table	Collection
Row	Document
Column	Field
Primary key	ObjectId
Index	Index
View	View
Nested table or object	Embedded document
Array	Array

Simple Documents

All documents in a document database are self-describing. This documentation uses JSON-like formatted documents, although you can use other means of encoding.

A simple document has one or more fields that are all at the same level within the document. In the following example, the fields SSN, LName, FName, DOB, Street, City, State-Province, PostalCode, and Country are all siblings within the document.

```
{  
    "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"  
}
```

When information is organized in a simple document, each field is managed individually. To retrieve a person's address, you must retrieve Street, City, State-Province, PostalCode, and Country as individual data items.

Embedded Documents

A complex document organizes its data by creating embedded documents within the document. Embedded documents help manage data in groupings and as individual data items, whichever is more efficient in a given case. Using the preceding example, you could embed an Address document in the main document. Doing this results in the following document structure:

```
{  
    "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"  
    }  
}
```

You can now access the data in the document as individual fields ("SSN" :), as an embedded document ("Address" :), or as a member of an embedded document ("Address":{ "Street":{} }).

Example Documents in a Document Database

As stated earlier, because each document in a document database is self-describing, the structure of documents within a document database can be different from one another. The following two documents, one for a book and another for a periodical, are different structurally. Yet both of them can be in the same document database.

The following is a sample book document:

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

The following is a sample periodical document with two articles:

```
{  
    "_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": "How to Optimize Your Document Database",  
            "Author":  
            {  
                "LName": "Smith",  
                "FName": "John"  
            }  
        }  
    ]  
}
```

```

        "Title": "Databases for Online Solutions",
        "Author":
        {
            "LName": "Stiles",
            "FName": "John"
        }
    ],
    "Type": "periodical"
}

```

Compare the structure of these two documents. With a relational database, you need either separate "periodical" and "books" tables, or a single table with unused fields, such as "Publication," "Issue," "Articles," and "MI," as null values. Because document databases are semistructured, with each document defining its own structure, these two documents can coexist in the same document database with no null fields. Document databases are good at dealing with sparse data.

Developing against a document database enables quick, iterative development. This is because you can change the data structure of a document dynamically, without having to change the schema for the entire collection. Document databases are well suited for agile development and dynamically changing environments.

Understanding Normalization in a Document Database

Document databases are not normalized; data found in one document can be repeated in another document. Further, some data discrepancies can exist between documents. For example, consider the scenario in which you make a purchase at an online store and all the details of your purchases are stored in a single document. The document might look something like the following JSON document:

```

{
    "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"
}

```

All this information is stored as a document in a transaction collection. Later, you realize that you forgot to purchase one item. So you again log on to the same store and make another purchase, which is also stored as another document in the transaction collection.

```
{  
    "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"  
}
```

Notice the redundancy between these two documents—your name and shopper ID (and, if you used the same credit card, your credit card information). But that's okay because storage is inexpensive, and each document completely records a single transaction that can be retrieved quickly with a simple key-value query that requires no joins.

There is also an apparent discrepancy between the two documents—your credit card information. This is only an apparent discrepancy because it is likely that you used a different credit card for each purchase. Each document is accurate for the transaction that it documents.

Working with Documents

As a document database, Amazon DocumentDB makes it easy to store, query, and index JSON data. In Amazon DocumentDB, a collection is analogous to a table in a relational database, except there is no single schema enforced upon all documents. Collections let you group similar documents together while keeping them all in the same database, without requiring that they be identical in structure.

Using the example documents from earlier sections, it is likely that you'd have collections for `reading_material` and `office_supplies`. It is the responsibility of your software to enforce which collection a document belongs in.

The following examples use the MongoDB API to show how to add, query, update, and delete documents.

Topics

- [Adding Documents \(p. 22\)](#)
- [Querying Documents \(p. 24\)](#)
- [Updating Documents \(p. 26\)](#)
- [Deleting Documents \(p. 30\)](#)

Adding Documents

In Amazon DocumentDB, a database is created when first you add a document to a collection. In this example, you are creating a collection named `example` in the `test` database, which is the default

database when you connect to a cluster. Because the collection is implicitly created when the first document is inserted, there is no error checking of the collection name. Therefore, a typo in the collection name, such as eexample instead of example, will create and add the document to eexample collection rather than the intended collection. Error checking must be handled by your application.

The following examples use the MongoDB API to add documents.

Topics

- [Adding a Single Document \(p. 23\)](#)
- [Adding Multiple Documents \(p. 23\)](#)

Adding a Single Document

To add a single document to a collection, use the `insertOne({})` operation with the document that you want added to the collection.

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

Output from this operation looks something like the following (JSON format).

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

Adding Multiple Documents

To add multiple documents to a collection, use the `insertMany([{}], ... , {})` operation with a list of the documents that you want added to the collection. Although the documents in this particular list have different schemas, they can all be added to the same collection.

```
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,  
        "OrderQty": 36  
    }  
}  
]  
}
```

Output from this operation looks something like the following (JSON format).

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

Querying Documents

At times, you might need to look up your online store's inventory so that customers can see and purchase what you're selling. Querying a collection is relatively easy, whether you want all documents in the collection or only those documents that satisfy a particular criterion.

To query for documents, use the `find()` operation. The `find()` command has a single document parameter that defines the criteria to use in choosing the documents to return. The output from `find()` is a document formatted as a single line of text with no line breaks. To format the output document for easier reading, use `find().pretty()`. All the examples in this topic use `.pretty()` to format the output.

Use the four documents you inserted into the example collection in the preceding two exercises — `insertOne()` and `insertMany()`.

Topics

- [Retrieving All Documents in a Collection \(p. 24\)](#)
- [Retrieving Documents That Match a Field Value \(p. 25\)](#)
- [Retrieving Documents That Match an Embedded Document \(p. 25\)](#)
- [Retrieving Documents That Match a Field Value in an Embedded Document \(p. 25\)](#)
- [Retrieving Documents That Match an Array \(p. 25\)](#)
- [Retrieving Documents That Match a Value in an Array \(p. 25\)](#)
- [Retrieving Documents Using Operators \(p. 26\)](#)

Retrieving All Documents in a Collection

To retrieve all the documents in your collection, use the `find()` operation with an empty query document.

The following query returns all documents in the example collection.

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

Retrieving Documents That Match a Field Value

To retrieve all documents that match a field and value, use the `find()` operation with a query document that identifies the fields and values to match.

Using the preceding documents, this query returns all documents where the "Item" field equals "Pen".

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

Retrieving Documents That Match an Embedded Document

To find all the documents that match an embedded document, use the `find()` operation with a query document that specifies the embedded document name and all the fields and values for that embedded document.

When matching an embedded document, the document's embedded document must have the same name as in the query. In addition, the fields and values in the embedded document must match the query.

The following query returns only the "Poster Paint" document. This is because the "Pen" has different values for "OnHand" and "MinOnHand", and "Spray Paint" has one more field (`OrderQty`) than the query document.

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

Retrieving Documents That Match a Field Value in an Embedded Document

To find all the documents that match an embedded document, use the `find()` operation with a query document that specifies the embedded document name and all the fields and values for that embedded document.

Given the preceding documents, the following query uses "dot notation" to specify the embedded document and fields of interest. Any document that matches these are returned, regardless of what other fields might be present in the embedded document. The query returns "Poster Paint" and "Spray Paint" because they both match the specified fields and values.

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

Retrieving Documents That Match an Array

To find all documents that match an array, use the `find()` operation with the array name that you are interested in and all the values in that array. The query returns all documents that have an array with that name in which the array values are identical to and in the same order as in the query.

The following query returns only the "Pen" because the "Poster Paint" has an additional color (White), and "Spray Paint" has the colors in a different order.

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

Retrieving Documents That Match a Value in an Array

To find all the documents that have a particular array value, use the `find()` operation with the array name and the value that you're interested in.

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

The preceding operation returns all three documents because each of them has an array named `Colors` and the value "Red" somewhere in the array. If you specify the value "White," the query would only return "Poster Paint."

Retrieving Documents Using Operators

The following query returns all documents where the "Inventory.OnHand" value is less than 50.

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

For a listing of supported query operators, see [Query and Projection Operators \(p. 106\)](#).

Updating Documents

Typically, your documents are not static and are updated as part of your application workflows. The following examples show some of the ways that you can update documents.

To update an existing document, use the `update()` operation. The `update()` operation has two document parameters. The first document identifies which document or documents to update. The second document specifies the updates to make.

When you update an existing field — whether that field is a simple field, an array, or an embedded document — you specify the field name and its values. At the end of the operation, it is as though the field in the old document has been replaced by the new field and values.

Topics

- [Updating the Values of an Existing Field \(p. 26\)](#)
- [Adding a New Field \(p. 28\)](#)
- [Replacing an Embedded Document \(p. 28\)](#)
- [Inserting New Fields into an Embedded Document \(p. 29\)](#)
- [Removing a Field from a Document \(p. 29\)](#)
- [Removing a Field from Multiple Documents \(p. 30\)](#)

Updating the Values of an Existing Field

Use the following four documents that you added earlier for the following updating operations.

```
{  
    "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
    }
  }
}
```

To update a simple field

To update a simple field, use `update()` with `$set` to specify the field name and new value. The following example changes the `Item` from "Pen" to "Gel Pen".

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

Results from this operation look something like the following.

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

To update an array

The following example replaces the existing array of colors with a new array that includes Orange and drops White from the list of colors. The new list of colors is in the order specified in the `update()` operation.

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

Results from this operation look something like the following.

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

```
}
```

Adding a New Field

To modify a document by adding one or more new fields, use the `update()` operation with a query document that identifies the document to insert into and the new fields and values to insert using the `$set` operator.

The following example adds the field `UnitPrice` with the value `3.99` to the Spray Paints document. Note that the value `3.99` is numeric and not a string.

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

Results from this operation look something like the following (JSON format).

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

Replacing an Embedded Document

To modify a document by replacing an embedded document, use the `update()` operation with documents that identify the embedded document and its new fields and values using the `$set` operator.

Given the following document.

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

To replace an embedded document

The following example replaces the current `Date` document with a new one that has only the fields `Month` and `Day`; `Year` has been eliminated.

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

Results from this operation look something like the following.

```
{
```

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

Inserting New Fields into an Embedded Document

To add fields to an embedded document

To modify a document by adding one or more new fields to an embedded document, use the `update()` operation with documents that identify the embedded document and "dot notation" to specify the embedded document and the new fields and values to insert using the `$set` operator.

Given the following document, the following code uses "dot notation" to insert the `Year` and `DoW` fields to the embedded `Date` document, and `Words` into the parent document.

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

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

Results from this operation look something like the following.

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

Removing a Field from a Document

To modify a document by removing a field from the document, use the `update()` operation with a query document that identifies the document to remove the field from, and the `$unset` operator to specify the field to remove.

The following example removes the `Words` field from the preceding document.

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

Results from this operation look something like the following.

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

Removing a Field from Multiple Documents

To modify a document by removing a field from multiple documents, use the `update()` operation with the `$unset` operator and the `multi` option set to `true`.

The following example removes the `Inventory` field from all documents in the example collection. If a document does not have the `Inventory` field, no action is taken on that document. If `multi: true` is omitted, the action is performed only on the first document that meets the criterion.

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

Deleting Documents

To remove a document from your database, use the `remove()` operation, specifying which document to remove. The following code removes "Gel Pen" from your example collection.

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

To remove all documents from your database, use the `remove()` operation with an empty query, as shows following.

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

Get Started with Amazon DocumentDB

There are many ways to connect and get started with Amazon DocumentDB. We created this guide because we found this way to be the quickest, simplest and easiest way for users to get started using our powerful document database. This guide utilizes [AWS Cloud9](#), a web-based terminal to connect and query your Amazon DocumentDB cluster using the mongo shell directly from the AWS Management Console. New customers who are eligible for the AWS Free Tier can use Amazon DocumentDB and AWS Cloud9 for free. If your AWS Cloud9 environment or Amazon DocumentDB cluster makes use of resources beyond the free tier, you are charged the normal AWS rates for those resources. This guide will get you started with Amazon DocumentDB in less than 15 minutes.

Topics

- [Prerequisites \(p. 31\)](#)
- [Step 1: Create an AWS Cloud9 environment \(p. 32\)](#)
- [Step 2: Create a security group \(p. 35\)](#)
- [Step 3: Create an Amazon DocumentDB cluster \(p. 37\)](#)
- [Step 4: Install the mongo shell \(p. 40\)](#)
- [Step 5: Connect to your Amazon DocumentDB cluster \(p. 41\)](#)
- [Step 6: Insert and query data \(p. 41\)](#)
- [Step 7: Explore \(p. 43\)](#)

If you would rather connect to your Amazon DocumentDB from your local machine by creating an SSH connection to an Amazon EC2 instance, please see the [Connect with EC2 instructions](#)

Prerequisites

Before you create your first Amazon DocumentDB cluster, you must do the following:

Create an Amazon Web Services (AWS) account

Before you can begin using Amazon DocumentDB, you must have an Amazon Web Services (AWS) account. The AWS account is free. You pay only for the services and resources that you use.

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open <https://portal.aws.amazon.com/billing/signup>.
2. Follow the online instructions.

Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

When you sign up for an AWS account, an **AWS account root user** is created. The root user has access to all AWS services and resources in the account. As a security best practice, [assign administrative access to an administrative user](#), and use only the root user to perform [tasks that require root user access](#).

Set up the needed AWS Identity and Access Management (IAM) permissions.

Access to manage Amazon DocumentDB resources such as clusters, instances, and cluster parameter groups requires credentials that AWS can use to authenticate your requests. For more information, see [Identity and Access Management in Amazon DocumentDB \(p. 153\)](#).

1. In the search bar of the AWS Management Console, type in IAM and select **IAM** in the drop down menu that appears.
2. Once you're in the IAM console, select **Users** from the navigation pane.
3. Select your username.
4. Click the button **Add permissions**.
5. Select **Attach existing policies directly**.
6. Type **AmazonDocDBFullAccess** in the search bar and select it once it appears in the search results.
7. Click the blue button at the bottom that says **Next: Review**.
8. Click the blue button at the bottom that says **Add permissions**.

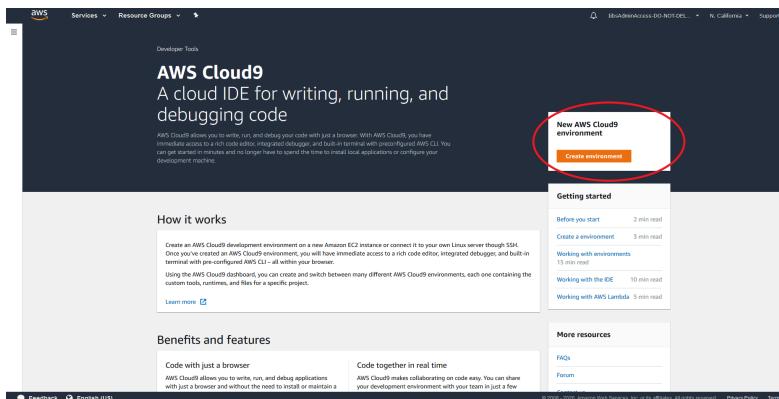
Create an Amazon Virtual Private Cloud (Amazon VPC)

This step is only necessary if you don't already have a default Amazon VPC. If you don't, then complete step 1 of the [Getting Started with Amazon VPC](#) in the *Amazon VPC User Guide*. This will take less than five minutes.

Step 1: Create an AWS Cloud9 environment

AWS Cloud9 provides a web-based terminal that you can use to connect to and query your Amazon DocumentDB cluster using the mongo shell.

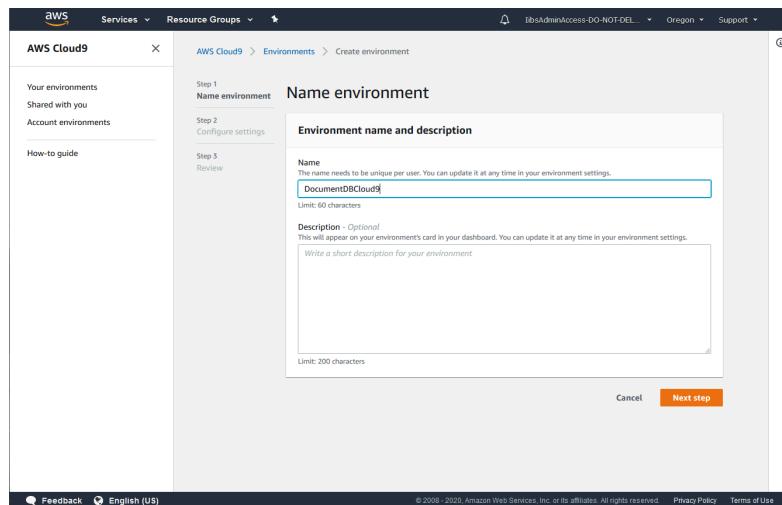
1. From the AWS Management Console navigate to the AWS Cloud9 console and choose **Create environment**.



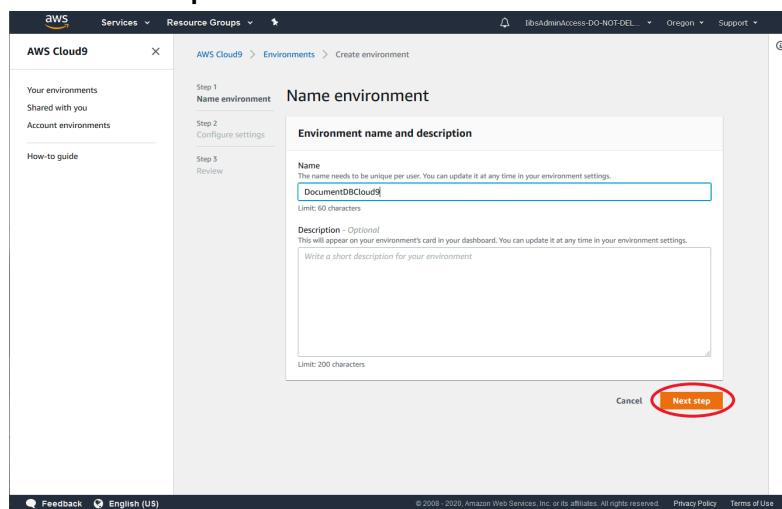
2. In the **Environment name and description** section, in the **Name** field, enter DocumentDBCloud9.

Amazon DocumentDB Developer Guide

Step 1: Create an AWS Cloud9 environment



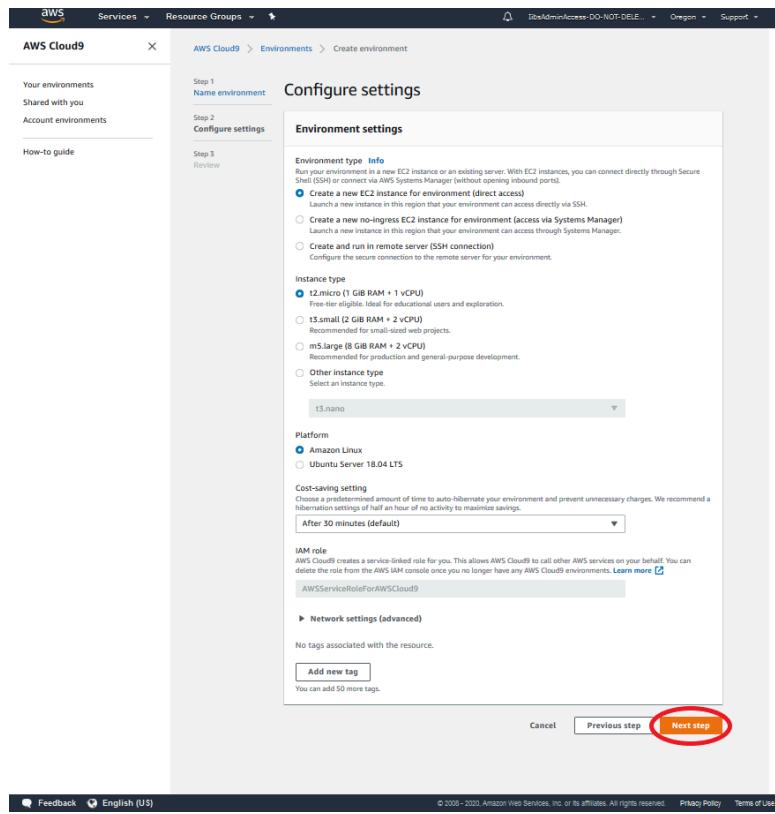
3. Choose Next step.



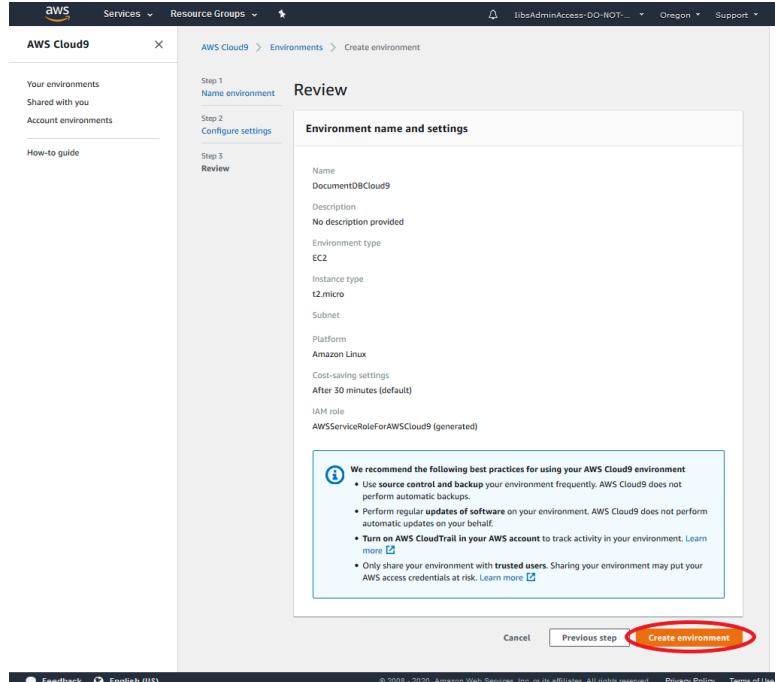
4. In the Configure settings section, choose Next step.

Amazon DocumentDB Developer Guide

Step 1: Create an AWS Cloud9 environment



5. In the Review section, choose **Create environment**.



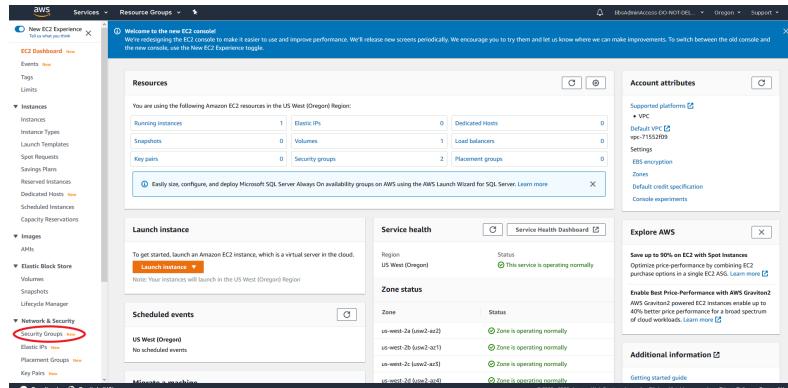
Note

The provisioning of the AWS Cloud9 environment can take up to three minutes.

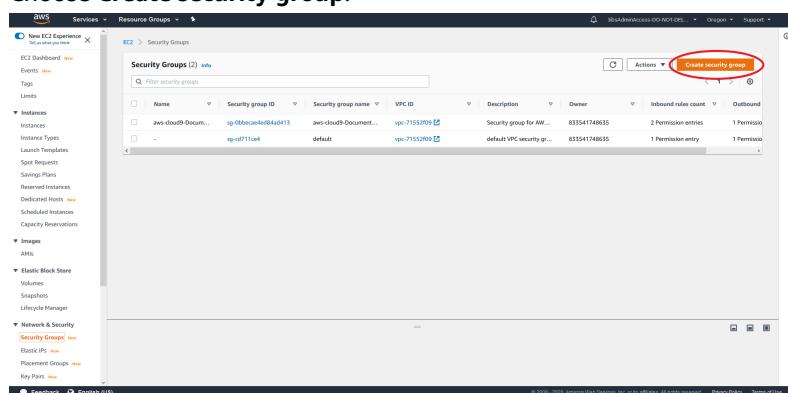
Step 2: Create a security group

This security group will enable you to connect to your Amazon DocumentDB cluster from your AWS Cloud9 environment.

- On the [Amazon EC2 Management Console](#), under Network and Security, choose **Security groups**.



- Choose **Create security group**.



- In the **Basic details** box, input the following for each field:

- For **Security group name**, enter `demoDocDB`.
- For **Description**, enter a description.
- For **VPC**, accept the usage of your default VPC.

The screenshot shows the 'Create security group' wizard. In the 'Basic details' step, the 'Security group name' field contains 'demoDocDB' and the 'Description' field contains 'For this awesome Get Started Guide'. The 'VPC' dropdown is set to 'vpc-71152b09'. The 'Inbound rules' and 'Outbound rules' sections are shown below.

4. In the Inbound rules section, choose Add rule.

A screenshot of the AWS EC2 Security Groups 'Create security group' page. The 'Inbound rules' section shows a single row with 'Type: All traffic', 'Protocol: All', 'Port range: All', 'Source: Custom', and 'Destination: 0.0.0.0/0'. A red circle highlights the 'Add rule' button below the table.

5. For Type, choose Custom TCP Rule.

A screenshot of the AWS EC2 Security Groups 'Create security group' page. The 'Inbound rules' section shows a row with 'Type: Custom TCP', 'Protocol: TCP', 'Port range: 0', 'Source: Custom', and 'Destination: 0.0.0.0/0'. A red circle highlights the 'Type' dropdown menu which is set to 'Custom TCP'.

6. For Port range, enter 27017.

A screenshot of the AWS EC2 Security Groups 'Create security group' page. The 'Inbound rules' section shows a row with 'Type: Custom TCP', 'Protocol: TCP', 'Port range: 27017', 'Source: Custom', and 'Destination: 0.0.0.0/0'. A red circle highlights the 'Port range' input field which contains '27017'.

7. The source is the security group for the AWS Cloud9 environment you just created. To see a list of available security groups, enter cloud9 in the destination field. Choose the security group with the name aws-cloud9-<environment name>.

AWS Services > Security Groups > Create security group

Create security group

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

Basic details

Security group name: **docsDbDB**
Name cannot be edited after creation.

Description: **For this awesome Get Started Guide!**

VPC: **vpc-7115209**

Inbound rules

Type: **Custom TCP**, Protocol: **TCP**, Port range: **27017**, Source: **Custom**. A red circle highlights the 'Custom' dropdown menu. Destination: **Custom**, Description: **aws-cloud9**.

Outbound rules

Type: **All traffic**, Protocol: **All**, Port range: **All**, Destination: **Custom**, Description: **aws-cloud9**.

Feedback English (US)

8. Accept all other defaults and choose **Create security group.**

AWS Services > Security Groups > Create security group

Inbound rules

Type: **Custom TCP**, Protocol: **TCP**, Port range: **27017**, Source: **Custom**, Destination: **Custom**, Description: **aws-cloud9**.

Outbound rules

Type: **All traffic**, Protocol: **All**, Port range: **All**, Destination: **Custom**, Description: **aws-cloud9**.

Tags - optional

No tags associated with the resource.

Add new tag

Cancel **Create security group**

Feedback English (US)

Note

Port 27017 is the default port for Amazon DocumentDB.

Step 3: Create an Amazon DocumentDB cluster

In this step you will create an Amazon DocumentDB cluster using the security group you created in the previous step.

1. On the Amazon DocumentDB management console, under Clusters, choose **Create.**

DocumentDB > Clusters

Clusters (2)

Actions **Create**

Cluster identifier	Role	Engine version	Region & AZ	Status	Size	Maintenance
docdb-cloud9-getstarted	Cluster	3.6.0	us-east-1	available	1 Instance	None
docdb-cloud9-getstarted	Primary	3.6.0	us-east-1f	available	db.r5.large	None
robo3t	Cluster	3.6.0	us-east-1	available	1 Instance	None
robo3t	Primary	3.6.0	us-east-1d	available	db.r5.large	None

2. On the Create Amazon DocumentDB cluster page, in the **Configuration section, choose **1** for **Number of instances**. Choosing one instance helps minimize costs. If this were a production system, it is recommended to provision three instances for high availability. You can leave the other settings in the **Configuration** section at their default.**

Amazon DocumentDB Developer Guide

Step 3: Create an Amazon DocumentDB cluster

The screenshot shows the 'Create Amazon DocumentDB cluster' configuration page. In the 'Configuration' section, the 'Cluster identifier' is set to 'ddocdb-2020-08-17-23-59-56'. The 'Instance class' is set to 'db.r5.large' (2 vCPUs, 16GiB RAM). The 'Number of instances' dropdown menu is open, with the value '1' selected. A red circle highlights the dropdown menu. Below the dropdown, a note states: 'The estimated hourly cost for 1 db.r5.large instance(s) is \$0.28/hr.'

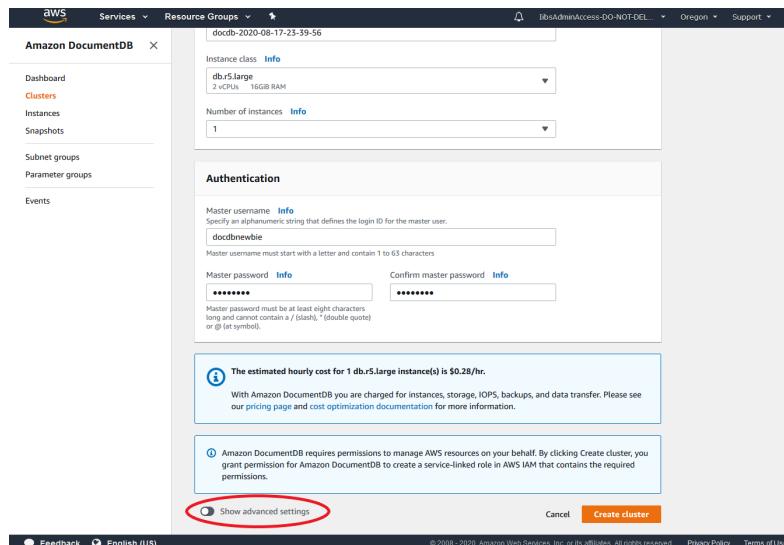
- 3. In the Authentication section, enter a username and password.**

The screenshot shows the 'Create Amazon DocumentDB cluster' configuration page. In the 'Configuration' section, the 'Cluster identifier' is 'ddocdb-2020-08-17-23-59-56', 'Instance class' is 'db.r5.large', and 'Number of instances' is '1'. In the 'Authentication' section, the 'Master username' is 'ddocdbnewbie' and the 'Master password' is '*****'. The 'Confirm master password' field also contains '*****'. A red circle highlights the 'Master username' field. Another red circle highlights the 'Master password' and 'Confirm master password' fields. Below the authentication section, a note states: 'The estimated hourly cost for 1 db.r5.large instance(s) is \$0.28/hr.'

- 4. Turn on Show advanced settings.**

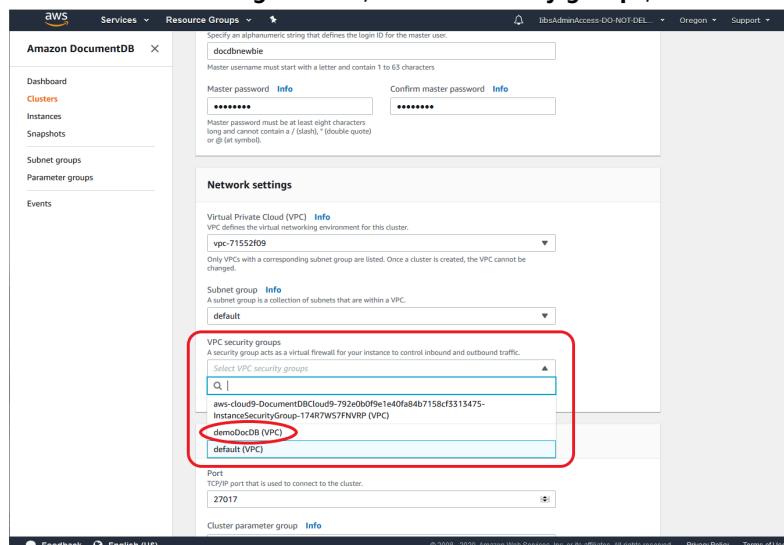
Amazon DocumentDB Developer Guide

Step 3: Create an Amazon DocumentDB cluster



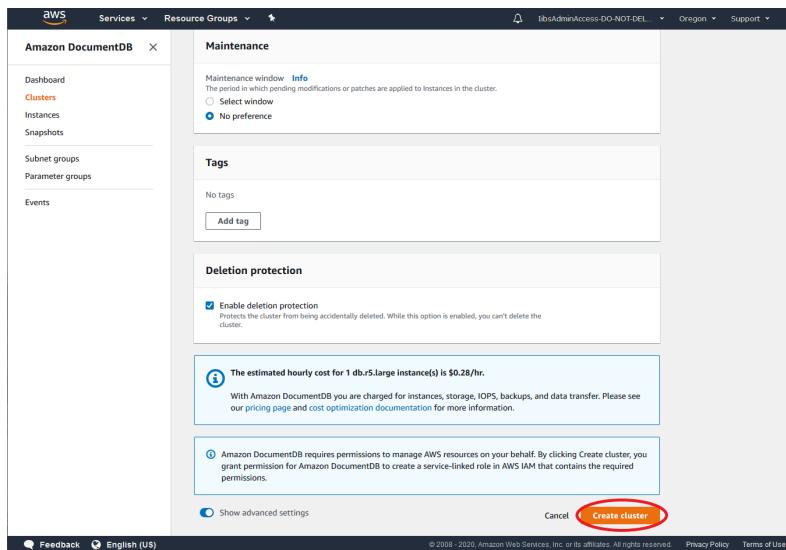
The screenshot shows the 'Create cluster' wizard for Amazon DocumentDB. In the 'Instance class' dropdown, 'db.r5.large' is selected. The 'Number of instances' dropdown is set to 1. Under 'Authentication', the 'Master username' is 'docdbnewbie'. Below it, there are fields for 'Master password' and 'Confirm master password', both containing '*****'. A note indicates the estimated hourly cost is \$0.28/hr. Another note states that permissions are required for the service-linked role in AWS IAM. At the bottom right are 'Cancel' and 'Create cluster' buttons, with 'Create cluster' being orange.

5. In the Network settings section, for VPC security groups, choose demoDocDB.



The screenshot shows the 'Network settings' section. Under 'Virtual Private Cloud (VPC)', the 'VPC' dropdown is set to 'vpc-71552609'. The 'Subnet group' dropdown is set to 'default'. In the 'VPC security groups' dropdown, 'demoDocDB (VPC)' is selected. Other options listed include 'aws-cloud9-DocumentDBCloud9-792e0b0f9e1e40fa84b7158cf3513475-InstanceSecurityGroup-174R7W57FNVNRP (VPC)' and 'default (VPC)'. The 'Port' field is set to '27017'. At the bottom right are 'Cancel' and 'Create cluster' buttons, with 'Create cluster' being orange.

6. Choose Create cluster.

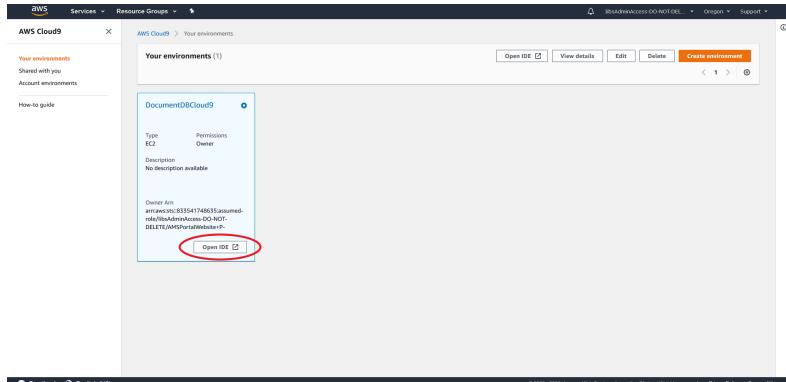


Amazon DocumentDB is now provisioning your cluster, which can take up to a few minutes to finish. You can connect to your cluster when both the cluster and instance status show as Available.

Step 4: Install the mongo shell

You will now install the mongo shell in your AWS Cloud9 environment that you created in Step 1. The mongo shell is a command-line utility that you use to connect and query your Amazon DocumentDB cluster.

1. If your AWS Cloud9 environment is still open from Step 1, go back to that environment and skip to instruction 3. If you navigated away from your AWS Cloud9 environment, in the AWS Cloud9 management console, under **Your environments**, find the environment labeled **DocumentDBCloud9**. Choose **Open IDE**.



2. At the command prompt, create the repository file with the following command:

```
echo -e "[mongodb-org-4.0] \nname=MongoDB Repository\nbaseurl=https://repo.mongodb.org/yum/amazon/2013.03/mongodb-org/4.0/x86_64/\npgpcheck=1 \nenabled=1 \npgpkey=https://www.mongodb.org/static/pgp/server-4.0.asc" | sudo tee /etc/yum.repos.d/mongodb-org-4.0.repo
```

3. When it is complete, install the mongo shell with the following command:

```
sudo yum install -y mongodb-org-shell
```

4. To encrypt data in transit, download the public key for Amazon DocumentDB from <https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem>. This operation downloads a file named rds-combined-ca-bundle.pem.

```
wget https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem
```

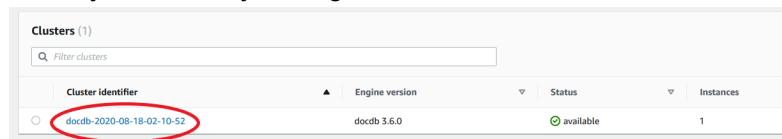
Note

Encryption-in-transit is enabled by default on Amazon DocumentDB. You can optionally disable TLS. For more information, see [Managing Amazon DocumentDB Cluster TLS Settings](#).

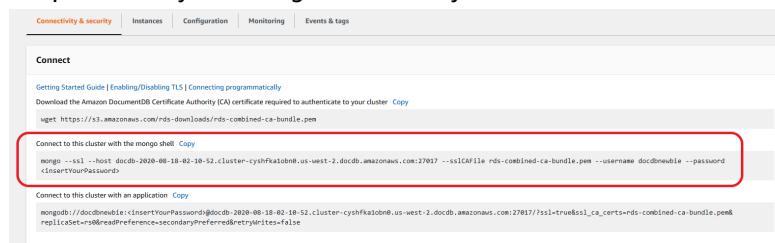
Step 5: Connect to your Amazon DocumentDB cluster

You will now connect to your Amazon DocumentDB cluster using the mongo shell that you installed in Step 4.

1. On the Amazon DocumentDB management console, under **Clusters**, locate your cluster. Choose the cluster you created by clicking on the cluster identifier.



2. In the **Connectivity and Security** tab, under **Connect to this cluster with the mongo shell**, copy the connection string provided. Omit copying <insertYourPassword> so that you are prompted for the password by the mongo shell when you connect.



3. Go back to your AWS Cloud9 environment and paste the connection string.

When you enter your password and your prompt becomes rs0:PRIMARY> prompt, you are successfully connected to your Amazon DocumentDB cluster.

Note

For information about troubleshooting, see [Troubleshooting Amazon DocumentDB](#).

Step 6: Insert and query data

Now that you are connected to your cluster, you can run a few queries to get familiar with using a document database.

1. To insert a single document, enter the following:

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

2. You get the following output:

```
WriteResult({ "nInserted" : 1 })
```

3. You can read the document that you wrote with the `findOne()` command (because it only returns a single document). Input the following:

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

4. You get the following output:

```
{ "_id" : ObjectId("5e401fe56056fda7321fdb67"), "hello" : "DocumentDB" }
```

5. To perform a few more queries, consider a gaming profiles use case. First, insert a few entries into a collection titled `profiles`. Input the following:

```
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}
])
```

6. You get the following output:

```
{ "acknowledged" : true, "insertedIds" : [ 1, 2, 3, 4 ] }
```

7. Use the `find()` command to return all the documents in the `profiles` collection. Input the following:

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

8. You will get an output that will match the data you typed in Step 5.

9. Use a query for a single document using a filter. Input the following:

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

10. You should get back this output:

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

11. Now let's try to find a profile and modify it using the `findAndModify` command. We'll give the user Matt an extra ten points with the following code:

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

12. You get the following output (note that his score hasn't increased yet):

```
{  
  "_id" : 1,  
  "name" : "Matt",  
  "status" : "active",  
  "level" : 12,  
  "score" : 202  
}
```

13. You can verify that his score has changed with the following query:

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

14. You get the following output:

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

Step 7: Explore

Congratulations! You have successfully completed the Get Started Guide to Amazon DocumentDB.

What's next? Learn how to fully leverage this database with some of its popular features:

- [Managing Amazon DocumentDB](#)
- [Scaling](#)
- [Backing up and restoring](#)

Note

The cluster you created from this get started exercise will continue to accrue cost unless you delete it. For directions, see [Deleting an Amazon DocumentDB Cluster](#).

Amazon DocumentDB Quick Start Using AWS CloudFormation

This section contains steps and other information to help you get started quickly with Amazon DocumentDB (with MongoDB compatibility) using [AWS CloudFormation](#). For general information about Amazon DocumentDB, see [What Is Amazon DocumentDB \(with MongoDB Compatibility\) \(p. 1\)](#).

These instructions use an AWS CloudFormation template to create a cluster and instances in your default Amazon VPC. For instructions on creating these resources yourself, see [Get Started with Amazon DocumentDB \(p. 31\)](#).

Important

The AWS CloudFormation stack that is created by this template creates multiple resources, including resources in Amazon DocumentDB (for example, a cluster and instances) and Amazon Elastic Compute Cloud (for example, a subnet group).

Some of these resources are not free-tier resources. For pricing information, see [Amazon DocumentDB Pricing](#) and [Amazon EC2 Pricing](#). You can delete the stack when you are finished with it to stop any charges.

This AWS CloudFormation stack is intended as for a tutorial purpose only. If you use this template for a production environment, we recommend that you use stricter IAM policies and security. For information about securing resources, see [Amazon VPC Security](#) and [Amazon EC2 Network and Security](#).

Topics

- [Prerequisites \(p. 44\)](#)
- [Launching an Amazon DocumentDB AWS CloudFormation Stack \(p. 46\)](#)
- [Accessing the Amazon DocumentDB Cluster \(p. 49\)](#)
- [Termination Protection and Deletion Protection \(p. 49\)](#)

Prerequisites

Before you create an Amazon DocumentDB (with MongoDB compatibility) cluster, you must have the following:

- A default Amazon VPC
- The required IAM permissions

Required IAM Permissions

The following permissions allow you to create resources for the AWS CloudFormation stack:

AWS Managed Policies

- [AWSCloudFormationReadOnlyAccess](#)
- [AmazonDocDBFullAccess](#)

Additional IAM Permissions

The following policy outlines the additional permissions that are required to create and delete this AWS CloudFormation stack.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetSSHPublicKey",
                "iam>ListSSHPublicKeys",
                "iam>CreateRole",
                "iam>CreatePolicy",
                "iam>PutRolePolicy",
                "iam>CreateInstanceProfile",
                "iam>AddRoleToInstanceProfile",
                "iam>GetAccountSummary",
                "iam>ListAccountAliases",
                "iam>GetRole",
                "iam>DeleteRole",
                "iam>RemoveRoleFromInstanceProfile",
                "iam>DeleteRolePolicy",
                "iam>DeleteInstanceProfile",
                "cloudformation:*Stack",
                "ec2:DescribeKeyPairs",
                "ec2:*Vpc",
                "ec2:DescribeInternetGateways",
                "ec2:*InternetGateway",
                "ec2:createTags",
                "ec2:*VpcAttribute",
                "ec2:DescribeRouteTables",
                "ec2:*RouteTable",
                "ec2:*Subnet",
                "ec2:*SecurityGroup",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:DescribeVpcEndpoints",
                "ec2:*VpcEndpoint",
                "ec2:*SubnetAttribute",
                "ec2:*Route",
                "ec2:*Instances",
                "ec2>DeleteVpcEndpoints"
            ],
            "Resource": "*"
        },
        {
            "Sid": "iamPassRole",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "rds.amazonaws.com"
                }
            }
        }
    ]
}
```

Note

The bolded permissions in the preceding policy are only required to delete a stack:
iam>DeleteRole, iam>RemoveRoleFromInstanceProfile, iam>DeleteRolePolicy,

`iam:DeleteInstanceProfile`, and `ec2:DeleteVpcEndpoints`. Also note that `ec2: *Vpc` grants `ec2:DeleteVpc` permissions.

Amazon EC2 Key Pair

You must have a key pair (and the PEM file) available in the Region where you will create the AWS CloudFormation stack. If you need to create a key pair, see [Creating a Key Pair Using Amazon EC2](#) in the *Amazon EC2 User Guide for Linux Instances*.

Launching an Amazon DocumentDB AWS CloudFormation Stack

This section describes how to launch and configure an Amazon DocumentDB AWS CloudFormation stack.

1. Sign in to the AWS Management Console at <https://console.aws.amazon.com/>.
2. The following table lists the Amazon DocumentDB stack templates for each AWS Region. Choose **Launch Stack** for the AWS Region you want to launch your stack in.

Region	View Template	View in Designer	Launch
US East (Ohio)	View Template	View in Designer	Launch Stack 
US East (N. Virginia)	View Template	View in Designer	Launch Stack 
US West (Oregon)	View Template	View in Designer	Launch Stack 
Asia Pacific (Mumbai)	View Template	View in Designer	Launch Stack 
Asia Pacific (Seoul)	View Template	View in Designer	Launch Stack 
Asia Pacific (Singapore)	View Template	View in Designer	Launch Stack 
Asia Pacific (Sydney)	View Template	View in Designer	Launch Stack 
Asia Pacific (Tokyo)	View Template	View in Designer	Launch Stack 
Canada (Central)	View Template	View in Designer	Launch Stack 
Europe (Frankfurt)	View Template	View in Designer	Launch Stack 
Europe (Ireland)	View Template	View in Designer	Launch Stack 
Europe (London)	View Template	View in Designer	Launch Stack 

Region	View Template	View in Designer	Launch
Europe (Paris)	View Template	View in Designer	Launch Stack 

3. **Create stack** — Describes the Amazon DocumentDB template that you selected. Every stack is based on a template — a JSON or YAML file — that contains configuration about the AWS resources you want to include in the stack. Because you chose to launch a stack from the provided templates above, your template has already been configured to create an Amazon DocumentDB stack for the AWS Region you chose.

When you launch an AWS CloudFormation stack, [deletion protection \(p. 49\)](#) for your Amazon DocumentDB cluster is disabled by default. If you want to enable deletion protection for your cluster, complete the following steps. Otherwise, choose **Next** to continue to the next step.

To enable deletion protection for your Amazon DocumentDB cluster:

1. Choose **View in Designer** from the bottom right corner of the **Create stack** page.
2. Modify the template using the integrated JSON and YAML editor in the resulting AWS CloudFormation Designer page of the console. Scroll to the Resources section and modify it to include `DeletionProtection`, as follows. For more information about using AWS CloudFormation Designer, see [What Is AWS CloudFormation Designer?](#).

JSON:

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

YAML:

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

3. Choose **Create Stack** () from the top left corner of the page to save your changes and create a stack with these changes enabled.
4. After you save your changes, you will be redirected to the **Create stack** page.
5. Choose **Next** to continue.

4. **Specify stack details** — Enter the stack name and parameters for your template. Parameters are defined in your template and allow you to input custom values when you create or update a stack.
 - Under **Stack name**, enter a name for your stack or accept the provided name. The stack name can include letters (A—Z and a—z), numbers (0—9), and dashes (—).
 - Under **Parameters**, enter the following details:
 - **DBClusterName** — Enter a name for your Amazon DocumentDB cluster or accept the provided name.

Cluster naming constraints:

 - Length is [1—63] letters, numbers, or hyphens.
 - First character must be a letter.
 - Cannot end with a hyphen or contain two consecutive hyphens.
 - Must be unique for all clusters across Amazon RDS, Neptune, and Amazon DocumentDB per AWS account, per Region.
 - **DBInstanceClass** — From the drop-down list, select the instance class for your Amazon DocumentDB cluster.
 - **DBInstanceName** — Enter a name for your Amazon DocumentDB instance or accept the provided name.
 - Instance naming constraints:
 - Length is [1—63] letters, numbers, or hyphens.
 - First character must be a letter.
 - Cannot end with a hyphen or contain two consecutive hyphens.
 - Must be unique for all instances across Amazon RDS, Neptune, and Amazon DocumentDB per AWS account, per Region.
 - **MasterPassword** — The database admin account password.
 - **MasterUser** — The database admin account username. The MasterUser must begin with a letter and can contain only alphanumeric characters.

Choose **Next** to save your changes and continue.

5. **Configure stack options** — Configure your stack's tags, permissions, and additional options.
 - **Tags** — Specify tags (key-value) pairs to apply to your resources in your stack. You can add up to 50 unique tags for each stack.
 - **Permissions** — Optional. Choose an IAM role to explicitly define how AWS CloudFormation can create, modify, or delete resources in the stack. If you don't choose a role, AWS CloudFormation uses permissions based on your user credentials. Before you specify a service role, ensure that you have permission to pass it (`iam:PassRole`). The `iam:PassRole` permission specifies which roles you can use.

Note

When you specify a service role, AWS CloudFormation always uses that role for all operations that are performed on that stack. Other users that have permissions to perform operations on this stack will be able to use this role, even if they don't have permission to pass it. If the role includes permissions that the user shouldn't have, you can unintentionally escalate a user's permissions. Ensure that the role grants [least privilege](#).

- **Advanced options** — You can set the following advanced options:
 - **Stack policy** — Optional. Defines the resources that you want to protect from unintentional updates during a stack update. By default, all resources can be updated during a stack update.

You can enter the stack policy directly as JSON, or upload a JSON file containing the stack policy. For more information, see [Prevent Updates to Stack Resources](#).

- **Rollback configuration** — Optional. Specify CloudWatch Logs alarms for AWS CloudFormation to monitor when creating and updating the stack. If the operation breaches an alarm threshold, AWS CloudFormation rolls it back.
- **Notification options** — Optional. Specify topics for Simple Notification System (SNS).
- **Stack creation options** — Optional. You can specify the following options:
 - **Rollback on failure** — Whether or not the stack should be rolled back if the stack creation fails.
 - **Timeout** — The number of minutes before a stack creation times out.
 - **Termination protection** — Prevents the stack from being accidentally deleted.

Note

AWS CloudFormation termination protection is different from the Amazon DocumentDB concept of deletion protection. For more information, see [Termination Protection and Deletion Protection \(p. 49\)](#).

Choose **Next** to continue.

6. **Review <stack-name>** — Review your stack template, details, and configuration options. You can also open a **quick-create link** at the bottom of the page to create stacks with the same basic configurations as this one.
 - Choose **Create** to create the stack.
 - Alternatively, you can choose **Create change set**. A change set is a preview of how this stack will be configured before creating the stack. This allows you to examine various configurations before executing the change set.

Accessing the Amazon DocumentDB Cluster

Once the AWS CloudFormation stack has been completed, you can use an Amazon EC2 instance to connect to your Amazon DocumentDB cluster. For information about connecting to an Amazon EC2 instance using SSH, see [Connect to Your Linux Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

After you are connected, see the following sections, which contain information about using Amazon DocumentDB.

- [Step 4: Install the mongo shell \(p. 40\)](#)
- [Deleting an Amazon DocumentDB Cluster \(p. 296\)](#)

Termination Protection and Deletion Protection

It is an Amazon DocumentDB best practice to enable deletion protection and termination protection. CloudFormation termination protection is a distinctly different feature from the Amazon DocumentDB deletion protection feature.

- **Termination protection** — You can prevent a stack from being accidentally deleted by enabling termination protection for your CloudFormation stack. If a user attempts to delete a stack with termination protection enabled on it, the deletion fails and the stack remains unchanged. Termination protection is disabled by default when you create a stack using CloudFormation. You can enable termination protection on a stack when you create it. For more information, see [Setting AWS CloudFormation Stack Options](#).
- **Deletion protection** — Amazon DocumentDB also provides the ability to enable deletion protection for a cluster. If a user attempts to delete an Amazon DocumentDB cluster with deletion protection

enabled on it, the deletion fails and the cluster remains unchanged. Deletion protection, when enabled, safeguards against accidental deletes from the Amazon DocumentDB AWS Management Console, AWS CLI, and CloudFormation. For more information on enabling and disabling deletion protection for an Amazon DocumentDB cluster, see [Deletion Protection \(p. 296\)](#).

MongoDB 4.0 Compatibility

Amazon DocumentDB supports MongoDB 4.0 compatibility including ACID transactions. MongoDB 4.0 compatibility means that a vast majority of the applications, drivers, and tools you already use today with your MongoDB 4.0 databases can be used with Amazon DocumentDB 4.0 with little or no change. This section describes everything you need to know about Amazon DocumentDB 4.0 including new capabilities and features, getting started, migrations paths, and functional differences.

Topics

- [What's new in Amazon DocumentDB 4.0 \(p. 51\)](#)
- [Get Started with Amazon DocumentDB 4.0 \(p. 52\)](#)
- [Upgrade or Migrate to Amazon DocumentDB 4.0 \(p. 52\)](#)
- [Functional Differences \(p. 52\)](#)

What's new in Amazon DocumentDB 4.0

Amazon DocumentDB 4.0 introduces many new features and capabilities that include ACID transactions and improvements to change streams. The summary below introduces some of major features that were introduced in Amazon DocumentDB 4.0. To see a full list of the new capabilities, see the [Release Notes \(p. 759\)](#).

- **ACID Transactions:** Amazon DocumentDB now supports the ability to perform transactions across multiple documents, statements, collections, and databases. Transactions simplify application development by enabling you to perform atomic, consistent, isolated, and durable (ACID) operations across one or more documents within an Amazon DocumentDB cluster. For more information, see [Transactions \(p. 54\)](#).
- **Change streams:** You now have the ability to open a change stream at the cluster level (`client.watch()` or `mongo.watch()`) and the database (`db.watch()`), you can specify a `startAtOperationTime` to open a change stream cursor, and lastly you can now extend your change stream retention period to 7 days (previously 24 hours). For more information, see [Using Change Streams with Amazon DocumentDB \(p. 486\)](#).
- **AWS Database Migration Service (AWS DMS):** You can now use AWS DMS to migrate your MongoDB 4.0 workloads to Amazon DocumentDB. AWS DMS now supports a MongoDB 4.0 source, Amazon DocumentDB 4.0 target, and an Amazon DocumentDB 3.6 source for performing upgrades between Amazon DocumentDB 3.6 and 4.0. For more information, see [AWS DMS Documentation](#).
- **Performance and indexing:** You can now utilize an index with `$lookup`, find queries with a projection that contain one field or one field and the `_id` field can be served direct from the index and without needing to read from the collection (covered query), the ability to `hint()` with `findAndModify`, performance optimizations for `$addToSet`, and improvements to reduce overall index sizes. For more information, see [Release Notes \(p. 759\)](#).
- **Operators:** Amazon DocumentDB 4.0 now supports a number of new aggregation operators: `$ifNull`, `$replaceRoot`, `$setIsSubset`, `$setIntersection`, `$setUnion`, `$setEquals`. You can see all the MongoDB APIs, Operations, and Data Types that we support at [Supported MongoDB APIs, Operations, and Data Types \(p. 102\)](#).
- **Role based access control (RBAC):** With both `ListCollection` and `ListDatabase` commands you can now optionally use the `authorizedCollections` and `authorizedDatabases` parameters to allow users to list the collections and databases that they have permission to access without requiring

the `listCollections` and `listDatabase` roles, respectively. You also have the ability to kill your own cursors without requiring the `KillCursor` role.

Amazon DocumentDB does not support every MongoDB 4.0 feature. When we built Amazon DocumentDB 4.0, we worked backwards from the feature and capabilities that our customers asked us to build the most. We will continue to add additional MongoDB 4.0 capabilities based on what customers ask us to build. For example, Amazon DocumentDB 4.0 does not currently support the type conversion operators or the string operators that were introduced in MongoDB 4.0. For the latest list of supported APIs, please see [Supported MongoDB APIs, Operations, and Data Types \(p. 102\)](#).

Get Started with Amazon DocumentDB 4.0

To get started with Amazon DocumentDB 4.0, please see the [Get Started Guide](#). You can create a new Amazon DocumentDB 4.0 cluster using the AWS Management Console or the AWS SDK, AWS CLI, or AWS CloudFormation. When connecting to Amazon DocumentDB, it is required that you use a MongoDB driver or utility that is compatible with MongoDB 4.0 or higher.

Note

When using the AWS SDK, AWS CLI, or AWS CloudFormation, the engine version will default to 3.6. You must explicitly specify the parameter `engineVersion=4.0` to create a new Amazon DocumentDB 4.0.0 cluster. For a given Amazon DocumentDB cluster, you can determine the cluster version using the AWS CLI to call `describe-db-clusters` or use the Amazon DocumentDB management console to view the engine version number for a particular cluster.

Amazon DocumentDB 4.0 supports `r5` and `t3.medium` instance types for your clusters and is available in all supported regions. There is no additional cost for using Amazon DocumentDB 4.0. For more information on pricing, see [Amazon DocumentDB \(with MongoDB compatibility\) Pricing](#).

Upgrade or Migrate to Amazon DocumentDB 4.0

You can migrate from MongoDB 3.6 or MongoDB 4.0 to Amazon DocumentDB 4.0 utilizing the [AWS DMS](#) or utilities like `mongodump`, `mongorestore`, `mongoimport`, and `mongoexport`. Similarly, you can use the same tools to upgrade from Amazon DocumentDB 3.6 to Amazon DocumentDB 4.0. For instructions on how to migrate, see [Upgrading your Amazon DocumentDB cluster from 3.6 to 4.0 using AWS Database Migration Service \(p. 121\)](#).

Functional Differences

Functional Differences Between Amazon DocumentDB 3.6 and 4.0

With the release of Amazon DocumentDB 4.0, there are functional differences between Amazon DocumentDB 3.6 and Amazon DocumentDB 4.0:

- **Projection for nested documents:** Amazon DocumentDB 3.6 considers the first field in a nested document when applying a projection. However, Amazon DocumentDB 4.0 will parse subdocuments and apply the projection to each sub document as well. For example: if the projection is `"a.b.c": 1`, then the behavior in both versions is identical. However, if the projection is `{a:{b:{c:1}}}` then Amazon DocumentDB 3.6 will only apply the projection to 'a' and not 'b' or 'c'.

- **Behavior for `minKey`, `maxKey`:** In Amazon DocumentDB 4.0, the behavior for `{x:{$gt:MaxKey}}` returns nothing, and for `{x:{$lt:MaxKey}}` returns everything.
- **Document comparison differences:** Comparing numerical values of different types (double, int, long) in subdocuments (e.g., b in `{"_id" :1, "a" :{"b":1}}`) now provides a consistent output across numerical data types and for each level of a document.

Functional Differences Between Amazon DocumentDB 4.0 and MongoDB 4.0

Below are functional differences between Amazon DocumentDB 4.0 and MongoDB 4.0.

- **Lookup with empty key in path:** When a collection contains a document with empty key inside the array (e.g. `{"x" : [{ "" : 10 }, { "b" : 20 }]}`), and when the key used in the query ends in an empty string (e.g. `x .`), then Amazon DocumentDB will return that document since it traverses all the documents in the array whereas MongoDB will not return that document.
- **\$setOnInsert along with \$ in the path:** The field operator `$setOnInsert` will not work in combination with `$` in the path in Amazon DocumentDB, which is also consistent with MongoDB 4.0.

Transactions

Amazon DocumentDB (with MongoDB compatibility) now supports MongoDB 4.0 compatibility including transactions. You can perform transactions across multiple documents, statements, collections, and databases. Transactions simplify application development by enabling you to perform atomic, consistent, isolated, and durable (ACID) operations across one or more documents within an Amazon DocumentDB cluster. Common use cases for transactions include financial processing, fulfilling and managing orders, and building multi-player games.

There is no additional cost for transactions. You only pay for the read and write IOs that you consume as part of the transactions.

Topics

- [Requirements \(p. 54\)](#)
- [Best Practices \(p. 54\)](#)
- [Limitations \(p. 54\)](#)
- [Monitoring and Diagnostics \(p. 55\)](#)
- [Transaction Isolation Level \(p. 56\)](#)
- [Use Cases \(p. 56\)](#)
- [Supported Commands \(p. 81\)](#)
- [Unsupported Capabilities \(p. 81\)](#)
- [Sessions \(p. 82\)](#)
- [Transaction Errors \(p. 83\)](#)

Requirements

To use the transactions feature, you need to meet the following requirements:

- You must be using the Amazon DocumentDB 4.0 engine.
- You must use a driver compatible with MongoDB 4.0 or greater.

Best Practices

Here are some best practices so that you can get the most using transactions with Amazon DocumentDB.

- Always commit or abort the transaction after it is complete. Leaving a transaction in an incomplete state ties up database resources and can cause write conflicts.
- It is recommended to keep transactions to the smallest number of commands needed. If you have transactions with multiple statements that can be divided up into multiple smaller transactions, it is advisable to do so to reduce the likelihood of a timeout. Always aim to create short transactions, not long-running reads.

Limitations

- Amazon DocumentDB does not support cursors within a transaction.

- Amazon DocumentDB cannot create new collections in a transaction and cannot query/update against non-existing collections.
- Document-level write locks are subject to a 1 minute timeout, which is not configurable by the user.
- Retryable writes, retryable commit, and retryable abort commands are not supported in Amazon DocumentDB. Exception: If you are using mongo shell, do not include the `retryWrites=false` command in any code string. By default, retryable writes are disabled. Including `retryWrites=false` might cause failure in normal read commands.
- Each Amazon DocumentDB instance has an upper bound limit on the number of concurrent transaction open on the instance at one time. For the limits, please see [Instance Limits \(p. 539\)](#).
- For a given transaction, the transaction log size must be less than 32MB.
- Amazon DocumentDB does support `count()` within a transactions, but not all drivers support this capability. An alternative is to use the `countDocuments()` API, which translates the `count` query into an aggregation query on the client side.
- Transactions have a one minute execution limit and sessions have a 30-minute timeout. If a transaction times out, it will be aborted, and any subsequent commands issued within the session for the existing transaction will yield the following error:

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

Monitoring and Diagnostics

With the support for transactions in Amazon DocumentDB 4.0, additional CloudWatch metrics were added to help you monitor your transactions.

New CloudWatch Metrics

- `DatabaseTransactions`: The number of open transactions taken at a one-minute period.
- `DatabaseTransactionsAborted`: The number of aborted transactions taken at a one-minute period.
- `DatabaseTransactionsMax`: The maximum number of open transactions in a one-minute period.
- `TransactionsAborted`: The number of transactions aborted on an instance in a one-minute period.
- `TransactionsCommitted`: The number of transactions committed on an instance in a one-minute period.
- `TransactionsOpen`: The number of transactions open on an instance taken at a one-minute period.
- `TransactionsOpenMax`: The maximum number of transactions open on an instance in a one-minute period.
- `TransactionsStarted`: The number of transactions started on an instance in a one-minute period.

Note

For more CloudWatch metrics for Amazon DocumentDB, go to [Monitoring Amazon DocumentDB with CloudWatch \(p. 414\)](#).

Additionally, new fields were added to both `currentOp.lsid`, `transactionThreadId`, and a new state for "idle transaction" and `serverStatus.transactions`: `currentActive`, `currentInactive`, `currentOpen`, `totalAborted`, `totalCommitted`, and `totalStarted`.

Transaction Isolation Level

When starting a transaction, you have the ability to specify both the `readConcern` and `writeConcern` as shown in the example below:

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

For `readConcern`, Amazon DocumentDB supports snapshot isolation by default. If a `readConcern` of local, available, or majority are specified, Amazon DocumentDB will upgrade the `readConcern` level to snapshot. Amazon DocumentDB does not support the linearizable `readConcern` and specifying such a read concern will result in an error.

For `writeConcern`, Amazon DocumentDB supports majority by default and a write quorum is achieved when four copies of the data are persisted across three AZs. If a lower `writeConcern` is specified, Amazon DocumentDB will upgrade the `writeConcern` to majority. Further, all Amazon DocumentDB writes are journaled and journaling cannot be disabled.

Use Cases

In this section, we will walk through two use cases for transactions: multi-statement and multi-collection.

Multi-Statement Transactions

Amazon DocumentDB transactions are multi-statement, which means you can write a transaction that spans multiple statements with an explicit commit or rollback. You can group `insert`, `update`, `delete`, and `findAndModify` actions as a single atomic operation.

A common use case for multi-statement transactions is a debit-credit transaction. For example: you owe a friend money for clothes. Thus, you need to debit (withdraw) \$500 from your account and credit \$500 (deposit) to your friend's account. To perform that operation, you perform both the debt and credit operations within a single transaction to ensure atomicity. Doing so prevents scenarios where \$500 is debited from your account, but not credited to your friend's account. Here's what this use case would look like:

```
// *** 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 Transactions

Our transactions are also multi-collection, which means they can be used to perform multiple operations within a single transaction and across multiple collections. This provides a consistent view of data and maintains your data's integrity. When you commit the commands as a single <>, the transactions are all-or-nothing executions—in that, they will either all succeed or all fail.

Here is an example of multi-collection transactions, using the same scenario and data from the example for multi-statement transactions.

```

// *** 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

```

Transaction API Examples for Callback API

The callback API is only available for 4.2+ drivers.

Javascript

The following code demonstrates how to utilize the Amazon DocumentDB transaction API with 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

The following code demonstrates how to utilize the Amazon DocumentDB transaction API with 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#

The following code demonstrates how to utilize the Amazon DocumentDB transaction API with 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

The following code demonstrates how to utilize the Amazon DocumentDB transaction API with 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

The following code demonstrates how to utilize the Amazon DocumentDB transaction API with 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)
// 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

The following code demonstrates how to utilize the Amazon DocumentDB transaction API with 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

The following code demonstrates how to utilize the Amazon DocumentDB transaction API with 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

The following code demonstrates how to utilize the Amazon DocumentDB transaction API with 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
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

```

Transaction API Examples for Core API

Javascript

The following code demonstrates how to utilize the Amazon DocumentDB transaction API with 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#

The following code demonstrates how to utilize the Amazon DocumentDB transaction API with 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.ToInt32() - 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.ToInt32() + 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

The following code demonstrates how to utilize the Amazon DocumentDB transaction API with 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
```

Java

The following code demonstrates how to utilize the Amazon DocumentDB transaction API with 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);

    MongoDB 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.beginTransaction(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(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
observervation", e);
    }
    if (!errors.isEmpty()) {
        throw errors.get(0);
    }
    return this;
}

public boolean getCompleted() {
    return this.completed;
}

public void close() {
    subscription.cancel();
    received.clear();
}
}

```

C

The following code demonstrates how to utilize the Amazon DocumentDB transaction API with 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

The following code demonstrates how to utilize the Amazon DocumentDB transaction API with 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()

}

```

Supported Commands

Command	Supported
abortTransaction	Yes
commitTransaction	Yes
endSessions	Yes
killSession	Yes
killAllSession	Yes
killAllSessionsByPattern	No
refreshSessions	No
startSession	Yes

Unsupported Capabilities

Methods	Stages or Commands
db.collection.aggregate()	\$collStats \$currentOp \$indexStats \$listSessions \$out
db.collection.count() db.collection.countDocuments()	\$where \$near \$nearSphere
db.collection.insert()	insert is not supported if it is not run against an existing collection. This method is supported if it targets a pre-existing collection.

Sessions

MongoDB sessions are a framework that is used to support retryable writes, causal consistency, transactions, and manage operations across databases. When a session is created, a logical session identifier (`lsid`) is generated by the client and is used to tag all operations within that session when sending commands to the server.

Amazon DocumentDB supports the use of sessions to enable transactions, but does not support causal consistency or retryable writes.

When utilizing transactions within Amazon DocumentDB, a transaction will be initiated from within a session using the `session.startTransaction()` API and a session supports a single transaction at a time. Similarly, transactions are completed using either the `commit` (`session.commitTransaction()`) or `abort` (`session.abortTransaction()`) APIs.

Causal consistency

Causal consistency guarantees that within a single client session the client will observe read-after-write consistency, monatomic reads/writes, and writes will follow reads and these guarantees apply across all instances in a cluster, not just the primary. Amazon DocumentDB does not support causal consistency and the following statement will result in an error.

```
var mySession = db.getMongo().startSession();
var mySessionObject = mySession.getDatabase('test').getCollection('account');

mySessionObject.updateOne({"_id": 2}, {"$inc": {"balance": 400}});
//Result:{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

mySessionObject.find()
//Error: error: {
//    "ok" : 0,
//    "code" : 303,
//    "errmsg" : "Feature not supported: 'causal consistency'",
//    "operationTime" : Timestamp(1603461817, 493214)
//}

mySession.endSession()
```

You can disable causal consistency within a session. Please note, doing so will enable you to utilize the session framework, but will not provide causal consistency guarantees for reads. When using Amazon DocumentDB, reads from the primary will be read-after-write consistent and reads from the replica instances will be eventually consistent. Transactions are the primary use case for utilizing sessions.

```
var mySession = db.getMongo().startSession({causalConsistency: false});
var mySessionObject = mySession.getDatabase('test').getCollection('account');

mySessionObject.updateOne({"_id": 2}, {"$inc": {"balance": 400}});
//Result:{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

mySessionObject.find()
//[{"_id" : 1, "name" : "Bob", "balance" : 100 },
//{"_id" : 2, "name" : "Alice", "balance" : 1700 }]
```

Retryable writes

Retryable writes is a capability in which the client will attempt to retry write operations, one time, when network errors occur or if the client is unable to find the primary. In Amazon DocumentDB, retryable writes are not supported and must be disabled. You can disable it with the command (`retryWrites=false`) in the connection string.

Exception: If you are using mongo shell, do not include the `retryWrites=false` command in any code string. By default, retryable writes are disabled. Including `retryWrites=false` might cause failure in normal read commands.

Transaction Errors

When using transactions, there are scenarios that can yield an error that states that a transaction number does not match any in progress transaction.

The error can be generated in at least two different scenarios:

- After the one-minute transaction timeout.
- After an instance restart (due to patching, crash recovery, etc.), it is possible to receive this error even in cases where the transaction successfully committed. During an instance restart, the database can't tell the difference between a transaction that successfully completed versus a transaction that aborted. In other words, the transaction completion state is ambiguous.

The best way to handle this error is to make transactional updates idempotent -- for example, by using the `$set` mutator instead of an increment/decrement operation. See below:

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

Best Practices for Amazon DocumentDB

Learn best practices for working with Amazon DocumentDB (with MongoDB compatibility). This section is continually updated as new best practices are identified.

Topics

- [Basic Operational Guidelines \(p. 84\)](#)
- [Instance Sizing \(p. 85\)](#)
- [Working with Indexes \(p. 86\)](#)
- [Security Best Practices \(p. 87\)](#)
- [Cost Optimization \(p. 87\)](#)
- [Using Metrics to Identify Performance Issues \(p. 88\)](#)
- [TTL and Timeseries Workloads \(p. 90\)](#)
- [Migrations \(p. 90\)](#)
- [Working with Cluster Parameter Groups \(p. 90\)](#)
- [Aggregation Pipeline Queries \(p. 90\)](#)
- [batchInsert and batchUpdate \(p. 91\)](#)

Basic Operational Guidelines

The following are basic operational guidelines that everyone should follow when working with Amazon DocumentDB. The Amazon DocumentDB Service Level Agreement requires that you follow these guidelines.

- Deploy a cluster consisting of two or more Amazon DocumentDB instances in two AWS Availability Zones. For production workloads, we recommend deploying a cluster consisting of three or more Amazon DocumentDB instances in three Availability Zones.
- Use the service within the stated service limits. For more information, see [Amazon DocumentDB Quotas and Limits \(p. 535\)](#).
- Monitor your memory, CPU, connections, and storage usage. To help you maintain system performance and availability, set up Amazon CloudWatch to notify you when usage patterns change or when you approach the capacity of your deployment.
- Scale up your instances when you are approaching capacity limits. Your instances should be provisioned with enough compute resources (i.e., RAM, CPU) to accommodate unforeseen increases in demand from your applications.
- Set your backup retention period to align with your recovery point objective.
- Test failover for your cluster to understand how long the process takes for your use case. For more information, see [Amazon DocumentDB Failover \(p. 345\)](#).
- Connect to your Amazon DocumentDB cluster with the cluster endpoint (see [Amazon DocumentDB Endpoints \(p. 8\)](#)) and in replica set mode (see [Connecting to Amazon DocumentDB as a Replica Set \(p. 496\)](#)) to minimize the impact of a failover on your application.
- Choose a driver read preference setting that maximizes read scaling while meeting your application's read consistency requirements. The secondaryPreferred read preference enables replica reads and frees up the primary instance to do more work. For more information, see [Read Preference Options \(p. 12\)](#).

- Design your application to be resilient in the event of network and database errors. Use your driver's error mechanism to distinguish between transient errors and persistent errors. Retry transient errors using an exponential backoff mechanism when appropriate. Ensure that your application considers data consistency when implementing retry logic.
- Enable cluster deletion protection for all production clusters, or any cluster that has valuable data. Before deleting an Amazon DocumentDB cluster, take a final snapshot. If you are deploying resources with AWS CloudFormation, enable termination protection. For more information, see [Termination Protection and Deletion Protection \(p. 49\)](#).
- When creating an Amazon DocumentDB cluster, the --engine-version is an optional parameter that defaults to the latest major engine version. The current major engine version is 4.0.0. When new major engine versions are released, the default engine version for --engine-version will be updated to reflect the lastest major engine version. As a result, for production workloads, and especially those that are dependent on scripting, automation, or AWS CloudFormation templates, we recommend that you explicitly specify the --engine-version to the intended major version.

Instance Sizing

One of the most critical aspects of choosing an instance size in Amazon DocumentDB is the amount of RAM for your cache. Amazon DocumentDB reserves one-third of the RAM for its own services, meaning that only two-thirds of the instance RAM is available for the cache. Thus, it is an Amazon DocumentDB best practice to choose an instance type with enough RAM to fit your working set (i.e., data and indexes) in memory. Having properly sized instances will help optimize for overall performance and potentially minimize I/O cost. You can use the third party [Amazon DocumentDB sizing calculator](#) to estimate the instance size for a particular workload.

To determine whether your application's working set fits in memory, monitor the `BufferCacheHitRatio` using Amazon CloudWatch for each instance in a cluster that is under load.

The `BufferCacheHitRatio` CloudWatch metric measures the percentage of data and indexes served from an instance's memory cache (versus the storage volume). Generally speaking, the value of `BufferCacheHitRatio` should be as high as possible, as reading data from working set memory is faster and more cost-effective than reading from the storage volume. While it is desirable to keep `BufferCacheHitRatio` as close to 100% as possible, the best achievable value will depend on your application's access patterns and performance requirements. To maintain the highest possible `BufferCacheHitRatio`, it is recommended that the instances in your cluster are provisioned with enough RAM to be able to fit your indexes and working data set in memory.

If your indexes do not fit into memory, you will see a lower `BufferCacheHitRatio`. Continually reading from disk incurs additional I/O costs and is not as performant as reading from memory. If your `BufferCacheHitRatio` ratio is lower than expected, scale up the instance size for your cluster to provide more RAM to fit working set data in memory. If scaling up the instance class results in a dramatic increase in `BufferCacheHitRatio`, then your application's working set did not fit in memory. Continue to scale up until `BufferCacheHitRatio` no longer increases dramatically after a scaling operation. For information about monitoring an instance's metrics, see [Amazon DocumentDB Metrics \(p. 414\)](#).

Depending on your workload and latency requirements, it may be acceptable for your application to have higher `BufferCacheHitRatio` values during steady state usage, but have the `BufferCacheHitRatio` dip periodically as analytic queries that need to scan an entire collection are run on an instance. These periodic dips in `BufferCacheHitRatio` may manifest as higher latency for subsequent queries that need to repopulate the working set data from the storage volume back into the buffer cache. **We recommend that you test your workloads in a pre-production environment with a representative production workload first in order to understand the performance characteristics and `BufferCacheHitRatio` before deploying the workload to production.**

The `BufferCacheHitRatio` is an instance-specific metric, so different instances within the same cluster may have different `BufferCacheHitRatio` values depending on how reads are distributed

among the primary and replica instances. If your operational workload cannot handle periodic increases in latency from repopulating the working set cache after running analytic queries, you should try to isolate the regular workload's buffer cache from that of the analytic queries. You can achieve complete BufferCacheHitRatio isolation by directing operational queries to the primary instance and analytic queries only to the replica instances. You can also achieve partial isolation by directing analytic queries to a specific replica instance with the understanding that some percentage of regular queries will also run on that replica and could potentially be affected.

Appropriate BufferCacheHitRatio values depend on your use case and application requirements. There is no one best or minimum value for this metric; only you can decide if the tradeoff from a temporarily lower BufferCacheHitRatio is acceptable from a cost and performance perspective.

Working with Indexes

Building Indexes

When importing data into Amazon DocumentDB, you should create your indexes before importing large datasets. You can use the [Amazon DocumentDB Index Tool](#) to extract indexes from a running MongoDB instance or mongodump directory, and create those indexes in an Amazon DocumentDB cluster. For more guidance on migrations, see [Migrating to Amazon DocumentDB \(p. 121\)](#).

Index Selectivity

We recommend that you limit the creation of indexes to fields where the number of duplicate values is less than 1% of the total number of documents in the collection. As an example, if your collection contains 100,000 documents, only create indexes on fields where the same value occurs 1000 times or fewer.

Choosing an index with a high number of unique values (i.e., a high cardinality) ensures that filter operations return a small number of documents, thereby yielding good performance during index scans. An example of a high-cardinality index is a unique index, which guarantees that equality predicates return at most a single document. Examples of low-cardinality include an index over a Boolean field and an index over day of the week. Due to their poor performance, low cardinality indexes are unlikely to be chosen by the database's query optimizer. At the same time, low cardinality indexes continue to consume resources such as disk space and I/Os. As a rule of thumb, you should target indexes on fields where the typical value frequency is 1% of the total collection size or less.

Additionally, it is recommended to only create indexes on fields that are commonly utilized as a filter and regularly look for unused indexes. For more information, see [How Do I Identify Unused Indexes? \(p. 567\)](#).

Impact of Indexes on Writing Data

While indexes can improve query performance by avoiding the need to scan every document in a collection, this improvement comes with a tradeoff. For each index on a collection, every time a document is inserted, updated, or deleted, the database must update the collection and write the fields to each of the indexes for the collection. For example, if a collection has nine indexes, the database must perform ten writes before acknowledging the operation to the client. Thus, each additional index incurs additional write latency, I/O's, and increase in the overall utilized storage.

Cluster instances need to be appropriately sized to keep all working set memory. This avoids the need to continuously read index pages from the storage volume, which negatively impacts performance and generates higher I/O costs. For more information, see [Instance Sizing \(p. 85\)](#).

For best performance, minimize the number of indexes in your collections, adding only those indexes necessary to improve performance for common queries. While workloads vary, a good guideline is to keep the number of indexes per collection to five or fewer.

Identifying Missing Indexes

Identifying missing indexes is a best practice that we recommend performing on a regular basis. For more information, please see [How Do I Identify Missing Indexes? \(p. 567\)](#).

Identifying Unused Indexes

Identifying and removing unused indexes is a best practice that we recommend performing on a regular basis. For more information, please see [How Do I Identify Unused Indexes? \(p. 567\)](#).

Security Best Practices

For security best practices, you must use AWS Identity and Access Management (IAM) accounts to control access to Amazon DocumentDB API operations, especially operations that create, modify, or delete Amazon DocumentDB resources. Such resources include clusters, security groups, and parameter groups. You must also use IAM to control actions that perform common administrative actions such as backing up restoring clusters. When creating IAM roles, employ the principle of least privilege.

- Enforce least privilege with [role-based access control \(p. 169\)](#).
- Assign an individual IAM account to each person who manages Amazon DocumentDB resources. Do not use the AWS account root user to manage Amazon DocumentDB resources. Create an IAM user for everyone, including yourself.
- Grant each IAM user the minimum set of permissions that are required to perform their duties.
- Use IAM groups to effectively manage permissions for multiple users. For more information about IAM, see the [IAM User Guide](#). For information about IAM best practices, see [IAM Best Practices](#).
- Regularly rotate your IAM credentials.
- Configure AWS Secrets Manager to automatically rotate the secrets for Amazon DocumentDB. For more information, see [Rotating Your AWS Secrets Manager Secrets](#) and [Rotating Secrets for Amazon DocumentDB](#) in the [AWS Secrets Manager User Guide](#).
- Grant each Amazon DocumentDB user the minimum set of permissions that are required to perform their duties. For more information, see [Restricting Database Access Using Role-Based Access Control \(p. 169\)](#).
- Use Transport Layer Security (TLS) to encrypt your data in transit and AWS KMS to encrypt your data at rest.

Cost Optimization

The following best practices can help you manage and minimize your costs when using Amazon DocumentDB. For pricing information, see [Amazon DocumentDB \(with MongoDB compatibility\) pricing](#) and [Amazon DocumentDB \(with MongoDB compatibility\) FAQs](#).

- Create billing alerts at thresholds of 50 percent and 75 percent of your expected bill for the month. For more information about creating billing alerts, see [Creating a Billing Alarm](#).
- Amazon DocumentDB's architecture separates storage and compute, so even a single-instance cluster is highly durable. The cluster storage volume replicates data six ways across three Availability Zones, providing extremely high durability regardless of the number of instances in the cluster. A typical

production cluster has three or more instances to provide high availability. However, you can optimize costs by using a single instance development cluster when high availability is not required.

- For development and test scenarios, stop a cluster when it is no longer needed and start the cluster when development resumes. For more information, see [Stopping and Starting an Amazon DocumentDB Cluster \(p. 293\)](#).
- Both TTL and change streams incur I/O's when data is written, read, and deleted. If you have enabled these features but are not utilizing them in your application, disabling the features can help reduce costs.

Using Metrics to Identify Performance Issues

To identify performance issues caused by insufficient resources and other common bottlenecks, you can monitor the metrics available for your Amazon DocumentDB cluster.

Viewing Performance Metrics

Monitor performance metrics on a regular basis to see the average, maximum, and minimum values for a variety of time ranges. This helps you identify when performance is degraded. You can also set Amazon CloudWatch alarms for particular metric thresholds so that you are alerted if they are reached.

To troubleshoot performance issues, it's important to understand the baseline performance of the system. After you set up a new cluster and get it running with a typical workload, capture the average, maximum, and minimum values of all the performance metrics at different intervals (for example, 1 hour, 24 hours, 1 week, 2 weeks). This gives you an idea of what is normal. It helps to get comparisons for both peak and off-peak hours of operation. You can then use this information to identify when performance is dropping below standard levels.

You can view performance metrics using the AWS Management Console or AWS CLI. For more information, see the following:

- [View CloudWatch Metrics Using the Amazon DocumentDB Management Console \(p. 422\)](#)
- [View CloudWatch Metrics Using the AWS CLI \(p. 423\)](#)

Setting a CloudWatch Alarm

To set a CloudWatch alarm, see [Using Amazon CloudWatch Alarms](#) in the *Amazon CloudWatch User Guide*.

Evaluating Performance Metrics

An instance has several different categories of metrics. How you determine acceptable values depends on the metric.

CPU

- **CPU Utilization** — The percentage of the computer processing capacity used.

Memory

- **Freeable Memory** — How much RAM is available on the instance.
- **Swap Usage** — How much swap space is used by the instance, in megabytes.

Input/output operations

- **Read IOPS, Write IOPS** — The average number of disk read or write operations per second.
- **Read Latency, Write Latency** — The average time for a read or write operation in milliseconds.
- **Read Throughput, Write Throughput** — The average number of megabytes read from or written to disk per second.
- **Disk Queue Depth** — The number of I/O operations that are waiting to be written to or read from disk.

Network traffic

- **Network Receive Throughput, Network Transmit Throughput** — The rate of network traffic to and from the instance in megabytes per second.

Database connections

- **DB Connections** — The number of client sessions that are connected to the instance.

Generally speaking, acceptable values for performance metrics depend on what your baseline looks like and what your application is doing. Investigate consistent or trending variances from your baseline.

The following are recommendations and advice about specific types of metrics:

- **High CPU consumption** — High values for CPU consumption might be appropriate, provided that they are in keeping with your goals for your application (like throughput or concurrency) and are expected. If your CPU consumption is consistently over 80 percent, consider scaling up your instances.
- **High RAM consumption** — If your FreeableMemory metric frequently dips below 10% of the total instance memory, consider scaling up your instances. For more information on what happens when your DocumentDB instance is experiencing high memory pressure, see [Amazon DocumentDB Resource Governance \(p. 12\)](#).
- **Swap usage** — This metric should remain at or near zero. If your swap usage is significant, consider scaling up your instances.
- **Network traffic** — For network traffic, talk with your system administrator to understand what the expected throughput is for your domain network and internet connection. Investigate network traffic if throughput is consistently lower than expected.
- **Database connections** — Consider constraining database connections if you see high numbers of user connections together with decreases in instance performance and response time. The best number of user connections for your instance varies based on your instance class and the complexity of the operations being performed. For issues with any performance metrics, one of the first things you can do to improve performance is tune the most used and most expensive queries to see if that lowers the pressure on system resources.

If your queries are tuned and an issue persists, consider upgrading your Amazon DocumentDB instance class to one with more of the resource (CPU, RAM, disk space, network bandwidth, I/O capacity) that is related to the issue you're experiencing.

Tuning Queries

One of the best ways to improve cluster performance is to tune your most commonly used and most resource-intensive queries to make them less expensive to run.

You can use the profiler (see [Profiling Amazon DocumentDB Operations \(p. 427\)](#)) to log the execution time and details of operations that were performed on your cluster. Profiler is useful for monitoring the

slowest operations on your cluster to help you improve individual query performance and overall cluster performance.

You can also use the `explain` command to learn how to analyze a query plan for a particular query. Use this information to modify a query or underlying collection to improve your query performance (for example, adding an index).

TTL and Timeseries Workloads

Document deletion resulting from TTL index expiry is a best effort process. Documents are not guaranteed to be deleted within any specific period. Factors like instance size, instance resource utilization, document size, overall throughput, the number of indexes, and whether indexes and the working set fit in memory can all affect the timing of when expired documents are deleted by the TTL process.

When the TTL monitor deletes your documents, each deletion incurs I/O costs, which increases your bill. If throughput and TTL delete rates increase, you should expect a higher bill due to increased I/O usage. However, if you do not create a TTL index to delete documents, but instead segment documents into collections based on time and simply drop those collections when they are no longer needed, you will not incur any IO costs. This can be significantly more cost effective than using a TTL index.

For time-series workloads, you can consider creating rolling collections instead of a TTL index as rolling collections can be a more performant way to delete data and less I/O intensive. If you have large collections (especially collections over 1TB) or TTL deletion I/O costs are a concern, we recommend that you partition documents into collections based on time, and drop collections when the documents are no longer needed. You can create one collection per day or one per week, depending on your data ingest rate. While requirements will vary depending on your application, a good rule of thumb is to have more smaller collections rather than a few large collections. Dropping these collections does not incur I/O costs, and can be faster and more cost effective than using a TTL index.

Migrations

As a best practice, we recommend that when migrating data to Amazon DocumentDB, you first create your indexes in Amazon DocumentDB before migrating the data. Creating the indexes first can reduce the overall time and increase the speed of the migration. To do this, you can use the Amazon DocumentDB [Index Tool](#). For more information on migrations, see the [Amazon DocumentDB migration guide](#).

We also recommend that before you migrate your production database, it is a best practice to fully test your application on Amazon DocumentDB, taking into consideration functionality, performance, operations, and cost.

Working with Cluster Parameter Groups

We recommend that you try out cluster parameter group changes on a test cluster before applying the changes to your production clusters. For information about backing up your cluster, see [Backing Up and Restoring in Amazon DocumentDB \(p. 215\)](#).

Aggregation Pipeline Queries

When creating an aggregation pipeline query with multiple stages and evaluating only a subset of the data in the query, use the `$match` stage as the first stage or in the beginning of the pipeline. Using

`$match` first will reduce the number of documents subsequent stages within the aggregation pipeline query will need to process, thus improving the performance of your query.

batchInsert and batchUpdate

When performing a high rate of concurrent batchInsert and/or batchUpdate operations, and the amount of FreeableMemory (CloudWatch Metric) goes to zero on your primary instance, you can either reduce the concurrency of the batch insert or update workload or, if concurrency of the workload cannot be reduced, increase the instance size to increase the amount of FreeableMemory.

Functional Differences: Amazon DocumentDB and MongoDB

The following are the functional differences between Amazon DocumentDB (with MongoDB compatibility) and MongoDB.

Topics

- [Functional Benefits of Amazon DocumentDB \(p. 92\)](#)
- [Updated Functional Differences \(p. 93\)](#)
- [Functional Differences with MongoDB \(p. 95\)](#)

Functional Benefits of Amazon DocumentDB

Implicit Transactions

In Amazon DocumentDB, all CRUD statements (`findAndModify`, `update`, `insert`, `delete`) guarantee atomicity and consistency, even for operations that modify multiple documents. With the launch of Amazon DocumentDB 4.0, explicit transactions that provide ACID properties for multi-statement and multi-collection operations are now supported. For more on using transactions in Amazon DocumentDB, please see [Transactions \(p. 54\)](#).

The following are examples of operations in Amazon DocumentDB that modify multiple documents that satisfy both atomic and consistent behaviors.

```
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 }]  
})
```

The individual operations that compose bulk operations such as updateMany and deleteMany are atomic but the entirety of the bulk operation is not atomic. For example, the entirety of the insertMany operation is atomic if the individual insert operations execute successfully without error. If an error is encountered with an insertMany operation, each individual insert statement within the insertMany operation will execute as an atomic operation. If you require ACID properties for insertMany, updateMany, and deleteMany operations, it is recommended to use a transaction.

Updated Functional Differences

Amazon DocumentDB continues to improve compatibility with MongoDB by working backwards from the capabilities our customers ask us to build. This section contains the functional differences that we have removed in Amazon DocumentDB to make migrations and building applications easier for our customers.

Topics

- [Array Indexing \(p. 93\)](#)
- [Multi-key Indexes \(p. 94\)](#)
- [Null Characters in Strings \(p. 94\)](#)
- [Role-Based Access Control \(p. 94\)](#)
- [\\$regex Indexing \(p. 95\)](#)
- [Projection for Nested Documents \(p. 95\)](#)

Array Indexing

As of April 23, 2020, Amazon DocumentDB now supports the ability to index arrays that are greater than 2,048 bytes. The limit for an individual item in an array still remains as 2,048 bytes, which is consistent with MongoDB.

If you are creating a new index, no action is needed to take advantage of the improved functionality. If you have an existing index, you can take advantage of the improved functionality by dropping the index and then recreating it. The current index version with the improved capabilities is "v" : 3.

Note

For production clusters, the dropping of the index may have an impact on your application performance. We recommend that you first test and proceed with caution when making changes to a production system. In addition, the time it will take to recreate the index will be a function of the overall data size of the collection.

You can query for the version of your indexes using the following command.

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

Output from this operation looks something like the following. In this output, the version of the index is "v" : 3, which is the most current index version.

```
[  
  {  
    "v" : 3,  
    "key" : {  
      "_id" : 1  
    },  
    "name" : "_id_",  
    "ns" : "test.test"  
  }  
]
```

Multi-key Indexes

As of April 23, 2020, Amazon DocumentDB now supports the ability to create a compound index with multiple keys in the same array.

If you are creating a new index, no action is needed to take advantage of the improved functionality. If you have an existing index, you can take advantage of the improved functionality by dropping the index and then recreating it. The current index version with the improved capabilities is "v" : 3.

Note

For production clusters, the dropping of the index may have an impact on your application performance. We recommend that you first test and proceed with caution when making changes to a production system. In addition, the time it will take to recreate the index will be a function of the overall data size of the collection.

You can query for the version of your indexes using the following command.

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

Output from this operation looks something like the following. In this output, the version of the index is "v" : 3, which is the most current index version.

```
[  
  {  
    "v" : 3,  
    "key" : {  
      "_id" : 1  
    },  
    "name" : "_id_",  
    "ns" : "test.test"  
  }  
]
```

Null Characters in Strings

As of June 22, 2020, Amazon DocumentDB now supports null characters ('\0') in strings.

Role-Based Access Control

As of March 26, 2020, Amazon DocumentDB supports role-based access control (RBAC) for built-in roles. To learn more, see [Role-Based Access Control \(p. 169\)](#). Amazon DocumentDB does not yet support custom roles for RBAC.

\$regex Indexing

As of June 22, 2020, Amazon DocumentDB now supports the ability for \$regex operators to utilize an index.

To utilize an index with the \$regex operator, you must use the `hint()` command. When using `hint()`, you must specify the name of the field you are applying the \$regex on. For example, if you have an index on field `product` with the index name as `p_1`, `db.foo.find({product: /
^x.*/}).hint({product:1})` will utilize the `p_1` index, but `db.foo.find({product: /
^x.*/}).hint("p_1")` will not utilize the index. You can verify if an index is chosen by utilizing the `explain()` command or using the profiler for logging slow queries. For example, `db.foo.find({product: /
^x.*/}).hint("p_1").explain()`.

Note

The `hint()` method can only be used with one index at a time.

The use of an index for a \$regex query is optimized for regex queries that utilize a prefix and do not specify the I, m, or o regex options.

When using an index with \$regex, it is recommended that you create an index on highly selective fields where the number of duplicate values is less than 1% of the total number of documents in the collection. As an example, if your collection contains 100,000 documents, only create indexes on fields where the same value occurs 1000 times or fewer.

Projection for Nested Documents

There is a functional difference with \$project operator between Amazon DocumentDB and MongoDB in version 3.6 that has been resolved in Amazon DocumentDB 4.0 but will remain unsupported in Amazon DocumentDB 3.6.

Amazon DocumentDB 3.6 only considers the first field in a nested document when applying a projection whereas MongoDB 3.6 will parse subdocuments and apply the projection to each sub document as well.

For example: if the projection is “`a.b.c`” : 1, then the behavior works as expect in both Amazon DocumentDB and MongoDB. However, if the projection is `{a:{b:{c:1}}}` then Amazon DocumentDB 3.6 will only apply the projection to `a` and not `b` or `c`. In Amazon DocumentDB 4.0, the projection `{a:{b:{c:1}}}` will be applied to `a`, `b`, and `c`.

Functional Differences with MongoDB

Topics

- [Admin Databases and Collections \(p. 96\)](#)
- [cursormaxTimeMS \(p. 96\)](#)
- [explain\(\) \(p. 96\)](#)
- [Field Name Restrictions \(p. 96\)](#)
- [Index Builds \(p. 97\)](#)
- [Lookup with empty key in path \(p. 97\)](#)
- [MongoDB APIs, Operations, and Data Types \(p. 97\)](#)
- [mongodump and mongorestore Utilities \(p. 97\)](#)
- [Result Ordering \(p. 97\)](#)
- [Retryable Writes \(p. 98\)](#)

- [Sparse Index \(p. 98\)](#)
- [Storage Compression \(p. 98\)](#)
- [Using \\$elemMatch Within an \\$all Expression \(p. 98\)](#)
- [\\$ne, \\$nin, \\$nor, \\$not, \\$exists, and \\$elemMatch Indexing \(p. 99\)](#)
- [\\$lookup \(p. 99\)](#)

Admin Databases and Collections

Amazon DocumentDB does not support the admin or local database nor MongoDB system.* or startup_log collections respectively.

cursor.maxTimeMS

In Amazon DocumentDB, `cursor.maxTimeMS` resets the counter for each `getMore` request. Thus, if a 3000MS `maxTimeMS` is specified, the query takes 2800MS, and each subsequent `getMore` request takes 300MS, then the cursor will not timeout. The cursor will only timeout when a single operations, either the query or an individual `getMore` request, takes more than the specified `maxTimeMS`. Further, the sweeper that checks cursor execution time runs at a minute granularity.

explain()

Amazon DocumentDB emulates the MongoDB 4.0 API on a purpose-built database engine that utilizes a distributed, fault-tolerant, self-healing storage system. As a result, query plans and the output of `explain()` may differ between Amazon DocumentDB and MongoDB. Customers who want control over their query plan can use the `$hint` operator to enforce selection of a preferred index.

Field Name Restrictions

Amazon DocumentDB does not support dots `.` in a document field name, for example, `db.foo.insert({‘x.1’:1})`.

Amazon DocumentDB also does not support the `$` prefix in field names.

For example, try the following command in Amazon DocumentDB or MongoDB:

```
rs0:PRIMARY< db.foo.insert({"a":{"$a":1}})
```

MongoDB will return the following:

```
WriteResult({ "nInserted" : 1 })
```

Amazon DocumentDB will return an error:

```
WriteResult({
  "nInserted" : 0,
  "writeError" : {
    "code" : 2,
    "errmsg" : "Document can't have $ prefix field names: $a"
  }
})
```

Note

There is an exception to this functional difference. The following field names that begin with the \$ prefix have been whitelisted and can be successfully used in Amazon DocumentDB: \$id, \$ref and \$db.

Index Builds

Amazon DocumentDB allows only one index build to occur on a collection at any given time. Either in the foreground or the background. If operations such as `createIndex()` or `dropIndex()` occur on the same collection when an index build is currently in progress, the newly attempted operation will fail.

By default, index builds in Amazon DocumentDB and MongoDB version 4.0 occur in the background. MongoDB version 4.2, and later ignores the background index build option if specified to `createIndexes()` or its shell helpers `createIndex()` and `createIndexes()`.

A Time to Live (TTL) index starts expiring documents after the index build is completed.

Lookup with empty key in path

When you look up with a key that includes empty string as part of the path (e.g. `x..`, `x..b`), and the object has an empty string key path (e.g. `{"x" : [{ "" : 10 }, { "b" : 20 }]}`) inside an array, Amazon DocumentDB will return different results than if you were to run the same look up in MongoDB.

In MongoDB, the empty key path look up within array works as expected when the empty string key is not at the end of path look up. However, when the empty string key is at the end of path look up, it does not look into the array.

However in Amazon DocumentDB, only the first element within the array is read, because `getArrayIndexFromKeyString` converts empty string to `0`, so string key look up is treated as array index look up.

MongoDB APIs, Operations, and Data Types

Amazon DocumentDB is compatible with the MongoDB 3.6 and 4.0 APIs. For an up-to-date list of supported functionality, see [Supported MongoDB APIs, Operations, and Data Types \(p. 102\)](#).

mongodump and mongorestore Utilities

Amazon DocumentDB does not support an admin database and thus does not dump or restore the admin database when using the `mongodump` or `mongorestore` utilities. When you create a new database in Amazon DocumentDB using `mongorestore`, you need to re-create the user roles in addition to the restore operation.

Result Ordering

Amazon DocumentDB does not guarantee implicit result sort ordering of result sets. To ensure the ordering of a result set, explicitly specify a sort order using `sort()`.

The following example sorts the items in the `inventory` collection in descending order based on the `stock` field.

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

When using the \$sort aggregation stage, the sort order is not preserved unless the \$sort stage is the last stage in the aggregation pipeline. When using the \$sort aggregation stage in combination with the \$group aggregation stage, the \$sort aggregation stage is only applied to the \$first and \$last accumulators. In Amazon DocumentDB 4.0, support was added for \$push to respect sort order from the previous \$sort stage.

Retryable Writes

Starting with MongoDB 4.2 compatible drivers, retryable writes is enabled by default. However, Amazon DocumentDB does not currently support retryable writes. The functional difference will manifest itself in an error message similar to the following.

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

Retryable writes can be disabled via the connection string (for example, MongoClient("mongodb://my.mongodb.cluster/db?retryWrites=false")) or the MongoClient constructor's keyword argument (for example, MongoClient("mongodb://my.mongodb.cluster/db", retryWrites=False)).

The following is a Python example that disables retryable writes in the connection string.

```
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)
```

Sparse Index

To use a sparse index that you have created in a query, you must use the \$exists clause on the fields that cover the index. If you omit \$exists, Amazon DocumentDB does not use the sparse index.

The following is an example.

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

For sparse, multi-key indexes, Amazon DocumentDB does not support a unique key constraint if the look up of a document results in a set of values and only a subset of the indexed fields is missing. For example, createIndex({"a.b" : 1}, { unique : true, sparse :true }) is not supported, given the input of "a" : [{ "b" : 2 }, { "c" : 1 }], as "a.c" is stored in the index.

Storage Compression

Amazon DocumentDB doesn't currently support compression for stored data or indexes. Data sizes for stored data and indexes might be larger than when you use other options.

Using \$elemMatch Within an \$all Expression

Amazon DocumentDB does not currently support the use of the \$elemMatch operator within an \$all expression. As a workaround, you can use the \$and operator with \$elemMatch as follows.

Original operation:

```
db.col.find({  
    qty: {
```

```

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

```

Updated operation:

```

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

```

\$ne, \$nin, \$nor, \$not, \$exists, and \$elemMatch Indexing

Amazon DocumentDB does not currently support the ability to use indexes with the \$ne, \$nin, \$nor, \$not, \$exists, \$distinct, and \$elemMatch operators. As a result, utilizing these operators will result in collection scans. Performing a filter or match before utilizing one of these operators will reduce the amount of data that needs to be scanned, and thus can improve performance.

\$lookup

Amazon DocumentDB supports the ability to do equality matches (for example, left outer join) but does not support uncorrelated subqueries.

Utilizing an index with \$lookup

You can now utilize an index with the \$lookup stage operator. Based on your use case, there are multiple indexing algorithms that you can use to optimize for performance. This section will explain the different indexing algorithms for \$lookup and help you choose the best one for your workload.

By default, Amazon DocumentDB will utilize the hash algorithm when allowDiskUse:false is used and sort merge when allowDiskUse:true is used. For some use cases, it may be desirable to force the query optimizer to use a different algorithm. Below are the different indexing algorithms that the \$lookup aggregation operator can utilize:

- Nested loop:** A nested loop plan is typically beneficial for a workload if the foreign collection is <1 GB and the field in the foreign collection has an index. If the nested loop algorithm is being used, the explain plan will show the stage as NESTED_LOOP_LOOKUP.
- Sort merge:** A sort merge plan is typically beneficial for a workload if the foreign collection does not have an index on the field used in lookup and the working dataset doesn't fit in memory. If the sort merge algorithm is being used, the explain plan will show the stage as SORT_LOOKUP.
- Hash:** A hash plan is typically beneficial for a workload if the foreign collection is < 1GB and the working dataset fits in memory. If the hash algorithm is being used, the explain plan will show the stage as HASH_LOOKUP.

You can identify the indexing algorithm that is being used for the \$lookup operator by using explain on the query. Below is an example.

```

db.localCollection.explain().
aggregate( [
    {
        $lookup:
            {
                from: "foreignCollection",
                localField: "a",
                foreignField: "b",
                as: "joined"
            }
    }
],
{
    "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
    }
]
)

```

As an alternative to the using the `explain()` method, you can use the profiler to review the algorithm that is being utilized with your use of the `$lookup` operator. For more information on the profiler, please see [Profiling Amazon DocumentDB Operations \(p. 427\)](#).

Using a planHint

If you wish to force the query optimizer to use a different indexing algorithm with `$lookup`, you can use a `planHint`. To do that, use the comment in the aggregation stage options to force a different plan. Below is an example of the syntax for the comment:

```
comment : {
```

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

Below is an example of using the `planHint` to force the query optimizer to use the HASH indexing algorithm:

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

To test which algorithm is best for your workload, you can use the `executionStats` parameter of the `explain` method to measure the execution time of the `$lookup` stage while modifying the indexing algorithm (i.e., HASH/SORT/NESTED_LOOP).

The following example shows how to use `executionStats` to measure the execution time of the `$lookup` stage using the SORT algorithm.

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

Supported MongoDB APIs, Operations, and Data Types

Amazon DocumentDB (with MongoDB compatibility) is a fast, scalable, highly-available, and fully managed document database service that supports MongoDB workloads. Amazon DocumentDB is compatible with the MongoDB 3.6 and 4.0 APIs. This section lists the supported functionality. For support using MongoDB APIs and drivers, please consult the MongoDB Community Forums. For support using the Amazon DocumentDB service, please contact the appropriate AWS support team. For functional differences between Amazon DocumentDB and MongoDB, please see [Functional Differences: Amazon DocumentDB and MongoDB \(p. 92\)](#).

MongoDB commands and operators that are internal-only or not applicable to a fully-managed service are not supported and are not included in the list of supported functionality.

We have added over 50+ additional capabilities since launch, and will continue to work backwards from our customers to deliver the capabilities that they need. For information on the most recent launches, see [Amazon DocumentDB Announcements](#).

If there is a feature that isn't supported that you'd like us to build, let us know by sending an email with your accountID, the requested features, and use case to the [Amazon DocumentDB service team](#).

Topics

- [Database Commands \(p. 102\)](#)
- [Query and Projection Operators \(p. 106\)](#)
- [Update Operators \(p. 108\)](#)
- [Geospatial \(p. 110\)](#)
- [Cursor Methods \(p. 111\)](#)
- [Aggregation Pipeline Operators \(p. 112\)](#)
- [Data Types \(p. 119\)](#)
- [Indexes and Index Properties \(p. 120\)](#)

Database Commands

Topics

- [Administrative Commands \(p. 103\)](#)
- [Aggregation \(p. 103\)](#)
- [Authentication \(p. 104\)](#)
- [Diagnostic Commands \(p. 104\)](#)
- [Query and Write Operations \(p. 104\)](#)
- [Role Management Commands \(p. 105\)](#)
- [Sessions Commands \(p. 105\)](#)

- [User Management \(p. 106\)](#)

Administrative Commands

Command	3.6	4.0
Capped Collections	No	No
cloneCollectionAsCapped	No	No
collMod	Partial	Partial
collMod: expireAfterSeconds	Yes	Yes
convertToCapped	No	No
copydb	No	No
create	Yes	Yes
createView	No	No
createIndexes	Yes	Yes
currentOp	Yes	Yes
drop	Yes	Yes
dropDatabase	Yes	Yes
dropIndexes	Yes	Yes
filemd5	No	No
killCursors	Yes	Yes
killOp	Yes	Yes
listCollections	Yes	Yes
listDatabases	Yes	Yes
listIndexes	Yes	Yes
reIndex	No	No
renameCollection	Yes	Yes

Aggregation

Command	3.6	4.0
aggregate	Yes	Yes
count	Yes	Yes
distinct	Yes	Yes

Command	3.6	4.0
mapReduce	No	No

Authentication

Command	3.6	4.0
authenticate	Yes	Yes
logout	Yes	Yes

Diagnostic Commands

Command	3.6	4.0
buildInfo	Yes	Yes
collStats	Yes	Yes
connPoolStats	No	No
connectionStatus	Yes	Yes
dataSize	Yes	Yes
dbHash	No	No
dbStats	Yes	Yes
explain	Yes	Yes
explain: executionStats	Yes	Yes
features	No	No
hostInfo	Yes	Yes
listCommands	Yes	Yes
profiler	Yes	Yes
serverStatus	Yes	Yes
top	Yes	Yes

Query and Write Operations

Command	3.6	4.0
delete	Yes	Yes
find	Yes	Yes

Command	3.6	4.0
findAndModify	Yes	Yes
getLastError	No	No
getMore	Yes	Yes
getPrevError	No	No
insert	Yes	Yes
parallelCollectionScan	No	No
resetError	No	No
update	Yes	Yes
Change streams	Yes	Yes
GridFS	No	No

Role Management Commands

Command	3.6	4.0
createRole	Yes	Yes
dropRole	Yes	Yes
dropAllRolesFromDatabase	No	No
grantRolesToRole	Yes	Yes
revokePrivilegesFromRole	Yes	Yes
revokeRolesFromRole	Yes	Yes
updateRole	Yes	Yes
rolesInfo	No	No

Sessions Commands

Command	3.6	4.0
abortTransaction	No	Yes
commitTransaction	No	Yes
endSessions	No	Yes
killAllSessions	No	Yes
killAllSessionsByPattern	No	No
killSessions	No	Yes

Command	3.6	4.0
refreshSessions	No	No
startSession	No	Yes

User Management

Command	3.6	4.0
createUser	Yes	Yes
dropAllUsersFromDatabase	Yes	Yes
dropUser	Yes	Yes
grantRolesToUser	Yes	Yes
revokeRolesFromUser	Yes	Yes
updateUser	Yes	Yes
userInfo	Yes	Yes

Query and Projection Operators

Topics

- [Array Operators \(p. 106\)](#)
- [Bitwise Operators \(p. 107\)](#)
- [Comment Operator \(p. 107\)](#)
- [Comparison Operators \(p. 107\)](#)
- [Element Operators \(p. 107\)](#)
- [Evaluation Query Operators \(p. 108\)](#)
- [Logical Operators \(p. 108\)](#)
- [Projection Operators \(p. 108\)](#)

Array Operators

Command	3.6	4.0
\$all	Yes	Yes
\$elemMatch	Yes	Yes
\$size	Yes	Yes

Bitwise Operators

Command	3.6	4.0
\$bitsAllSet	Yes	Yes
\$bitsAnySet	Yes	Yes
\$bitsAllClear	Yes	Yes
\$bitsAnyClear	Yes	Yes

Comment Operator

Command	3.6	4.0
\$comment	Yes	Yes

Comparison Operators

Command	3.6	4.0
\$eq	Yes	Yes
\$gt	Yes	Yes
\$gte	Yes	Yes
\$lt	Yes	Yes
\$lte	Yes	Yes
\$ne	Yes	Yes
\$in	Yes	Yes
\$nin	Yes	Yes

Element Operators

Command	3.6	4.0
\$exists	Yes	Yes
\$type	Yes	Yes

Evaluation Query Operators

Command	3.6	4.0
\$expr	No	No
\$jsonSchema	No	No
\$mod	Yes	Yes
\$regex	Yes	Yes
\$text	No	No
\$where	No	No

Logical Operators

Command	3.6	4.0
\$or	Yes	Yes
\$and	Yes	Yes
\$not	Yes	Yes
\$nor	Yes	Yes

Projection Operators

Command	3.6	4.0
\$	Yes	Yes
\$elemMatch	Yes	Yes
\$meta	No	No
\$slice	Yes	Yes

Update Operators

Topics

- [Array Operators \(p. 109\)](#)
- [Bitwise Operators \(p. 109\)](#)
- [Field Operators \(p. 109\)](#)
- [Update Modifiers \(p. 110\)](#)

Array Operators

Command	3.6	4.0
\$	Yes	Yes
\$[]	Yes	Yes
\$[<identifier>]	Yes	Yes
\$addToSet	Yes	Yes
\$pop	Yes	Yes
\$pullAll	Yes	Yes
\$pull	Yes	Yes
\$push	Yes	Yes

Bitwise Operators

Command	3.6	4.0
\$bit	Yes	Yes

Field Operators

Operator	3.6	4.0
\$inc	Yes	Yes
\$mul	Yes	Yes
\$rename	Yes	Yes
\$setOnInsert	Yes	Yes
\$set	Yes	Yes
\$unset	Yes	Yes
\$min	Yes	Yes
\$max	Yes	Yes
\$currentDate	Yes	Yes

Update Modifiers

Operator	3.6	4.0
\$each	Yes	Yes
\$slice	Yes	Yes
\$sort	Yes	Yes
\$position	Yes	Yes

Geospatial

Geometry Specifiers

Query Selectors	3.6	4.0
\$box	No	No
\$center	No	No
\$centerSphere	No	No
\$nearSphere	Yes	Yes
\$geometry	Yes	Yes
\$maxDistance	Yes	Yes
\$minDistance	Yes	Yes
\$polygon	No	No
\$uniqueDocs	No	No

Query Selectors

Command	3.6	4.0
\$geoIntersects	Yes	Yes
\$geoWithin	Yes	Yes
\$near	No	No
\$nearSphere	Yes	Yes
\$polygon	No	No
\$uniqueDocs	No	No

Cursor Methods

Command	3.6	4.0
cursor.batchSize()	Yes	Yes
cursor.close()	Yes	Yes
cursor.isClosed()	Yes	Yes
cursor.collation()	No	No
cursor.comment()	Yes	Yes
cursor.count()	Yes	Yes
cursor.explain()	Yes	Yes
cursor.forEach()	Yes	Yes
cursor.hasNext()	Yes	Yes
cursor_hint()	Yes	Yes
cursor.isExhausted()	Yes	Yes
cursor_itcount()	Yes	Yes
cursor_limit()	Yes	Yes
cursor_map()	Yes	Yes
cursor_maxScan()	Yes	Yes
cursor_maxTimeMS()	Yes	Yes
cursor_max()	No	No
cursor_min()	No	No
cursor_next()	Yes	Yes
cursor_noCursorTimeout()	No	No
cursor_objsLeftInBatch()	Yes	Yes
cursor_pretty()	Yes	Yes
cursor_readConcern()	Yes	Yes
cursor_readPref()	Yes	Yes
cursor_returnKey()	No	No
cursor_showRecordId()	No	No
cursor_size()	Yes	Yes
cursor_skip()	Yes	Yes
cursor_sort()	Yes	Yes

Command	3.6	4.0
cursor.tailable()	No	No
cursor.toArray()	Yes	Yes

Aggregation Pipeline Operators

Topics

- [Accumulator Expressions \(p. 112\)](#)
- [Arithmetic Operators \(p. 113\)](#)
- [Array Operators \(p. 113\)](#)
- [Boolean Operators \(p. 114\)](#)
- [Comparison Operators \(p. 114\)](#)
- [Conditional Expression Operators \(p. 114\)](#)
- [Data Type Operator \(p. 115\)](#)
- [Date Operators \(p. 115\)](#)
- [Literal Operator \(p. 115\)](#)
- [Merge Operator \(p. 116\)](#)
- [Natural Operator \(p. 116\)](#)
- [Set Operators \(p. 116\)](#)
- [Stage Operators \(p. 116\)](#)
- [String Operators \(p. 117\)](#)
- [System Variables \(p. 118\)](#)
- [Text Search Operator \(p. 118\)](#)
- [Type Conversion Operators \(p. 118\)](#)
- [Variable Operators \(p. 119\)](#)

Accumulator Expressions

Expression	3.6	4.0
\$sum	Yes	Yes
\$avg	Yes	Yes
\$first	Yes	Yes
\$last	Yes	Yes
\$max	Yes	Yes
\$min	Yes	Yes
\$push	Yes	Yes
\$addToSet	Yes	Yes
\$stdDevPop	No	No

Expression	3.6	4.0
\$stdDevSamp	No	No

Arithmetic Operators

Command	3.6	4.0
\$abs	Yes	Yes
\$add	Yes	Yes
\$ceil	No	No
\$divide	Yes	Yes
\$exp	No	No
\$floor	No	No
\$ln	No	No
\$log	No	No
\$log10	No	No
\$mod	Yes	Yes
\$multiply	Yes	Yes
\$pow	No	No
\$sqrt	No	No
\$subtract	Yes	Yes
\$trunc	No	No

Array Operators

Command	3.6	4.0
\$arrayElemAt	Yes	Yes
\$arrayToObject	Yes	Yes
\$concatArrays	Yes	Yes
\$filter	Yes	Yes
\$indexOfArray	Yes	Yes
\$isArray	Yes	Yes
\$objectToArray	Yes	Yes
\$range	Yes	Yes

Command	3.6	4.0
\$reverseArray	Yes	Yes
\$reduce	Yes	Yes
\$size	Yes	Yes
\$slice	Yes	Yes
\$zip	Yes	Yes
\$in	Yes	Yes

Boolean Operators

Command	3.6	4.0
\$and	Yes	Yes
\$or	Yes	Yes
\$not	Yes	Yes

Comparison Operators

Command	3.6	4.0
\$cmp	Yes	Yes
\$eq	Yes	Yes
\$gt	Yes	Yes
\$gte	Yes	Yes
\$lt	Yes	Yes
\$lte	Yes	Yes
\$ne	Yes	Yes

Conditional Expression Operators

Command	3.6	4.0
\$cond	Yes	Yes
\$ifNull	Yes	Yes
\$switch	No	No

Data Type Operator

Command	3.6	4.0
\$type	Yes	Yes

Date Operators

Command	3.6	4.0
\$dayOfYear	Yes	Yes
\$dayOfMonth	Yes	Yes
\$dayOfWeek	Yes	Yes
\$year	Yes	Yes
\$month	Yes	Yes
\$week	Yes	Yes
\$hour	Yes	Yes
\$minute	Yes	Yes
\$second	Yes	Yes
\$millisecond	Yes	Yes
\$dateToString	Yes	Yes
\$isoDayOfWeek	Yes	Yes
\$isoWeek	Yes	Yes
\$dateFromParts	No	No
\$dateToParts	No	No
\$dateFromString	Yes	Yes
\$isoWeekYear	Yes	Yes

Literal Operator

Command	3.6	4.0
\$literal	Yes	Yes

Merge Operator

Command	3.6	4.0
\$mergeObjects	Yes	Yes

Natural Operator

Command	3.6	4.0
\$natural	Yes	Yes

Set Operators

Command	3.6	4.0
\$setEquals	Yes	Yes
\$setIntersection	Yes	Yes
\$setUnion	Yes	Yes
\$setDifference	No	No
\$setIsSubset	Yes	Yes
\$anyElementTrue	No	No
\$allElementsTrue	No	No

Stage Operators

Command	3.6	4.0
\$collStats	No	No
\$project	Yes	Yes
\$match	Yes	Yes
\$redact	Yes	Yes
\$limit	Yes	Yes
\$skip	Yes	Yes
\$unwind	Yes	Yes
\$group	Yes	Yes
\$sample	Yes	Yes

Command	3.6	4.0
\$sort	Yes	Yes
\$geoNear	Yes	Yes
\$lookup	Yes	Yes
\$out	Yes	Yes
\$indexStats	Yes	Yes
\$facet	No	No
\$bucket	No	No
\$bucketAuto	No	No
\$sortByCount	No	No
\$addFields	Yes	Yes
\$replaceRoot	Yes	Yes
\$count	Yes	Yes
\$currentOp	Yes	Yes
\$listLocalSessions	No	No
\$listSessions	No	No
\$graphLookup	No	No

String Operators

Command	3.6	4.0
\$concat	Yes	Yes
\$indexOfBytes	Yes	Yes
\$indexOfCP	Yes	Yes
\$ltrim	No	No
\$rtrim	No	No
\$split	Yes	Yes
\$strcasecmp	Yes	Yes
\$strLenBytes	Yes	Yes
\$strLenCP	Yes	Yes
\$substr	Yes	Yes
\$substrBytes	Yes	Yes

Command	3.6	4.0
\$substrCP	Yes	Yes
\$toLower	Yes	Yes
\$toUpper	Yes	Yes
\$trim	No	No

System Variables

Command	3.6	4.0
\$\$CURRENT	No	No
\$\$DESCEND	Yes	Yes
\$\$KEEP	Yes	Yes
\$\$PRUNE	Yes	Yes
\$\$REMOVE	No	No
\$\$ROOT	Yes	Yes

Text Search Operator

Command	3.6	4.0
\$meta	No	No

Type Conversion Operators

Command	3.6	4.0
\$convert	No	No
\$toBool	No	No
\$toDate	No	No
\$toDecimal	No	No
\$toDouble	No	No
\$toInt	No	No
\$toLong	No	No
\$objectId	No	No
\$toString	No	No

Variable Operators

Command	3.6	4.0
\$map	Yes	Yes
\$let	No	No

Data Types

Command	3.6	4.0
Double	Yes	Yes
String	Yes	Yes
Object	Yes	Yes
Array	Yes	Yes
Binary Data	Yes	Yes
ObjectId	Yes	Yes
Boolean	Yes	Yes
Date	Yes	Yes
Null	Yes	Yes
32-bit Integer (int)	Yes	Yes
Timestamp	Yes	Yes
64-bit Integer (long)	Yes	Yes
MinKey	Yes	Yes
MaxKey	Yes	Yes
Decimal128	Yes	Yes
Regular Expression	Yes	Yes
JavaScript	No	No
JavaScript (with scope)	No	No
Undefined	No	No
Symbol	No	No
DBPointer	No	

Indexes and Index Properties

Topics

- [Indexes \(p. 120\)](#)
- [Index Properties \(p. 120\)](#)

Indexes

Command	3.6	4.0
Single Field Index	Yes	Yes
Compound Index	Yes	Yes
Multikey Index	Yes	Yes
Text Index	No	No
2dsphere	Yes	Yes
2d Index	No	No
Hashed Index	No	No

Index Properties

Command	3.6	4.0
TTL	Yes	Yes
Unique	Yes	Yes
Partial	No	No
Case Insensitive	No	No
Sparse	Yes	Yes
Background	Yes	Yes

Migrating to Amazon DocumentDB

Amazon DocumentDB (with MongoDB compatibility) is a fully managed database service that is compatible with the MongoDB API. You can migrate your data to Amazon DocumentDB from MongoDB databases running on premises or on Amazon Elastic Compute Cloud (Amazon EC2) using the process detailed in this section.

Topics

- [Upgrading your Amazon DocumentDB cluster from 3.6 to 4.0 using AWS Database Migration Service \(p. 121\)](#)
- [Migration Tools \(p. 128\)](#)
- [Discovery \(p. 129\)](#)
- [Planning: Amazon DocumentDB Cluster Requirements \(p. 132\)](#)
- [Migration Approaches \(p. 134\)](#)
- [Migration Sources \(p. 137\)](#)
- [Migration Connectivity \(p. 138\)](#)
- [Testing \(p. 140\)](#)
- [Performance Testing \(p. 142\)](#)
- [Failover Testing \(p. 142\)](#)
- [Additional Resources \(p. 142\)](#)

Upgrading your Amazon DocumentDB cluster from 3.6 to 4.0 using AWS Database Migration Service

Important

Amazon DocumentDB does not follow the same support lifecycles as MongoDB and MongoDB's end-of-life schedule does not apply to Amazon DocumentDB. There are no current plans for end-of-life for Amazon DocumentDB 3.6, and your existing MongoDB 3.6 drivers, applications, and tools, will continue to work with Amazon DocumentDB.

You can upgrade your Amazon DocumentDB 3.6 cluster to 4.0 with minimal downtime using AWS DMS. AWS DMS is a fully managed service that makes it easy to migrate relational databases and non-relational databases to Amazon DocumentDB.

Topics

- [Step 1: Enable Change Streams \(p. 122\)](#)
- [Step 2: Modify the Change Streams Retention Duration \(p. 122\)](#)
- [Step 3: Migrate Your Indexes \(p. 122\)](#)
- [Step 4: Create a AWS DMS Replication Instance \(p. 123\)](#)
- [Step 5: Create an AWS DMS Source Endpoint \(p. 124\)](#)
- [Step 6: Create an AWS DMS Target Endpoint \(p. 126\)](#)

- [Step 7: Create and run a migration task \(p. 127\)](#)
- [Step 8: Changing the application endpoint to the Amazon DocumentDB cluster 4.0 \(p. 128\)](#)

Step 1: Enable Change Streams

To perform a minimal downtime migration, AWS DMS requires access to the cluster's change streams. [Amazon DocumentDB change streams](#) provide a time-ordered sequence of update events that occur within your cluster's collections and databases. Reading from the change stream enables AWS DMS to perform change data capture (CDC) and apply incremental updates to the target Amazon DocumentDB cluster.

To enable change streams for all collections on a specific database, authenticate to your Amazon DocumentDB cluster using the mongo shell and execute the following commands:

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

Step 2: Modify the Change Streams Retention Duration

Next, modify the change stream retention period based on how long you would like to retain change events in the change stream. For example, if you expect your AWS DMS migration from Amazon DocumentDB v3.6 to v4.0 to take 12 hours, you should set the change stream retention to a value greater than 12 hours. The default retention period for your Amazon DocumentDB cluster is three hours. You can modify the change stream log retention duration for your Amazon DocumentDB cluster to be between one hour and seven days using the AWS Management Console or the AWS CLI. For more details, refer to [Modifying the Change Stream Log Retention Duration](#).

Step 3: Migrate Your Indexes

Create the same indexes on your Amazon DocumentDB 4.0 cluster that you have on your Amazon DocumentDB 3.6 cluster. Although AWS DMS handles the migration of data, it does not migrate indexes. To migrate the indexes, use the Amazon DocumentDB Index Tool to export indexes from the Amazon DocumentDB 3.6 cluster. You can get the tool by creating a clone of the Amazon DocumentDB tools GitHub repo and following the instructions in [README.md](#). You can run the tool from an Amazon EC2 instance or an AWS Cloud9 environment running in the same Amazon VPC as your Amazon DocumentDB cluster.

The following code dumps indexes from your Amazon DocumentDB v3.6 cluster:

```
python migrationtools/documentdb_index_tool.py --dump-indexes  
--dir ~/index.js/  
--host docdb-36-xx.cluster-xxxxxxxx.us-west-2.docdb.amazonaws.com:27017  
--tls --tls-ca-file ~/rds-ca-2019-root.pem  
--username user  
--password <password>  
  
2020-02-11 21:51:23,245: Successfully authenticated to database: admin  
2020-02-11 21:46:50,432: Successfully connected to instance docdb-36-xx.cluster-  
xxxxxxxx.us-west-2.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/
```

Once your indexes are successfully exported, restore those indexes in your Amazon DocumentDB 4.0 cluster. To restore the indexes that you exported in the preceding step, use the Amazon DocumentDB Index Tool. The following command restores the indexes in your Amazon DocumentDB 4.0 cluster from the specified directory.

```
python migrationtools/documentdb_index_tool.py --restore-indexes  
--dir ~/index.js/  
--host docdb-40-xx.cluster-xxxxxxxx.us-west-2.docdb.amazonaws.com:27017  
--tls --tls-ca-file ~/rds-ca-2019-root.pem  
--username user  
--password <password>  
  
2020-02-11 21:51:23,245: Successfully authenticated to database: admin  
2020-02-11 21:51:23,245: Successfully connected to instance docdb-40-xx.cluster-  
xxxxxxxx.us-west-2.docdb.amazonaws.com:27017  
2020-02-11 21:51:23,264: testdb.coll: added index: _id
```

To confirm that you restored the indexes correctly, connect to your Amazon DocumentDB 4.0 cluster with the mongo shell and list the indexes for a given collection. See the following code:

```
mongo --ssl  
--host docdb-40-xx.cluster-xxxxxxxx.us-west-2.docdb.amazonaws.com:27017  
--sslCAFile rds-ca-2019-root.pem --username documentdb --password documentdb  
  
db.coll.getIndexes()
```

Step 4: Create a AWS DMS Replication Instance

An AWS DMS replication instance connects and reads data from your source (in this case an Amazon DocumentDB 3.6 cluster) and writes it to your target (Amazon DocumentDB 4.0 cluster). The AWS DMS replication instance can perform both bulk load and CDC operations. Most of this processing happens in memory. However, large operations might require some buffering on disk. Cached transactions and log files are also written to disk. Once the data is migrated, the replication instance also streams any change events to make sure the source and target are in sync.

To create an AWS DMS replication instance:

1. Open the AWS DMS [console](#).
2. In the navigation pane, choose **Replication instances**.
3. Choose **Create replication instance** and enter the following information:
 - For **Name**, enter a name of your choice. For example, docdb36todocdb40.
 - For **Description**, enter a description of your choice. For listitem, Amazon DocumentDB 3.6 to Amazon DocumentDB 4.0 replication instance.
 - For **Instance class**, choose the size based on your needs.
 - For **Engine version**, choose 3.4.1.
 - For **Amazon VPC**, choose the Amazon VPC that houses your Amazon DocumentDB 3.6 and 4.0 clusters.
 - For **Allocated storage (GiB)**, use the default of 50 GiB. If you have a high write throughput workload, increase this value to match your workload.
 - For **Multi-AZ**, choose Yes if you need high availability and failover support.
 - For **Publicly accessible**, enable this option.

Replication instance configuration

Name
The name must be unique among all of your replication instances in the current AWS region.

Replication instance name must not start with a numeric value.

Description
Amazon DocumentDB v3.6 to Amazon DocumentDB v4.0 replication
The description must only have unicode letters, digits, whitespace, or one of these symbols: _:/=-@. 1000 maximum character.

Instance class [Info](#)
Choose an appropriate instance class for your replication needs. Each instance class provides differing levels of compute, network and memory capacity. [DMS pricing](#)

16 vCPUs 30 GiB Memory

Include previous-generation instance classes

Engine version
Choose an AWS DMS version to run on your replication instance. [DMS versions](#)

Include Beta DMS versions

Allocated storage (GiB)
Choose the amount of storage space you want for your replication instance. AWS DMS uses this storage for log files and cached transactions while replication tasks are in progress.

VPC
Choose an Amazon Virtual Private Cloud (VPC) where your replication instance should run.

Multi AZ
If you choose this option, AWS DMS will perform a multi-AZ deployment, with a primary instance in one availability zone (AZ) and a standby instance in another AZ. This configuration provides a highly available, fault-tolerant replication environment. Billing is based on [DMS pricing](#)

Publicly accessible
If you choose this option, AWS DMS will assign a public IP address to your replication instance, and you'll be able to connect to databases outside of your Amazon VPC.

4. Choose **Create replication instance**.

Step 5: Create an AWS DMS Source Endpoint

The source endpoint is used for the Amazon DocumentDB 3.6 cluster you are looking to upgrade to 4.0.

To create a source endpoint

1. Open the AWS DMS [console](#).
2. In the navigation pane, choose **Endpoints**.
3. Choose **Create endpoint** and enter the following information:
 - For **Endpoint type**, choose **Source**.
 - For **Endpoint identifier**, enter a name that's easy to remember, for example docdb-source.
 - For **Source engine**, choose docdb.
 - For **Server name**, enter the DNS name of your Amazon DocumentDB v3.6 cluster.

- For **Port**, enter the port number of your Amazon DocumentDB v3.6 cluster.
- For **SSL mode**, choose **verify-full**.
- For **CA certificate**, choose **Add new CA certificate**. Download the [new CA certificate](#) to create TLS connections bundle. For **Certificate identifier**, enter `rds-combined-ca-bundle`. For **Import certificate file**, choose **Choose file** and navigate to the .pem file that you previously downloaded. Select and open the file. Choose **Import certificate**, then choose `rds-combined-ca-bundle` from the **Choose a certificate** drop down
- For **Username**, enter the master username of your Amazon DocumentDB v3.6 cluster.
- For **Password**, enter the master password of your Amazon DocumentDB v3.6 cluster.
- For **Database name**, enter the database name you are looking to upgrade.

Endpoint configuration

Endpoint identifier [Info](#)
A label for the endpoint to help you identify it.

Source engine
The type of database engine this endpoint is connected to.

Server name

Port The port the database runs on for this endpoint. <input type="text" value="27017"/>	Secure Socket Layer (SSL) mode The type of Secure Socket Layer enforcement <input type="text" value="verify-full"/>
---	--

CA certificate
 [Add new CA certificate](#)

User name Info <input type="text" value="docdbadmin"/>	Password Info <input type="text" value="*****"/>
--	--

Database name

4. Test your connection to verify it was successfully setup.

▼ Test endpoint connection (optional)

VPC

Replication instance
A replication instance performs the database migration

Run test

Endpoint identifier	Replication instance	Status	Message
docdb36-source	docdb36todocdb40	successful	

5. Choose **Create Endpoint**.

Note

AWS DMS can only migrate one database at a time.

Step 6: Create an AWS DMS Target Endpoint

The target endpoint is for your Amazon DocumentDB 4.0 cluster.

To create a target endpoint:

1. Open the [AWS DMS console](#).
2. In the navigation pane, choose **Endpoints**.
3. Choose **Create endpoint** and enter the following information:
 - For **Endpoint type**, choose **Target**.
 - For **Endpoint identifier**, enter a name that's easy to remember, for example docdb-target.
 - For **Source engine**, choose docdb.
 - For **Server name**, enter the DNS name of your Amazon DocumentDB v4.0 cluster.
 - For **Port**, enter the port number of your Amazon DocumentDB v4.0 cluster.
 - For **SSL mode**, choose **verify-full**.
 - For **CA certificate**, choose the existing `rds-combined-ca-bundle` certificate from the **Choose a certificate** drop down.
 - For **Username**, enter the master username of your Amazon DocumentDB v4.0 cluster.
 - For **Password**, enter the master password of your Amazon DocumentDB v4.0 cluster.
 - For **Database name**, enter the same database name you used to setup your source endpoint.

Endpoint configuration

Endpoint identifier [Info](#)
A label for the endpoint to help you identify it.

Target engine
The type of database engine this endpoint is connected to.

Server name

Port
The port the database runs on for this endpoint.

Secure Socket Layer (SSL) mode
The type of Secure Socket Layer enforcement

CA certificate
 [Add new CA certificate](#)

User name [Info](#)

Password [Info](#)

Database name

4. Test your connection to verify it was successfully set up.

Endpoint identifier	Replication instance	Status	Message
docdb36-target	docdb36todocdb40	successful	

5. Choose **Create Endpoint**.

Step 7: Create and run a migration task

An AWS DMS task binds the replication instance with your source and target instance. When you create a migration task, you specify the source endpoint, target endpoint, replication instance and any desired migration settings. An AWS DMS task can be created with three different migration types - migrate existing data, migrate existing data and replicate ongoing changes, or replicate data changes only. Since the purpose of this walk through is to upgrade an Amazon DocumentDB 3.6 cluster to Amazon DocumentDB 4.0 with minimal downtime, the steps utilize the option to migrate existing data and replicate ongoing changes. With this option, AWS DMS captures changes while migrating your existing data. AWS DMS continues to capture and apply changes even after the bulk data has been loaded. Eventually the source and target databases will be in sync, allowing for a minimal downtime migration.

Below are the steps to create a migration task for a minimal downtime migration:

1. Open the AWS DMS [console](#).
2. In the navigation pane, choose **Tasks**.
3. Choose **Create task** and enter the following information:
 - For **Task name**, enter a name that's easy to remember, for example my-dms-upgrade-task.
 - For **Replication instance**, choose the replication instance that you created in [Step 3: Create an AWS Database Migration Service Replication Instance](#)
 - For **Source endpoint**, choose the source endpoint that you created in [Step 4: Create an AWS Database Migration Service Source Endpoint](#)
 - For **Target endpoint**, choose the target endpoint that you created in [Step 5: Create an AWS Database Migration Service Target Endpoint](#)
 - For **Migration type**, choose **Migrate existing data and replicate ongoing changes**.

The screenshot shows the 'Task configuration' section of the AWS DMS console. It includes fields for 'Task identifier' (my-dms-upgrade-task), 'Replication instance' (docdb36todocdb40 - vpc-b06365ca), 'Source database endpoint' (docdb36-source), 'Target database endpoint' (docdb40-target), 'Migration type' (Info: Migrate existing data and replicate ongoing changes), and 'CloudWatch logs' (disabled).

4. In the **Task Settings** section, enable **CloudWatch logs**.
5. For **Table mappings** section, keep everything at its default setting. This will ensure all collections from your database are migrated.
6. For **Migration task startup** configuration, choose **Automatically on create**. This will start the migration task automatically once you create it.
7. Choose **Create task**.

AWS DMS now begins migrating data from your Amazon DocumentDB 3.6 cluster to your Amazon DocumentDB 4.0 cluster. The task status should change from Starting to Running. You can monitor the progress by choosing Tasks in the AWS DMS console. After several minutes/hours (depending on the size of your migration), the status should change from Load complete, replication ongoing. This means that AWS DMS has completed a full load migration of your Amazon DocumentDB 3.6 cluster to an Amazon DocumentDB 4.0 cluster and is now replicating change events.

Summary			
Status	Type	Source	Target
Load complete, replication ongoing	Full load, ongoing replication	docdb36source	docdb40target

Eventually your source and target will be in sync. You can verify whether they are in sync by running a count() operation on your collections to verify all change events have migrated.

Step 8: Changing the application endpoint to the Amazon DocumentDB cluster 4.0

After the full load is complete and the CDC process is replicating continuously, you are ready to change your application's database connection endpoint from your Amazon DocumentDB 3.6 cluster to your Amazon DocumentDB cluster 4.0 cluster.

Migration Tools

To migrate to Amazon DocumentDB, the two primary tools that most customers use are the [AWS Database Migration Service \(AWS DMS\)](#) and command line utilities like mongodump and mongorestore. As a best practice, and for either of these options, we recommend that you first create indexes in Amazon

DocumentDB before beginning your migration as it can reduce the overall time and increase the speed of the migration. To do this, you can use the [Amazon DocumentDB Index Tool](#).

AWS Database Migration Service

AWS Database Migration Service (AWS DMS) is a cloud service that makes it easy to migrate relational databases and non-relational databases to Amazon DocumentDB. You can use AWS DMS to migrate your data to Amazon DocumentDB from databases hosted on-premises or on EC2. With AWS DMS, you can perform one-time migrations, or you can replicate ongoing changes to keep sources and targets in sync.

For more information on using AWS DMS to migrate to Amazon DocumentDB, please see:

- [Using MongoDB as a Source for AWS DMS](#)
- [Using Amazon DocumentDB as a Target for AWS Database Migration Service](#)
- [Walkthrough: Migrating from MongoDB to Amazon DocumentDB](#)

Command Line Utilities

Common utilities for migrating data to and from Amazon DocumentDB include mongodump, mongorestore, mongoexport, and mongoimport. Typically, mongodump and mongorestore are the most efficient utilities as they dump and restore data from your databases in a binary format. This is generally the most performant option and yields a smaller data size compared to logical exports. mongoexport and mongoimport are useful if you want to export and import data in a logical format like JSON or CSV as the data is human readable but is generally slower than the mongodump/mongorestore and yields a larger data size.

The [Migration Approaches \(p. 134\)](#) section below will discuss when it is best to use AWS DMS and command line utilities based on your use case and requirements.

Discovery

For each of your MongoDB deployments, you should identify and record two sets of data: *Architecture Details* and *Operational Characteristics*. This information will help you choose the appropriate migration approach and cluster sizing.

Architecture Details

- **Name**

Choose a unique name for tracking this deployment.

- **Version**

Record the version of MongoDB that your deployment is running. To find the version, connect to a replica set member with the mongo shell and run the db.version() operation.

- **Type**

Record whether your deployment is a standalone mongo instance, a replica set, or a sharded cluster.

- **Members**

Record the hostnames, addresses, and ports of each cluster, replica set, or standalone member.

For a clustered deployment, you can find shard members by connecting to a mongo host with the mongo shell and running the `sh.status()` operation.

For a replica set, you can obtain the members by connecting to a replica set member with the mongo shell and running the `rs.status()` operation.

- **Opslog sizes**

For replica sets or sharded clusters, record the size of the opslog for each replica set member. To find a member's opslog size, connect to the replica set member with the mongo shell and run the `ps.printReplicationInfo()` operation.

- **Replica set member priorities**

For replica sets or sharded clusters, record the priority for each replica set member. To find the replica set member priorities, connect to a replica set member with the mongo shell and run the `rs.conf()` operation. The priority is shown as the value of the `priority` key.

- **TLS/SSL usage**

Record whether Transport Layer Security (TLS)/Secure Sockets Layer (SSL) is used on each node for encryption in transit.

Operational Characteristics

- **Database statistics**

For each collection, record the following information:

- Name
- Data size
- Collection count

To find the database statistics, connect to your database with the mongo shell and run the command `db.runCommand({dbstats: 1})`.

- **Collection statistics**

For each collection, record the following information:

- Namespace
- Data size
- Index count
- Whether the collection is capped

- **Index statistics**

For each collection, record the following index information:

- Namespace
- ID
- Size
- Keys
- TTL
- Sparse
- Background

To find the index information, connect to your database with the mongo shell and run the command `db.collection.getIndexes()`.

- **Opcounters**

This information helps you understand your current MongoDB workload patterns (read-heavy, write-heavy, or balanced). It also provides guidance on your initial Amazon DocumentDB instance selection.

The following are the key pieces of information to collect over the monitoring period (in counts/sec):

- Queries
- Inserts
- Updates
- Deletes

You can obtain this information by graphing the output of the `db.serverStatus()` command over time. You can also use the `mongostat` tool to obtain instantaneous values for these statistics. However, with this option you run the risk of planning your migration on usage periods other than your peak load.

- **Network statistics**

This information helps you understand your current MongoDB workload patterns (read-heavy, write-heavy, or balanced). It also provides guidance on your initial Amazon DocumentDB instance selection.

The following are the key pieces of information to collect over the monitoring period (in counts/sec):

- Connections
- Network bytes in
- Network bytes out

You can get this information by graphing the output of the `db.serverStatus()` command over time. You can also use the `mongostat` tool to obtain instantaneous values for these statistics. However, with this option you run the risk of planning your migration on usage periods other than your peak load.

Planning: Amazon DocumentDB Cluster Requirements

Successful migration requires that you carefully consider both your Amazon DocumentDB cluster's configuration and how applications will access your cluster. Consider each of the following dimensions when determining your cluster requirements:

- **Availability**

Amazon DocumentDB provides high availability through the deployment of replica instances, which can be promoted to a primary instance in a process known as *failover*. By deploying replica instances to different Availability Zones, you can achieve higher levels of availability.

The following table provides guidelines for Amazon DocumentDB deployment configurations to meet specific availability goals.

Availability Goal	Total Instances	Replicas	Availability Zones
99%	1	0	1
99.9%	2	1	2
99.99%	3	2	3

Overall system reliability must consider all components, not just the database. For best practices and recommendations for meeting overall system reliability needs, see the [AWS Well-Architected Reliability Pillar Whitepaper](#).

- **Performance**

Amazon DocumentDB instances allow you to read from and write to your cluster's storage volume. Cluster instances come in a number of types, with varying amounts of memory and vCPU, which affect your cluster's read and write performance. Using the information you gathered in the discovery phase, choose an instance type that can support your workload performance requirements. For a list of supported instance types, see [Managing Instance Classes \(p. 313\)](#).

When choosing an instance type for your Amazon DocumentDB cluster, consider the following aspects of your workload's performance requirements:

- **vCPUs**—Architectures that require higher connection counts might benefit from instances with more vCPUs.
- **Memory**—When possible, keeping your working dataset in memory provides maximum performance. A starting guideline is to reserve a third of your instance's memory for the Amazon DocumentDB engine, leaving two-thirds for your working dataset.
- **Connections**—The minimum optimal connection count is eight connections per Amazon DocumentDB instance vCPU. Although the Amazon DocumentDB instance connection limit is much higher, performance benefits of additional connections decline above eight connections per vCPU.
- **Network**—Workloads with a large number of clients or connections should consider the aggregate network performance required for inserted and retrieved data. Bulk operations can make more efficient use of network resources.
- **Insert Performance**—Single document inserts are generally the slowest way to insert data into Amazon DocumentDB. Bulk insert operations can be dramatically faster than single inserts.
- **Read Performance**—Reads from working memory are always faster than reads returned from the storage volume. Therefore, optimizing your instance memory size to retain your working set in memory is ideal.

In addition to serving reads from your primary instance, Amazon DocumentDB clusters are automatically configured as replica sets. You can then route read-only queries to read replicas by setting read preference in your MongoDB driver. You can scale read traffic by adding replicas, reducing the overall load on the primary instance.

It is possible to deploy Amazon DocumentDB replicas of different instance types in the same cluster. An example use case might be to stand up a replica with a larger instance type to serve temporary analytics traffic. If you deploy a mixed set of instance types, be sure to configure the failover priority for each instance. This helps ensure that a failover event always promotes a replica of sufficient size to handle your write load.

- **Recovery**

Amazon DocumentDB continuously backs up your data as it is written. It provides point-in-time recovery (PITR) capabilities within a configurable period of 1–35 days, known as the *backup retention period*. The default backup retention period is one day. Amazon DocumentDB also automatically creates daily snapshots of your storage volume, which are also retained for the configured backup retention period.

If you want to retain snapshots beyond the backup retention period, you can also initiate manual snapshots at any time using the AWS Management Console and AWS Command Line Interface (AWS CLI). For more information, see [Backing Up and Restoring in Amazon DocumentDB \(p. 215\)](#).

Consider the following as you plan your migration:

- Choose a backup retention period of 1–35 days that meets your recovery point objective (RPO).
- Decide if you require manual snapshots, and if so, at what interval.

Migration Approaches

There are three primary approaches for migrating your data to Amazon DocumentDB.

Note

Although you can create indexes at any time in Amazon DocumentDB, it is faster overall to create your indexes before importing large datasets. As a best practice, we recommend that for each of the approaches below, you first create your indexes in Amazon DocumentDB before performing the migration. To do this, you can use the [Amazon DocumentDB Index Tool](#).

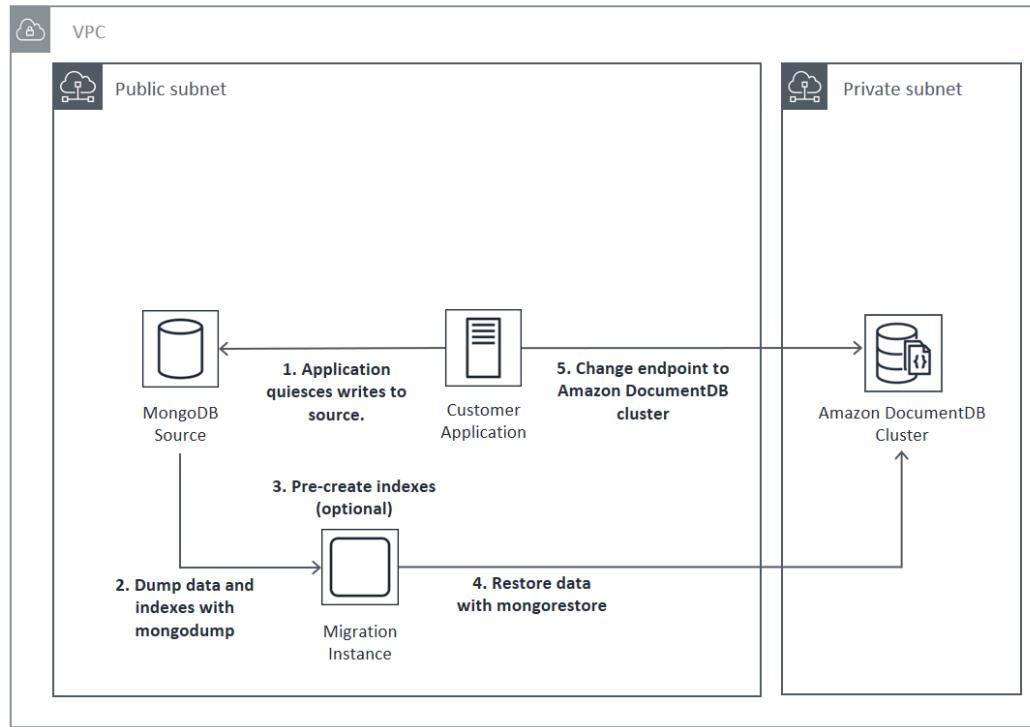
Offline

The *offline* approach uses the `mongodump` and `mongorestore` tools to migrate your data from your source MongoDB deployment to your Amazon DocumentDB cluster. The offline method is the simplest migration approach, but it also incurs the most downtime for your cluster.

The basic process for offline migration is as follows:

1. Quiesce writes to your MongoDB source.
2. Dump collection data and indexes from the source MongoDB deployment.
3. Restore indexes to the Amazon DocumentDB cluster.
4. Restore collection data to the Amazon DocumentDB cluster.
5. Change your application endpoint to write to the Amazon DocumentDB cluster.

Offline Migration Approach



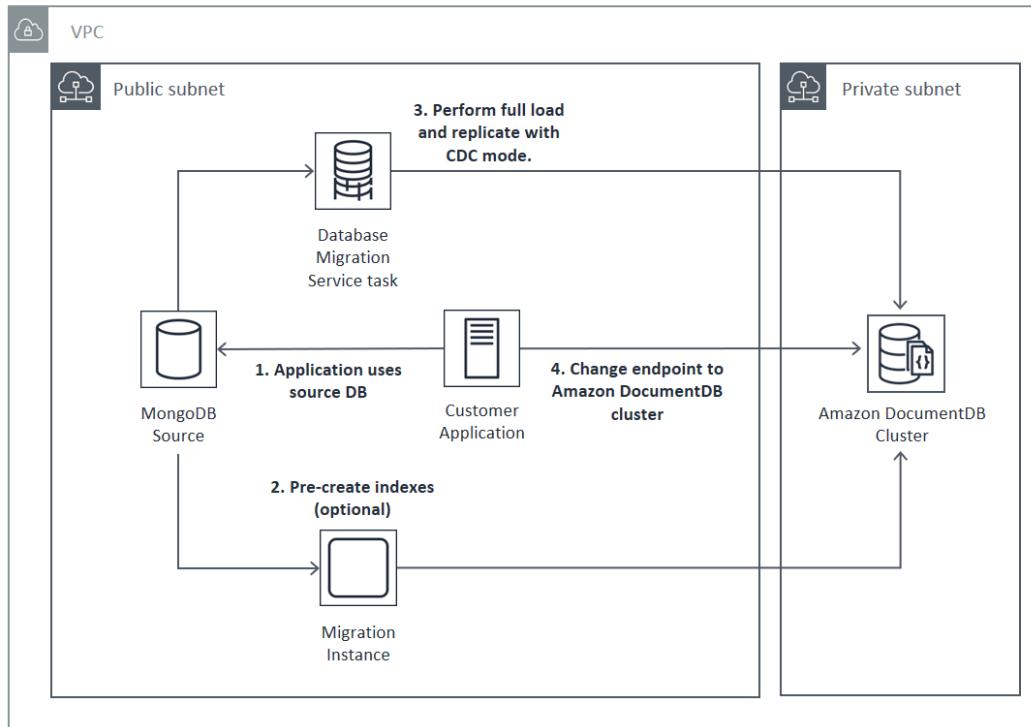
Online

The *online* approach uses AWS Database Migration Service (AWS DMS). It performs a full load of data from your source MongoDB deployment to your Amazon DocumentDB cluster. It then switches to change data capture (CDC) mode to replicate changes. The online approach minimizes downtime for your cluster, but it is the slowest of the three methods.

The basic process for online migration is as follows:

1. Your application uses the source DB normally.
2. Optionally, pre-create indexes in the Amazon DocumentDB cluster.
3. Create an AWS DMS task to perform a full load, and then enable CDC from the source MongoDB deployment to the Amazon DocumentDB cluster.
4. After the AWS DMS task has completed a full load and is replicating changes to the Amazon DocumentDB, switch the application's endpoint to the Amazon DocumentDB cluster.

Online Migration Approach



For more information about using AWS DMS to migrate, see [Using Amazon DocumentDB as a Target for AWS Database Migration Service](#) and the related [Tutorial](#) in the *AWS Database Migration Service User Guide*.

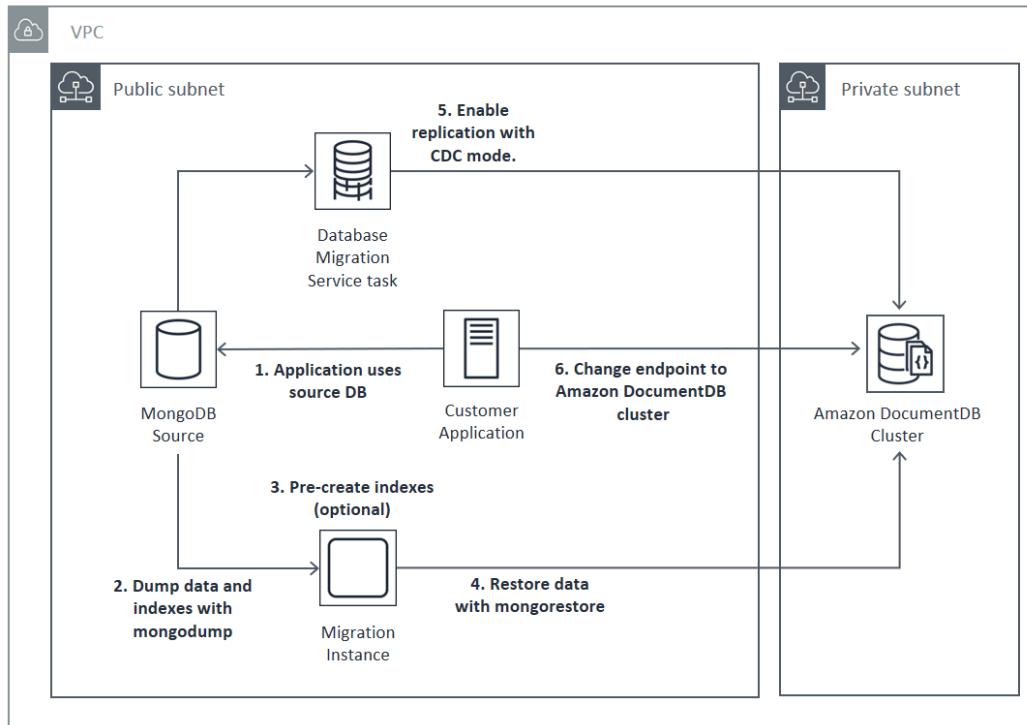
Hybrid

The *hybrid* approach uses the `mongodump` and `mongorestore` tools to migrate your data from your source MongoDB deployment to your Amazon DocumentDB cluster. It then uses AWS DMS in CDC mode to replicate changes. The hybrid approach balances migration speed and downtime, but it is the most complex of the three approaches.

The basic process for hybrid migration is as follows:

1. Your application uses the source MongoDB deployment normally.
2. Dump collection data and indexes from the source MongoDB deployment.
3. Restore indexes to the Amazon DocumentDB cluster.
4. Restore collection data to the Amazon DocumentDB cluster.
5. Create an AWS DMS task to enable CDC from the source MongoDB deployment to the Amazon DocumentDB cluster.
6. When the AWS DMS task is replicating changes within an acceptable window, change your application endpoint to write to the Amazon DocumentDB cluster.

Hybrid Migration Approach



Important

An AWS DMS task can currently migrate only a single database. If your MongoDB source has a large number of databases, you might need to automate the migration task creation, or consider using the offline method.

Regardless of the migration approach that you choose, it's most efficient to pre-create indexes in your Amazon DocumentDB cluster before migrating your data. This is because Amazon DocumentDB indexes are inserted data in parallel, but creating an index on existing data is a single-threaded operation.

Because AWS DMS does not migrate indexes (only your data), there is no extra step required to avoid creating indexes a second time.

Migration Sources

If your MongoDB source is a standalone mongo process and you want to use the online or hybrid migration approaches, first convert your standalone mongo to a replica set so that the oplog is created to use as a CDC source.

If you are migrating from a MongoDB replica set or sharded cluster, consider creating a chained or hidden secondary for each replica set or shard to use as your migration source. Performing data dumps can force working set data out of memory and impact performance on production instances. You can reduce this risk by migrating from a node not serving production data.

Migration Source Versions

If your source MongoDB database version is different from the compatibility version of your destination Amazon DocumentDB cluster, you might need to take other preparation steps to ensure a successful migration. The two most common requirements encountered are the need to upgrade the source

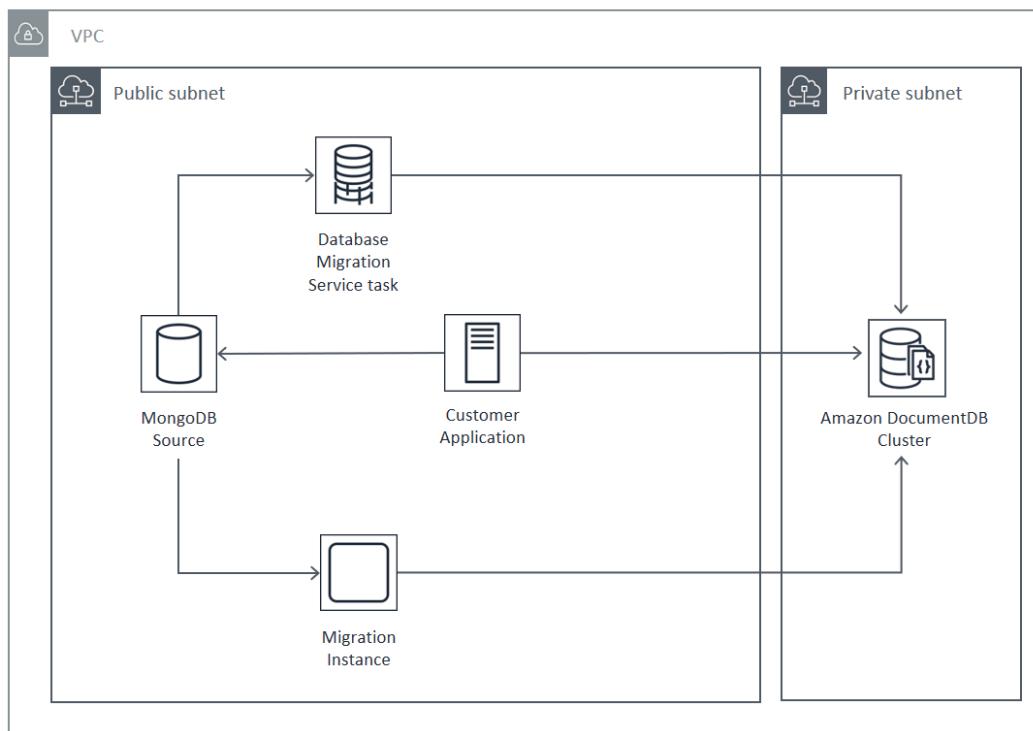
MongoDB installation to a supported version for migration (MongoDB version 3.0 or greater), and upgrading your application drivers to support the target Amazon DocumentDB version.

Ensure that if your migration has either of these requirements, you include those steps in your migration plan to upgrade and test any driver changes.

Migration Connectivity

You can migrate to Amazon DocumentDB from a source MongoDB deployment running in your data center or from a MongoDB deployment running on an Amazon EC2 instance. Migrating from MongoDB running on EC2 is straightforward, and only requires that you correctly configure your security groups and subnets.

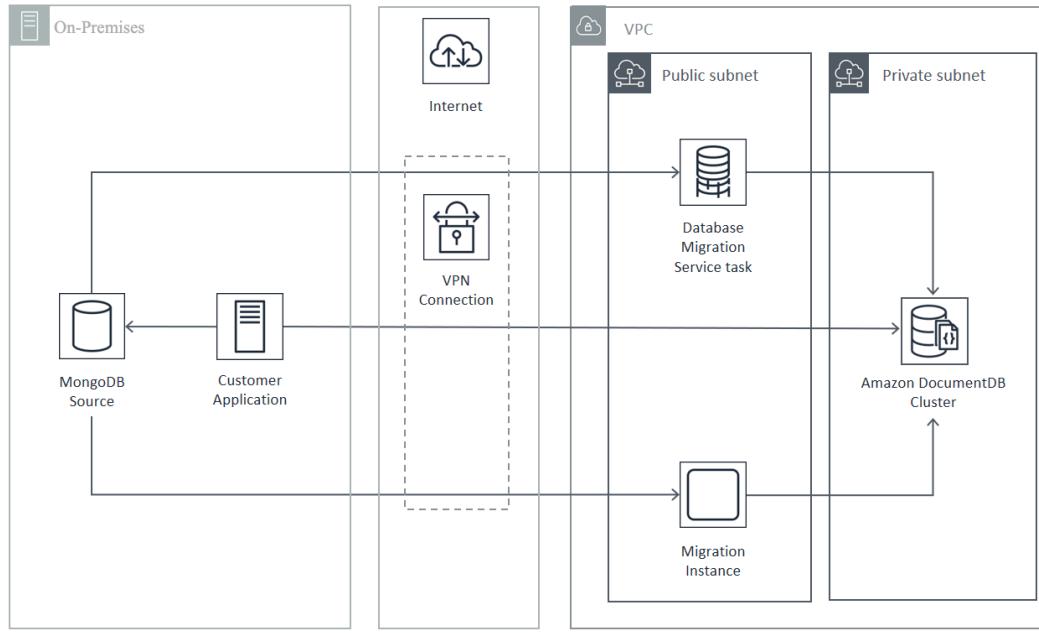
Migrating from EC2 Source



Migrating from an on-premises database requires connectivity between your MongoDB deployment and your virtual private cloud (VPC). You can accomplish this through a virtual private network (VPN) connection, or by using the AWS Direct Connect service. Although you can migrate over the internet to your VPC, this connection method is the least desirable from a security standpoint.

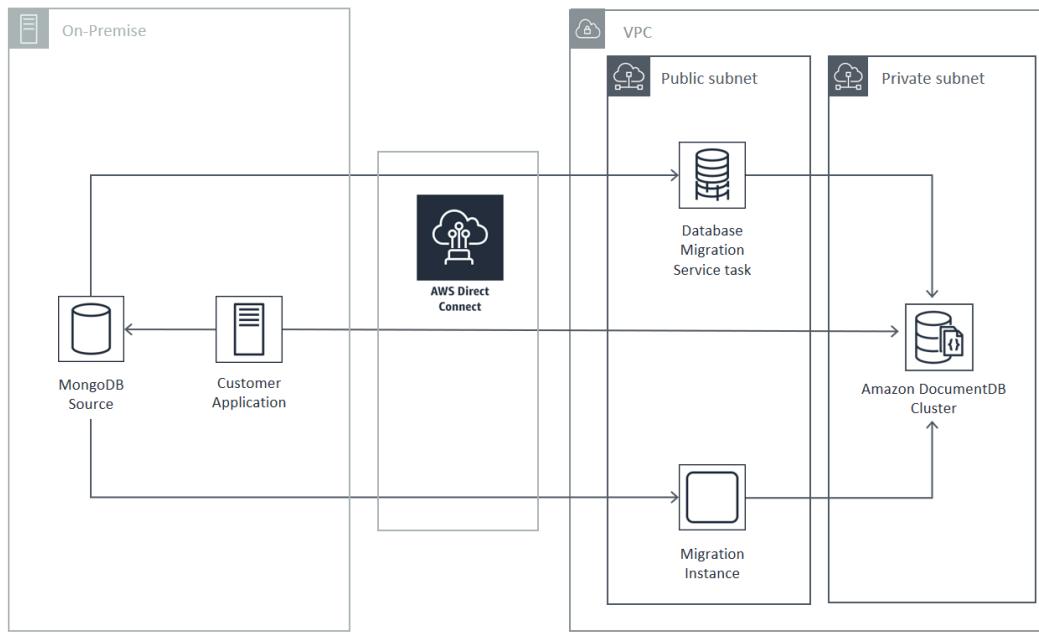
The following diagram illustrates a migration to Amazon DocumentDB from an on-premises source via a VPN connection.

Migrating from On-Premise Source (VPN)



The following represents a migration to Amazon DocumentDB from an on-premises source using AWS Direct Connect.

Migrating from On-Premise Source (Direct Connect)



Online and hybrid migration approaches require the use of an AWS DMS instance, which must run on Amazon EC2 in an Amazon VPC. All approaches require a migration server to run `mongodump` and `mongorestore`. It is generally easier to run the migration server on an Amazon EC2 instance in the VPC where your Amazon DocumentDB cluster is launched because it dramatically simplifies connectivity to your Amazon DocumentDB cluster.

Testing

The following are goals of pre-migration testing:

- Verify that your chosen approach achieves your desired migration outcome.
- Verify that your instance type and read preference choices meet your application performance requirements.
- Verify your application's behavior during failover.

Migration Plan Testing Considerations

Consider the following when testing your Amazon DocumentDB migration plan.

Topics

- [Restoring Indexes \(p. 140\)](#)
- [Dumping Data \(p. 140\)](#)
- [Restoring Data \(p. 140\)](#)
- [Oplog Sizing \(p. 141\)](#)
- [AWS Database Migration Service Configuration \(p. 141\)](#)
- [Migrating from a Sharded Cluster \(p. 141\)](#)

Restoring Indexes

By default, `mongorestore` creates indexes for dumped collections, but it creates them after the data is restored. It is faster overall to create indexes in Amazon DocumentDB before data is restored to the cluster. This is because the indexing operations are parallelized during the data load.

If you choose to pre-create your indexes, you can skip the index creation step when restoring data with `mongorestore` by supplying the `--noIndexRestore` option.

Dumping Data

The `mongodump` tool is the preferred method of dumping data from your source MongoDB deployment. Depending on the resources available on your migration instance, you might be able to speed up your `mongodump` by increasing the number of parallel connections dumped from the default 4 using the `--numParallelCollections` option.

Restoring Data

The `mongorestore` tool is the preferred method for restoring dumped data to your Amazon DocumentDB instance. You can improve restore performance by increasing the number of workers for each collection during restore with the `--numInsertionWorkersPerCollection` option. One worker per vCPU on your Amazon DocumentDB cluster primary instance is a good place to start.

Amazon DocumentDB does not currently support the `mongorestore` tool's `--oplogReplay` option.

By default, `mongorestore` skips insert errors and continues the restore process. This can occur if you are restoring unsupported data to your Amazon DocumentDB instance. For example, it can happen if you have a document that contains keys or values with null strings. If you prefer to have the `mongorestore` operation fail entirely if any restore error is encountered, use the `--stopOnError` option.

Opslog Sizing

The MongoDB operations log (opslog) is a capped collection that contains all data modifications to your database. You can view the size of the opslog and the time range it contains by running the `db.printReplicationInfo()` operation on a replica set or shard member.

If you are using the online or hybrid approaches, ensure that the opslog on each replica set or shard is large enough to contain all changes made during the entire duration of the data migration process (whether via mongodump or an AWS DMS task full load), plus a reasonable buffer. For more information, see *Check the Size of the Opslog* in the MongoDB documentation. Determine the minimum required opslog size by recording the elapsed time taken by the first test run of your mongodump or mongorestore process or AWS DMS full load task.

AWS Database Migration Service Configuration

The [AWS Database Migration Service User Guide](#) covers the components and steps required to migrate your MongoDB source data to your Amazon DocumentDB cluster. The following is the basic process for using AWS DMS to perform an online or hybrid migration:

To perform a migration using AWS DMS

1. Create a MongoDB source endpoint. For more information, see [Using MongoDB as a Source for AWS DMS](#).
2. Create an Amazon DocumentDB target endpoint. For more information, see [Working with AWS DMS Endpoints](#).
3. Create at least one AWS DMS replication instance. For more information, see [Working with an AWS DMS Replication Instance](#).
4. Create at least one AWS DMS replication task. For more information, see [Working with AWS DMS Tasks](#).

For an online migration, your migration task uses the migration type **Migrate existing data and replicate ongoing changes**.

For a hybrid migration, your migration task uses the migration type **Replicate data changes only**. You can choose the CDC start time to align with your dump time from your mongodump operation. The MongoDB opslog is idempotent. To avoid missing changes, it's a good idea to leave a few minutes worth of overlap between your mongodump finish time and your CDC start time.

Migrating from a Sharded Cluster

The process for migrating data from a sharded cluster to your Amazon DocumentDB instance is essentially that of several replica set migrations in parallel. A key consideration when testing a sharded cluster migration is that some shards might be more heavily used than others. This situation leads to varying elapsed times for data migration. Ensure that you evaluate each shard's opslog requirements when planning and testing.

The following are some configuration issues to consider when migrating a sharded cluster:

- Before running mongodump or starting an AWS DMS migration task, you must disable the sharded cluster balancer and wait for any in-process migrations to complete. For more information, see *Disable the Balancer* in the MongoDB documentation.
- If you are using AWS DMS to replicate data, run the `cleanupOrphaned` command on each shard before running the migration tasks. If you don't run this command, the tasks might fail because of duplicate document IDs. Note that this command might affect performance. For more information, see `cleanupOrphaned` in the MongoDB documentation.

- If you are using the mongodump tool to dump data, you should run one mongodump process per shard. The most time-efficient approach might require multiple migration servers to maximize your dump performance.
- If you are using AWS Database Migration Service to replicate data, you must create a source endpoint for each shard. Also run at least one migration task for each shard that you are migrating. The most time-efficient approach might require multiple replication instances to maximize your migration performance.

Performance Testing

After you successfully migrate your data to your test Amazon DocumentDB cluster, execute your test workload against the cluster. Verify through Amazon CloudWatch metrics that your performance meets or exceeds your MongoDB source deployment's current throughput.

Verify the following key Amazon DocumentDB metrics:

- Network throughput
- Write throughput
- Read throughput
- Replica lag

For more information, see [Monitoring Amazon DocumentDB \(p. 401\)](#).

Failover Testing

Verify that your application's behavior during an Amazon DocumentDB failover event meets your availability requirements. To initiate a manual failover of an Amazon DocumentDB cluster on the console, on the **Clusters** page, choose the **Failover** action on the **Actions** menu.

You can also initiate a failover by executing the `failover-db-cluster` operation from the AWS CLI. For more information, see [failover-db-cluster](#) in the Amazon DocumentDB section of the AWS CLI reference.

Additional Resources

See the following topics in the *AWS Database Migration Service User Guide*:

- [Using Amazon DocumentDB as a Target for AWS Database Migration Service](#)
- [Walkthrough: Migrating from MongoDB to Amazon DocumentDB](#)

Security in Amazon DocumentDB

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. This documentation helps you understand how to apply the shared responsibility model when using Amazon DocumentDB. The [shared responsibility model](#) describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** — AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the [AWS compliance programs](#). To learn about the compliance programs that apply to Amazon DocumentDB (with MongoDB compatibility), see [AWS Services in Scope by Compliance Program](#).
- **Security in the cloud** — Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your organization's requirements, and applicable laws and regulations.

You also learn how to use other AWS services that help you monitor and secure your Amazon DocumentDB resources. The following topics show you how to configure Amazon DocumentDB to meet your security and compliance objectives.

Topics

- [Data Protection in Amazon DocumentDB \(p. 143\)](#)
- [Identity and Access Management in Amazon DocumentDB \(p. 153\)](#)
- [Managing Amazon DocumentDB Users \(p. 166\)](#)
- [Restricting Database Access Using Role-Based Access Control \(p. 169\)](#)
- [Logging and Monitoring in Amazon DocumentDB \(p. 186\)](#)
- [Updating Your Amazon DocumentDB TLS Certificates \(p. 187\)](#)
- [Updating Your Amazon DocumentDB TLS Certificates — GovCloud \(US-West\) \(p. 195\)](#)
- [Compliance Validation in Amazon DocumentDB \(p. 202\)](#)
- [Resilience in Amazon DocumentDB \(p. 203\)](#)
- [Infrastructure Security in Amazon DocumentDB \(p. 204\)](#)
- [Security Best Practices for Amazon DocumentDB \(p. 204\)](#)
- [Auditing Amazon DocumentDB Events \(p. 205\)](#)

Data Protection in Amazon DocumentDB

The AWS [shared responsibility model](#) applies to data protection in . As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the [Data Privacy FAQ](#). For information about data protection in Europe, see the [AWS Shared Responsibility Model and GDPR](#) blog post on the [AWS Security Blog](#).

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM). That way each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We recommend TLS 1.2 or later.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.
- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see [Federal Information Processing Standard \(FIPS\) 140-2](#).

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form fields such as a **Name** field. This includes when you work with Amazon DocumentDB or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Topics

- [Encrypting Amazon DocumentDB Data at Rest \(p. 144\)](#)
- [Encrypting Data in Transit \(p. 147\)](#)
- [Key Management \(p. 153\)](#)

Encrypting Amazon DocumentDB Data at Rest

Note

AWS KMS is replacing the term *customer master key (CMK)* with *AWS KMS key* and *KMS key*. The concept has not changed. To prevent breaking changes, AWS KMS is keeping some variations of this term.

You encrypt data at rest in your Amazon DocumentDB cluster by specifying the storage encryption option when you create your cluster. Storage encryption is enabled cluster-wide and is applied to all instances, including the primary instance and any replicas. It is also applied to your cluster's storage volume, data, indexes, logs, automated backups, and snapshots.

Amazon DocumentDB uses the 256-bit Advanced Encryption Standard (AES-256) to encrypt your data using encryption keys stored in AWS Key Management Service (AWS KMS). When using an Amazon DocumentDB cluster with encryption at rest enabled, you don't need to modify your application logic or client connection. Amazon DocumentDB handles encryption and decryption of your data transparently, with minimal impact on performance.

Amazon DocumentDB integrates with AWS KMS and uses a method known as envelope encryption to protect your data. When an Amazon DocumentDB cluster is encrypted with an AWS KMS, Amazon DocumentDB asks AWS KMS to use your KMS key to [generate a ciphertext data key](#) to encrypt the storage volume. The ciphertext data key is encrypted using the KMS key that you define, and is stored along with the encrypted data and storage metadata. When Amazon DocumentDB needs to access your encrypted data, it requests AWS KMS to decrypt the ciphertext data key using your KMS key and caches the plaintext data key in memory to efficiently encrypt and decrypt data in the storage volume.

The storage encryption facility in Amazon DocumentDB is available for all supported instance sizes and in all AWS Regions where Amazon DocumentDB is available.

Enabling Encryption at Rest for an Amazon DocumentDB Cluster

You can enable or disable encryption at rest on an Amazon DocumentDB cluster when the cluster is provisioned using either the AWS Management Console or the AWS Command Line Interface (AWS CLI).

Clusters that you create using the console have encryption at rest enabled by default. Clusters that you create using the AWS CLI have encryption at rest disabled by default. Therefore, you must explicitly enable encryption at rest using the `--storage-encrypted` parameter. In either case, after the cluster is created, you can't change the encryption at rest option.

Amazon DocumentDB uses AWS KMS to retrieve and manage encryption keys, and to define the policies that control how these keys can be used. If you don't specify an AWS KMS key identifier, Amazon DocumentDB uses the default AWS managed service KMS key. Amazon DocumentDB creates a separate KMS key for each AWS Region in your AWS account. For more information, see [AWS Key Management Service Concepts](#).

To get started on creating your own KMS key, see [Getting Started](#) in the *AWS Key Management Service Developer Guide*.

Important

You must use a symmetric encryption KMS key to encrypt your cluster as Amazon DocumentDB supports only symmetric encryption KMS keys. Do not use an asymmetric KMS key to attempt to encrypt the data in your Amazon DocumentDB clusters. For more information, see [Asymmetric keys in AWS KMS](#) in the *AWS Key Management Service Developer Guide*.

If Amazon DocumentDB can no longer gain access to the encryption key for a cluster — for example, when access to a key is revoked — the encrypted cluster goes into a terminal state. In this case, you can only restore the cluster from a backup. For Amazon DocumentDB, backups are always enabled for 1 day.

In addition, if you disable the key for an encrypted Amazon DocumentDB cluster, you will eventually lose read and write access to that cluster. When Amazon DocumentDB encounters a cluster that is encrypted by a key that it doesn't have access to, it puts the cluster into a terminal state. In this state, the cluster is no longer available, and the current state of the database can't be recovered. To restore the cluster, you must re-enable access to the encryption key for Amazon DocumentDB, and then restore the cluster from a backup.

Important

You cannot change the KMS key for an encrypted cluster after you have already created it. Be sure to determine your encryption key requirements before you create your encrypted cluster.

Using the AWS Management Console

You specify the encryption at rest option when you create a cluster. Encryption at rest is enabled by default when you create a cluster using the AWS Management Console. It can't be changed after the cluster is created.

To specify the encryption at rest option when creating your cluster

1. Create an Amazon DocumentDB cluster as described in the [Getting Started](#) section. However, in step 6, do not choose **Create cluster**.
2. Under the **Authentication** section, choose **Show advanced settings**.
3. Scroll down to the **Encryption-at-rest** section.
4. Choose the option that you want for encryption at rest. Whichever option you choose, you can't change it after the cluster is created.
 - To encrypt data at rest in this cluster, choose **Enable encryption**.
 - If you don't want to encrypt data at rest in this cluster, choose **Disable encryption**.
5. Choose the master key that you want. Amazon DocumentDB uses the AWS Key Management Service (AWS KMS) to retrieve and manage encryption keys, and to define the policies that control how these keys can be used. If you don't specify an AWS KMS key identifier, Amazon DocumentDB uses the default AWS managed service KMS key. For more information, see [AWS Key Management Service Concepts](#).

Note

After you create an encrypted cluster, you can't change the KMS key for that cluster. Be sure to determine your encryption key requirements before you create your encrypted cluster.

6. Complete the other sections as needed, and create your cluster.

Using the AWS CLI

To encrypt an Amazon DocumentDB cluster using the AWS CLI, you must specify the `--storage-encrypted` option when creating the cluster. Amazon DocumentDB clusters created using the AWS CLI do not enable storage encryption by default.

The following example creates an Amazon DocumentDB cluster with storage encryption enabled.

Example

For Linux, macOS, or Unix:

```
aws docdb create-db-cluster \
    --db-cluster-identifier sample-cluster \
    --port 27017 \
    --engine docdb \
    --master-username yourMasterUsername \
    --master-user-password yourMasterPassword \
    --storage-encrypted
```

For Windows:

```
aws docdb create-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --port 27017 ^
    --engine docdb ^
    --master-username yourMasterUsername ^
    --master-user-password yourMasterPassword ^
    --storage-encrypted
```

When you create an encrypted Amazon DocumentDB cluster, you can specify an AWS KMS key identifier, as in the following example.

Example

For Linux, macOS, or Unix:

```
aws docdb create-db-cluster \
    --db-cluster-identifier sample-cluster \
    --port 27017 \
    --engine docdb \
    --master-username yourMasterUsername \
    --master-user-password yourMasterPassword \
    --storage-encrypted \
    --kms-key-id key-arn-or-alias
```

For Windows:

```
aws docdb create-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --port 27017 ^
    --engine docdb ^
    --master-username yourMasterUsername ^
```

```
--master-user-password yourMasterPassword ^
--storage-encrypted ^
--kms-key-id key-arn-or-alias
```

Note

After you create an encrypted cluster, you can't change the KMS key for that cluster. Be sure to determine your encryption key requirements before you create your encrypted cluster.

Limitations for Amazon DocumentDB Encrypted Clusters

The following limitations exist for Amazon DocumentDB encrypted clusters.

- You can enable or disable encryption at rest for an Amazon DocumentDB cluster only at the time that it is created, not after the cluster has been created. However, you can create an encrypted copy of an unencrypted cluster by creating a snapshot of the unencrypted cluster, and then restoring the unencrypted snapshot as a new cluster while specifying the encryption at rest option.

For more information, see the following topics:

- [Creating a Manual Cluster Snapshot \(p. 224\)](#)
- [Restoring from a Cluster Snapshot \(p. 236\)](#)
- [Copying Amazon DocumentDB Cluster Snapshots \(p. 226\)](#)
- Amazon DocumentDB clusters with storage encryption enabled can't be modified to disable encryption.
- All instances, automated backups, snapshots, and indexes in an Amazon DocumentDB cluster are encrypted with the same KMS key.

Encrypting Data in Transit

You can use Transport Layer Security (TLS) to encrypt the connection between your application and an Amazon DocumentDB cluster. By default, encryption in transit is enabled for newly created Amazon DocumentDB clusters. It can optionally be disabled when the cluster is created, or at a later time. When encryption in transit is enabled, secure connections using TLS are required to connect to the cluster. For more information connecting to Amazon DocumentDB using TLS, see [Connecting Programmatically to Amazon DocumentDB \(p. 469\)](#).

Managing Amazon DocumentDB Cluster TLS Settings

Encryption in transit for an Amazon DocumentDB cluster is managed via the TLS parameter in a [cluster parameter group](#). You can manage your Amazon DocumentDB cluster TLS settings using the AWS Management Console or the AWS Command Line Interface (AWS CLI). See the following sections to learn how to verify and modify your current TLS settings.

Using the AWS Management Console

Follow these steps to perform management tasks for TLS encryption using the console—such as identifying parameter groups, verifying the TLS value, and making needed modifications.

Note

Unless you specify differently when you create a cluster, your cluster is created with the default cluster parameter group. The parameters in the default cluster parameter group can't be modified (for example, tls enabled/disabled). So if your cluster is using a default cluster parameter group, you need to modify the cluster to use a non-default cluster parameter group. First, you might need to create a custom cluster parameter group. For more information, see [Creating Amazon DocumentDB Cluster Parameter Groups \(p. 360\)](#).

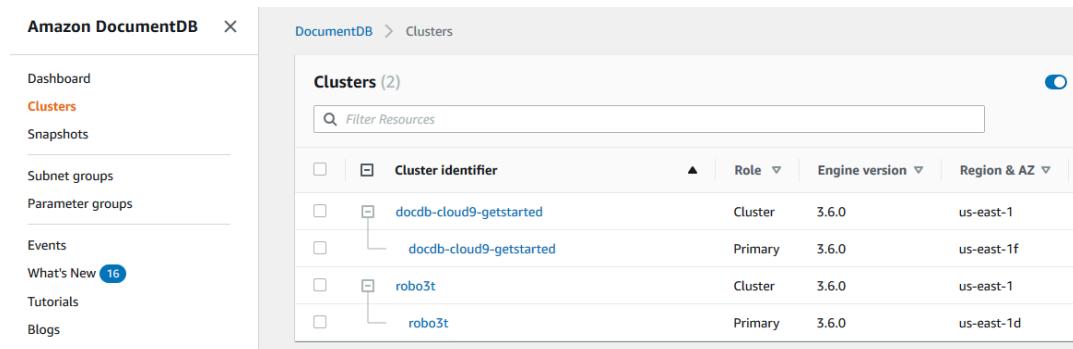
1. **Determine the cluster parameter group that your cluster is using.**

- a. Open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
- b. In the navigation pane, choose **Clusters**.

Tip

If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.

- c. Note that in the Clusters navigation box, the column **Cluster Identifier** shows both clusters and instances. Instances are listed underneath clusters. See the screenshot below for reference.



The screenshot shows the Amazon DocumentDB Clusters page. On the left, the navigation pane includes links for Dashboard, Clusters (which is selected and highlighted in orange), Snapshots, Subnet groups, Parameter groups, Events, What's New (with 16 items), Tutorials, and Blogs. The main content area is titled "Clusters (2)". It features a search bar labeled "Filter Resources". Below the search bar is a table with columns: Cluster identifier, Role, Engine version, and Region & AZ. The table contains four rows:

Cluster identifier	Role	Engine version	Region & AZ
docdb-cloud9-getstarted	Cluster	3.6.0	us-east-1
docdb-cloud9-getstarted	Primary	3.6.0	us-east-1f
robo3t	Cluster	3.6.0	us-east-1
robo3t	Primary	3.6.0	us-east-1d

- d. Choose the cluster that you're interested in.
- e. Scroll down to the bottom of **Cluster details** and locate the **Cluster parameter group**. Note the name of the cluster parameter group.

If the name of the cluster's parameter group is **default** (for example, **default.docdb3.6**), you must create a custom cluster parameter group and make it the cluster's parameter group before you continue. For more information, see the following:

1. [Creating Amazon DocumentDB Cluster Parameter Groups \(p. 360\)](#) — If you don't have a custom cluster parameter group that you can use, create one.
2. [Modifying an Amazon DocumentDB Cluster \(p. 287\)](#) — Modify your cluster to use the custom cluster parameter group.

2. Determine the current value of the `tls` cluster parameter.

- a. Open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
- b. In the navigation pane, choose **Parameter groups**.
- c. In the list of cluster parameter groups, choose the name of the cluster parameter group you are interested in.
- d. Locate the **Cluster parameters** section. In the list of cluster parameters, locate the `tls` cluster parameter row. At this point, the following four columns are important:
 - **Cluster parameter name** — The name of the cluster parameters. For managing TLS, you're interested in the `tls` cluster parameter.
 - **Values** — The current value of each cluster parameter.
 - **Allowed values** — A list of values that can be applied to a cluster parameter.
 - **Apply type** — Either **static** or **dynamic**. Changes to static cluster parameters can be applied only when the instances are rebooted. Changes to dynamic cluster parameters can be applied either immediately or when the instances are rebooted.

3. Modify the value of the `tls` cluster parameter.

If the value of `tls` is not what is needs to be, modify its value for this cluster parameter group. To change the value of the `tls` cluster parameter, continue from the preceding section by following these steps.

- a. Choose the button to the left of the cluster parameter's name (`tls`).
 - b. Choose **Edit**.
 - c. To change the value of `tls`, in the **Modify `tls`** dialog box, choose the value that you want for the cluster parameter in the drop-down list.
 - d. Choose **Modify cluster parameter**. The change is applied to each cluster instance when it is rebooted.
4. **Reboot the Amazon DocumentDB instance.**

Reboot each instance of the cluster so that the change is applied to all instances in the cluster.

 - a. Open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
 - b. In the navigation pane, choose **Instances**.
 - c. To specify an instance to reboot, locate the instance in the list of instances, and choose the button to the left of its name.
 - d. Choose **Actions**, and then **Reboot**. Confirm that you want to reboot by choosing **Reboot**.

Using the AWS CLI

Follow these steps to perform management tasks for TLS encryption using the AWS CLI—such as identifying parameter groups, verifying the TLS value, and making needed modifications.

Note

Unless you specify differently when you create a cluster, the cluster is created with the default cluster parameter group. The parameters in the default cluster parameter group can't be modified (for example, `tls` enabled/disabled). So if your cluster is using a default cluster parameter group, you need to modify the cluster to use a non-default cluster parameter group. You might need to first create a custom cluster parameter group. For more information, see [Creating Amazon DocumentDB Cluster Parameter Groups \(p. 360\)](#).

1. **Determine the cluster parameter group that your cluster is using.**

Use the `describe-db-clusters` command with the following parameters:

- **--db-cluster-identifier** — Required. The name of the cluster of interest.
- **--query** — Optional. A query that limits the output to just the fields of interest, in this case, the cluster name and its cluster parameter group name.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier docdb-2019-05-07-13-57-08 \
    --query 'DBClusters[*].[DBClusterIdentifier,DBClusterParameterGroup]'
```

Output from this operation looks something like the following (JSON format).

```
[  
  [  
    "docdb-2019-05-07-13-57-08",  
    "custom3-6-param-grp"  
  ]  
]
```

If the name of the cluster's parameter group is default (for example, default.docdb3.6), you must have a custom cluster parameter group and make it the cluster's parameter group before you continue. For more information, see the following topics:

1. [Creating Amazon DocumentDB Cluster Parameter Groups \(p. 360\)](#) — If you don't have a custom cluster parameter group that you can use, create one.
2. [Modifying an Amazon DocumentDB Cluster \(p. 287\)](#) — Modify your cluster to use the custom cluster parameter group.

2. Determine the current value of the `tls` cluster parameter.

To get more information about this cluster parameter group, use the `describe-db-cluster-parameters` operation with the following parameters:

- **--db-cluster-parameter-group-name** — Required. Use the cluster parameter group name from the output of the previous command.
- **--query** — Optional. A query that limits the output to just the fields of interest, in this case, the `ParameterName`, `ParameterValue`, `AllowedValues`, and `ApplyType`.

```
aws docdb describe-db-cluster-parameters \
  --db-cluster-parameter-group-name custom3-6-param-grp \
  --query 'Parameters[*].[ParameterName,ParameterValue,AllowedValues,ApplyType]'
```

Output from this operation looks something like the following (JSON format).

```
[  
  [  
    "audit_logs",  
    "disabled",  
    "enabled,disabled",  
    "dynamic"  
  ],  
  [  
    "tls",  
    "disabled",  
    "disabled,enabled",  
    "static"  
  ],  
  [  
    "ttl_monitor",  
    "enabled",  
    "disabled,enabled",  
    "dynamic"  
  ]  
]
```

3. Modify the value of the `tls` cluster parameter.

If the value of `tls` is not what it needs to be, modify its value for this cluster parameter group. To change the value of the `tls` cluster parameter, use the `modify-db-cluster-parameter-group` operation with the following parameters.

- **--db-cluster-parameter-group-name** — Required. The name of the cluster parameter group to modify. This cannot be a `default.*` cluster parameter group.
- **--parameters** — Required. A list of the cluster parameter group's parameters to modify.
 - **ParameterName** — Required. The name of the cluster parameter to modify.

- **ParameterValue** — Required. The new value for this cluster parameter. Must be one of the cluster parameter's AllowedValues.
 - **enabled** — The cluster only accepts secure connections using TLS.
 - **disabled** — The cluster does not accept secure connections using TLS.
- **ApplyMethod** — When this modification is to be applied. For static cluster parameters like `tls`, this value must be `pending-reboot`.
 - **pending-reboot** — Change is applied to an instance only after it is rebooted. You must reboot each cluster instance individually for this change to take place across all of the cluster's instances.

The following code *disables* `tls`, applying the change to each DB instance when it is rebooted.

```
aws docdb modify-db-cluster-parameter-group \
--db-cluster-parameter-group-name custom3-6-param-grp \
--parameters "ParameterName=tls,ParameterValue=disabled,ApplyMethod=pending-reboot"
```

The following code *enables* `tls`, applying the change to each DB instance when it is rebooted.

```
aws docdb modify-db-cluster-parameter-group \
--db-cluster-parameter-group-name custom3-6-param-grp \
--parameters "ParameterName=tls,ParameterValue=enabled,ApplyMethod=pending-reboot"
```

Output from this operation looks something like the following (JSON format).

```
{  
    "DBClusterParameterGroupName": "custom3-6-param-grp"  
}
```

4. Reboot your Amazon DocumentDB instance.

Reboot each instance of the cluster so that the change is applied to all instances in the cluster. To reboot an Amazon DocumentDB instance, use the `reboot-db-instance` operation with the following parameter:

- **--db-instance-identifier** — Required. The identifier for the instance to be rebooted.

The following code reboots the instance `sample-db-instance`.

Example

For Linux, macOS, or Unix:

```
aws docdb reboot-db-instance \
--db-instance-identifier sample-db-instance
```

For Windows:

```
aws docdb reboot-db-instance ^
--db-instance-identifier sample-db-instance
```

Output from this operation looks something like the following (JSON format).

```
{
    "DBInstance": {
        "AutoMinorVersionUpgrade": true,
        "PubliclyAccessible": false,
        "PreferredMaintenanceWindow": "fri:09:32-fri:10:02",
        "PendingModifiedValues": {},
        "DBInstanceState": "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": "sample-cluster-instance-00",
"Engine": "docdb",
"AvailabilityZone": "us-east-1a",
"DBInstanceArn": "arn:aws:rds:us-east-1:012345678901:db:sample-cluster-
instance-00",
"BackupRetentionPeriod": 1,
"Endpoint": {
    "Address": "sample-cluster-instance-00.corcjozrlsf.us-
east-1.docdb.amazonaws.com",
    "Port": 27017,
    "HostedZoneId": "Z2R2ITUGPM61AM"
},
"DBClusterIdentifier": "sample-cluster"
}
```

It takes a few minutes for your instance to reboot. You can use the instance only when its status is *available*. You can monitor the instance's status using the console or AWS CLI. For more information, see [Monitoring an Amazon DocumentDB Instance's Status \(p. 404\)](#).

Key Management

Amazon DocumentDB uses AWS Key Management Service (AWS KMS) to retrieve and manage encryption keys. AWS KMS combines secure, highly available hardware and software to provide a key management system scaled for the cloud. Using AWS KMS, you can create encryption keys and define the policies that control how these keys can be used. AWS KMS supports AWS CloudTrail, so you can audit key usage to verify that keys are being used appropriately.

Your AWS KMS keys can be used in combination with Amazon DocumentDB and supported AWS services such as Amazon Simple Storage Service (Amazon S3), Amazon Relational Database Service (Amazon RDS), Amazon Elastic Block Store (Amazon EBS), and Amazon Redshift. For a list of services that support AWS KMS, see [How AWS Services use AWS KMS](#) in the *AWS Key Management Service Developer Guide*. For information about AWS KMS, see [What is AWS Key Management Service?](#)

Identity and Access Management in Amazon DocumentDB

Access to manage Amazon DocumentDB (with MongoDB compatibility) resources such as clusters, instances, and cluster parameter groups requires credentials that AWS can use to authenticate your requests. Those credentials must have permissions to access AWS resources, such as an Amazon DocumentDB instance. The following sections provide details on how you can use [AWS Identity and Access Management \(IAM\)](#) and Amazon DocumentDB to help secure your resources by controlling who can access them.

Topics

- [Authentication \(p. 154\)](#)
- [Overview of Managing Access Permissions to Your Amazon DocumentDB Resources \(p. 155\)](#)

- [Managing Access Using Policies \(p. 158\)](#)
- [Using Identity-Based Policies \(IAM Policies\) for Amazon DocumentDB \(p. 158\)](#)
- [Amazon DocumentDB API Permissions: Actions, Resources, and Conditions Reference \(p. 161\)](#)

Authentication

You can access AWS as any of the following types of identities:

- **AWS account root user**

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see [Tasks that require root user credentials](#) in the *AWS General Reference*.

- **IAM users and groups**

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see [Rotate access keys regularly for use cases that require long-term credentials](#) in the *IAM User Guide*.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see [When to create an IAM user \(instead of a role\)](#) in the *IAM User Guide*.

- **IAM role**

An *IAM role* is an IAM identity that you can create in your account that has specific permissions. An IAM role is similar to an IAM user in that it is an AWS identity with permissions policies that determine what the identity can and cannot do in AWS. However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it. Also, a role does not have standard long-term credentials such as a password or access keys associated with it. Instead, when you assume a role, it provides you with temporary security credentials for your role session. IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see [Creating a role for a third-party Identity Provider](#) in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see [Permission sets](#) in the *AWS IAM Identity Center (successor to AWS Single Sign-On) User Guide*.
- **AWS service access** – A service role is an *IAM role* that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see [Creating a role to delegate permissions to an AWS service](#) in the *IAM User Guide*.

- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see [Using an IAM role to grant permissions to applications running on Amazon EC2 instances](#) in the *IAM User Guide*.

Overview of Managing Access Permissions to Your Amazon DocumentDB Resources

Every AWS resource is owned by an AWS account, and permissions to create or access the resources are governed by permissions policies. An account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles), and some services (such as AWS Lambda) also support attaching permissions policies to resources.

Note

An *account administrator* (or administrator user) is a user with administrator permissions. For more information, see [IAM Best Practices](#) in the *IAM User Guide*.

When granting permissions, you decide who is getting the permissions, the resources they get permissions for, and the specific actions that you want to allow on those resources.

Topics

- [Amazon DocumentDB Resources and Operations \(p. 155\)](#)
- [Understanding Resource Ownership \(p. 156\)](#)
- [Managing Access to Resources \(p. 156\)](#)
- [Specifying Policy Elements: Actions, Effects, Resources, and Principals \(p. 157\)](#)
- [Specifying Conditions in a Policy \(p. 158\)](#)

Amazon DocumentDB Resources and Operations

In Amazon DocumentDB, the primary resource is a *cluster*. Amazon DocumentDB supports other resources that can be used with the primary resource such as *instances*, *parameter groups*, and *event subscriptions*. These resources are referred to as *subresources*.

These resources and subresources have unique Amazon Resource Names (ARNs) associated with them, as shown in the following table.

Resource Type	ARN Format
Cluster	<code>arn:aws:rds:<i>region</i>:<i>account-id</i>:cluster:<i>db-cluster-name</i></code>
Cluster parameter group	<code>arn:aws:rds:<i>region</i>:<i>account-id</i>:cluster-pg:<i>cluster-parameter-group-name</i></code>
Cluster snapshot	<code>arn:aws:rds:<i>region</i>:<i>account-id</i>:cluster-snapshot:<i>cluster-snapshot-name</i></code>
Instance	<code>arn:aws:rds:<i>region</i>:<i>account-id</i>:db:<i>db-instance-name</i></code>
Security group	<code>arn:aws:rds:<i>region</i>:<i>account-id</i>:secgrp:<i>security-group-name</i></code>
Subnet group	<code>arn:aws:rds:<i>region</i>:<i>account-id</i>:subgrp:<i>subnet-group-name</i></code>

Amazon DocumentDB provides a set of operations to work with the Amazon DocumentDB resources. For a list of available operations, see [Actions](#).

Understanding Resource Ownership

A *resource owner* is the AWS account that created a resource. That is, the resource owner is the AWS account of the *principal entity* (the root account, an IAM user, or an IAM role) that authenticates the request that creates the resource. The following examples illustrate how this works:

- If you use the root account credentials of your AWS account to create an Amazon DocumentDB resource, such as an instance, your AWS account is the owner of the Amazon DocumentDB resource.
- If you create an IAM user in your AWS account and grant permissions to create Amazon DocumentDB resources to that user, the user can create Amazon DocumentDB resources. However, your AWS account, to which the user belongs, owns the Amazon DocumentDB resources.
- If you create an IAM role in your AWS account with permissions to create Amazon DocumentDB resources, anyone who can assume the role can create Amazon DocumentDB resources. Your AWS account, to which the role belongs, owns the Amazon DocumentDB resources.

Managing Access to Resources

A *permissions policy* describes who has access to what. The following section explains the available options for creating permissions policies.

Note

This section discusses using IAM in the context of Amazon DocumentDB. It doesn't provide detailed information about the IAM service. For complete IAM documentation, see [What Is IAM?](#) in the *IAM User Guide*. For information about IAM policy syntax and descriptions, see [AWS IAM Policy Reference](#) in the *IAM User Guide*.

Policies that are attached to an IAM identity are referred to as *identity-based* policies (IAM policies). Policies that are attached to a resource are referred to as *resource-based* policies. Amazon DocumentDB supports only identity-based policies (IAM policies).

Topics

- [Identity-Based Policies \(IAM Policies\) \(p. 156\)](#)
- [Resource-Based Policies \(p. 157\)](#)

Identity-Based Policies (IAM Policies)

You can attach policies to IAM identities. For example, you can do the following:

- **Attach a permissions policy to a user or a group in your account** – An account administrator can use a permissions policy that is associated with a particular user to grant permissions for that user to create an Amazon DocumentDB resource, such as an instance.
- **Attach a permissions policy to a role (grant cross-account permissions)** – You can attach an identity-based permissions policy to an IAM role to grant cross-account permissions. For example, the administrator in Account A can create a role to grant cross-account permissions to another AWS account (for example, Account B) or an AWS service as follows:
 1. Account A administrator creates an IAM role and attaches a permissions policy to the role that grants permissions on resources in Account A.
 2. Account A administrator attaches a trust policy to the role identifying Account B as the principal who can assume the role.
 3. Account B administrator can then delegate permissions to assume the role to any users in Account B. Doing this allows the users in Account B to create or access resources in Account A. The principal

in the trust policy can also be an AWS service principal if you want to grant permissions to an AWS service to assume the role.

For more information about using IAM to delegate permissions, see [Access Management](#) in the *IAM User Guide*.

The following is an example policy that allows the user with the ID 123456789012 to create instances for your AWS account. The new instance must use an option group and a parameter group that starts with default, and it must use the default subnet group.

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

For more information about using identity-based policies with Amazon DocumentDB, see [Using Identity-Based Policies \(IAM Policies\) for Amazon DocumentDB \(p. 158\)](#). For more information about users, groups, roles, and permissions, see [Identities \(Users, Groups, and Roles\)](#) in the *IAM User Guide*.

Resource-Based Policies

Other services, such as Amazon Simple Storage Service (Amazon S3), also support resource-based permissions policies. For example, you can attach a policy to an Amazon S3 bucket to manage access permissions to that bucket. Amazon DocumentDB doesn't support resource-based policies.

Specifying Policy Elements: Actions, Effects, Resources, and Principals

For each Amazon DocumentDB resource (see [Amazon DocumentDB Resources and Operations \(p. 155\)](#)), the service defines a set of API operations. For more information, see [Actions](#). To grant permissions for these API operations, Amazon DocumentDB defines a set of actions that you can specify in a policy. Performing an API operation can require permissions for more than one action.

The following are the basic policy elements:

- **Resource** – In a policy, you use an Amazon Resource Name (ARN) to identify the resource to which the policy applies.
- **Action** – You use action keywords to identify resource operations that you want to allow or deny. For example, the `rds:DescribeDBInstances` permission allows the user to perform the `DescribeDBInstances` operation.
- **Effect** – You specify the effect when the user requests the specific action—this can be either allow or deny. If you don't explicitly grant access to (allow) a resource, access is implicitly denied. You can also explicitly deny access to a resource, which you might do to make sure that a user cannot access it, even if a different policy grants access.

- **Principal** – In identity-based policies (IAM policies), the user that the policy is attached to is the implicit principal. For resource-based policies, you specify the user, account, service, or other entity that you want to receive permissions (applies to resource-based policies only). Amazon DocumentDB doesn't support resource-based policies.

To learn more about IAM policy syntax and descriptions, see [AWS IAM Policy Reference](#) in the *IAM User Guide*.

For a table showing all of the Amazon DocumentDB API actions and the resources that they apply to, see [Amazon DocumentDB API Permissions: Actions, Resources, and Conditions Reference \(p. 161\)](#).

Specifying Conditions in a Policy

When you grant permissions, you can use the IAM policy language to specify the conditions when a policy should take effect. For example, you might want a policy to be applied only after a specific date. For more information about specifying conditions in a policy language, see [Condition](#) in the *IAM User Guide*.

To express conditions, you use predefined condition keys. Amazon DocumentDB has no service-specific context keys that can be used in an IAM policy. For a list of global condition context keys that are available to all services, see [Available Keys for Conditions](#) in the *IAM User Guide*.

Managing Access Using Policies

You can have valid credentials to authenticate your requests, but if you don't have permissions, you can't create or access Amazon DocumentDB resources. For example, you must have permissions to create an Amazon DocumentDB cluster, create a cluster snapshot, modify cluster parameter groups, and so on.

The following sections describe how to manage permissions for Amazon DocumentDB. We recommend that you read the overview first.

- [Overview of Managing Access Permissions to Your Amazon DocumentDB Resources \(p. 155\)](#)
- [Using Identity-Based Policies \(IAM Policies\) for Amazon DocumentDB \(p. 158\)](#)

Using Identity-Based Policies (IAM Policies) for Amazon DocumentDB

Important

For certain management features, Amazon DocumentDB uses operational technology that is shared with Amazon RDS. Amazon DocumentDB console, AWS CLI, and API calls are logged as calls made to the Amazon RDS API.

We recommend that you first review the introductory topics that explain the basic concepts and options available for you to manage access to your Amazon DocumentDB resources. For more information, see [Overview of Managing Access Permissions to Your Amazon DocumentDB Resources \(p. 155\)](#).

This topic provides examples of identity-based policies in which an account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles).

The following is an example of an IAM policy.

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

The policy includes a single statement that specifies the following permissions for the IAM user:

- The policy allows the IAM user to create an instance using the [CreateDBInstance](#) action (this also applies to the [create-db-instance](#) AWS CLI operation and the AWS Management Console).
- The Resource element specifies that the user can perform actions on or with resources. You specify resources using an Amazon Resource Name (ARN). This ARN includes the name of the service that the resource belongs to (`rds`), the AWS Region (* indicates any Region in this example), the user account number (123456789012 is the user ID in this example), and the type of resource.

The Resource element in the example specifies the following policy constraints on resources for the user:

- The instance identifier for the new instance must begin with `test` (for example, `testCustomerData1`, `test-region2-data`).
- The cluster parameter group for the new instance must begin with `default`.
- The subnet group for the new instance must be the `default` subnet group.

The policy doesn't specify the Principal element because in an identity-based policy you don't specify the principal who gets the permission. When you attach policy to a user, the user is the implicit principal. When you attach a permissions policy to an IAM role, the principal identified in the role's trust policy gets the permissions.

For a table showing all of the Amazon DocumentDB API operations and the resources that they apply to, see [Amazon DocumentDB API Permissions: Actions, Resources, and Conditions Reference \(p. 161\)](#).

Permissions Required to Use the Amazon DocumentDB Console

For a user to work with the Amazon DocumentDB console, that user must have a minimum set of permissions. These permissions allow the user to describe the Amazon DocumentDB resources for their AWS account and to provide other related information, including Amazon EC2 security and network information.

If you create an IAM policy that is more restrictive than the minimum required permissions, the console won't function as intended for users with that IAM policy. To ensure that those users can still use the Amazon DocumentDB console, also attach the `AmazonDocDBConsoleFullAccess` managed policy to the user, as described in [AWS Managed \(Predefined\) Policies for Amazon DocumentDB \(p. 159\)](#).

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the Amazon DocumentDB API.

AWS Managed (Predefined) Policies for Amazon DocumentDB

AWS addresses many common use cases by providing standalone IAM policies that are created and administered by AWS. Managed policies grant necessary permissions for common use cases so you can

avoid having to investigate what permissions are needed. For more information, see [AWS Managed Policies](#) in the *IAM User Guide*.

The following AWS managed policies, which you can attach to users in your account, are specific to Amazon DocumentDB:

- **AmazonDocDBReadOnlyAccess** – Grants read-only access to all Amazon DocumentDB resources for the root AWS account.
- **AmazonDocDBFullAccess** – Grants full access to all Amazon DocumentDB resources for the root AWS account.
- **AmazonDocDBConsoleFullAccess** – Grants full access to manage Amazon DocumentDB resources using the AWS Management Console.

You can also create custom IAM policies that allow users to access the required Amazon DocumentDB API actions and resources. You can attach these custom policies to the IAM users or groups that require those permissions.

Customer Managed Policy Examples

In this section, you can find example user policies that grant permissions for various Amazon DocumentDB actions. These policies work when you are using Amazon DocumentDB API actions, AWS SDKs, or the AWS CLI. When you are using the console, you need to grant additional permissions specific to the console, which is discussed in [Permissions Required to Use the Amazon DocumentDB Console \(p. 159\)](#).

For certain management features, Amazon DocumentDB uses operational technology that is shared with Amazon Relational Database Service (Amazon RDS) and Amazon Neptune.

Note

All examples use the US East (N. Virginia) Region (us-east-1) and contain fictitious account IDs.

Examples

- [Example 1: Allow a User to Perform Any Describe Action on Any Amazon DocumentDB Resource \(p. 160\)](#)
- [Example 2: Prevent a User from Deleting an Instance \(p. 161\)](#)
- [Example 3: Prevent a User from Creating a Cluster unless Storage Encryption is Enabled \(p. 161\)](#)

Example 1: Allow a User to Perform Any Describe Action on Any Amazon DocumentDB Resource

The following permissions policy grants permissions to a user to run all of the actions that begin with `Describe`. These actions show information about an Amazon DocumentDB resource, such as an instance. The wildcard character (*) in the `Resource` element indicates that the actions are allowed for all Amazon DocumentDB resources that are owned by the account.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "AllowRDSDescribe",  
            "Effect": "Allow",  
            "Action": "rds:Describe*",  
            "Resource": "*"  
        }  
    ]  
}
```

```
}
```

Example 2: Prevent a User from Deleting an Instance

The following permissions policy grants permissions to prevent a user from deleting a specific instance. For example, you might want to deny the ability to delete your production instances to any user that is not an administrator.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "DenyDelete1",  
            "Effect": "Deny",  
            "Action": "rds:DeleteDBInstance",  
            "Resource": "arn:aws:rds:us-east-1:123456789012:db:my-db-instance"  
        }  
    ]  
}
```

Example 3: Prevent a User from Creating a Cluster unless Storage Encryption is Enabled

The following permissions policy denies permissions to a user from creating an Amazon DocumentDB cluster unless storage encryption is enabled.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "PreventUnencryptedDocumentDB",  
            "Effect": "Deny",  
            "Action": "RDS>CreateDBCluster",  
            "Condition": {  
                "Bool": {  
                    "rds:StorageEncrypted": "false"  
                },  
                "StringEquals": {  
                    "rds:DatabaseEngine": "docdb"  
                }  
            },  
            "Resource": "*"  
        }  
    ]  
}
```

Amazon DocumentDB API Permissions: Actions, Resources, and Conditions Reference

Use the following sections as a reference when you set up [Managing Access Using Policies \(p. 158\)](#) and write permissions policies that you can attach to an IAM identity (identity-based policies).

The following lists each Amazon DocumentDB API operation. Included in the list are the corresponding actions for which you can grant permissions to perform the action, the AWS resource that you can grant the permissions for, and condition keys that you can include for fine-grained access control. You specify the actions in the policy's Action field, the resource value in the policy's Resource field, and conditions in the policy's Condition field. For more information about conditions, see [Specifying Conditions in a Policy \(p. 158\)](#).

You can use AWS-wide condition keys in your Amazon DocumentDB policies to express conditions. For a complete list of AWS-wide keys, see [Available Keys](#) in the *IAM User Guide*.

You can test IAM policies with the IAM policy simulator. It automatically provides a list of resources and parameters required for each AWS action, including Amazon DocumentDB actions. The IAM policy simulator determines the permissions that are required for each of the actions that you specify. For information about the IAM policy simulator, see [Testing IAM Policies with the IAM Policy Simulator](#) in the *IAM User Guide*.

Note

To specify an action, use the `rds:` prefix followed by the API operation name (for example, `rds:CreateDBInstance`).

The following lists Amazon RDS API operations and their related actions, resources, and condition keys.

Topics

- [Amazon DocumentDB Actions That Support Resource-Level Permissions \(p. 162\)](#)
- [Amazon DocumentDB Actions That Don't Support Resource-Level Permissions \(p. 166\)](#)

Amazon DocumentDB Actions That Support Resource-Level Permissions

Resource-level permissions provide the ability to specify the resources on which users are allowed to perform actions. Amazon DocumentDB has partial support for resource-level permissions. This means that for certain Amazon DocumentDB actions, you can control when users are allowed to use those actions based on conditions that have to be fulfilled, or specific resources that users are allowed to use. For example, you can grant users permission to modify only specific instances.

The following lists Amazon DocumentDB API operations and their related actions, resources, and condition keys.

For certain management features, Amazon DocumentDB uses operational technology that is shared with Amazon RDS.

Amazon DocumentDB API Operations and Actions	Resources	Condition Keys
AddTagsToResource <code>rds:AddTagsToResource</code>	Instance <code>arn:aws:rds:region:account-id:db:db-instance-name</code>	<code>rds:db-tag</code>
	Subnet group <code>arn:aws:rds:region:account-id:subgrp:subnet-group-name</code>	<code>rds:subgrp-tag</code>
ApplyPendingMaintenanceAction <code>rds:ApplyPendingMaintenanceAction</code>	Instance <code>arn:aws:rds:region:account-id:db:db-instance-name</code>	<code>rds:db-tag</code>
CopyDBClusterSnapshot <code>rds:CopyDBClusterSnapshot</code>	Cluster snapshot <code>arn:aws:rds:region:account-id:cluster-snapshot:cluster-snapshot-name</code>	<code>rds:cluster-snapshot-tag</code>

Amazon DocumentDB API Operations and Actions	Resources	Condition Keys
CreateDBCluster <code>rds:CreateDBCluster</code>	Cluster <code>arn:aws:rds:<i>region:account-id:cluster:db-cluster-name</i></code>	<code>rds:cluster-tag</code>
	Cluster parameter group <code>arn:aws:rds:<i>region:account-id:cluster-pg:cluster-parameter-group-name</i></code>	<code>rds:cluster-pg-tag</code>
	Subnet group <code>arn:aws:rds:<i>region:account-id:subgrp:subnet-group-name</i></code>	<code>rds:subgrp-tag</code>
CreateDBClusterParameterGroup <code>rds:CreateDBClusterParameterGroup</code>	Cluster parameter group <code>arn:aws:rds:<i>region:account-id:cluster-pg:cluster-parameter-group-name</i></code>	<code>rds:cluster-pg-tag</code>
	Cluster snapshot <code>arn:aws:rds:<i>region:account-id:cluster-snapshot:cluster-snapshot-name</i></code>	<code>rds:cluster-snapshot-tag</code>
CreateDBClusterSnapshot <code>rds:CreateDBClusterSnapshot</code>	Cluster <code>arn:aws:rds:<i>region:account-id:cluster:db-cluster-name</i></code>	<code>rds:cluster-tag</code>
	Cluster snapshot <code>arn:aws:rds:<i>region:account-id:cluster-snapshot:cluster-snapshot-name</i></code>	<code>rds:cluster-snapshot-tag</code>
	Instance <code>arn:aws:rds:<i>region:account-id:db:db-instance-name</i></code>	<code>rds:DatabaseClass</code> <code>rds:db-tag</code>
CreateDBInstance <code>rds:CreateDBInstance</code>	Cluster <code>arn:aws:rds:<i>region:account-id:cluster:db-cluster-name</i></code>	<code>rds:cluster-tag</code>
	Subnet group <code>arn:aws:rds:<i>region:account-id:subgrp:subnet-group-name</i></code>	<code>rds:subgrp-tag</code>
DeleteDBInstance <code>rds:DeleteDBInstance</code>	Instance <code>arn:aws:rds:<i>region:account-id:db:db-instance-name</i></code>	<code>rds:db-tag</code>

Amazon DocumentDB API Operations and Actions	Resources	Condition Keys
DeleteDBSubnetGroup	Subnet group	rds:subgrp-tag
rds:DeleteDBSubnetGroup	arn:aws:rds:region:account-id:subgrp:subnet-group-name	
DescribeDBClusterParameterGroups	Cluster parameter group	rds:cluster-pg-tag
rds:DescribeDBClusterParameterGroups	arn:aws:rds:region:account-id:cluster-pg:cluster-parameter-group-name	
DescribeDBClusterParameters	Cluster parameter group	rds:cluster-pg-tag
rds:DescribeDBClusterParameters	arn:aws:rds:region:account-id:cluster-pg:cluster-parameter-group-name	
DescribeDBClusters	Cluster	rds:cluster-tag
rds:DescribeDBClusters	arn:aws:rds:region:account-id:cluster:db-cluster-instance-name	
DescribeDBClusterSnapshots	Cluster snapshot	rds:cluster-snapshot-tag
rds:DescribeDBClusterSnapshots	arn:aws:rds:region:account-id:cluster-snapshot:cluster-snapshot-name	
DescribeDBSubnetGroups	Subnet group	rds:subgrp-tag
rds:DescribeDBSubnetGroups	arn:aws:rds:region:account-id:subgrp:subnet-group-name	
DescribePendingMaintenanceActions	Instances	rds:DatabaseClass
rds:DescribePendingMaintenanceActions	arn:aws:rds:region:account-id:db:db-instance-name	rds:db-tag
FailoverDBCluster	Cluster	rds:cluster-tag
rds:FailoverDBCluster	arn:aws:rds:region:account-id:cluster:db-cluster-instance-name	
ListTagsForResource	Instance	rds:db-tag
rds>ListTagsForResource	arn:aws:rds:region:account-id:db:db-instance-name	
	Subnet group	rds:subgrp-tag
	arn:aws:rds:region:account-id:subgrp:subnet-group-name	

Amazon DocumentDB API Operations and Actions	Resources	Condition Keys
ModifyDBCluster	Cluster	rds:cluster-tag
rds:ModifyDBCluster	arn:aws:rds: <i>region:account-id:cluster:db-cluster-name</i>	
	Cluster parameter group arn:aws:rds: <i>region:account-id:cluster-pg:cluster-parameter-group-name</i>	rds:cluster-pg-tag
ModifyDBClusterParameter	Cluster parameter group	rds:cluster-pg-tag
rds:ModifyDBClusterParameter	arn:aws:rds: <i>region:account-id:cluster-pg:cluster-parameter-group-name</i>	
ModifyDBClusterSnapshot	Cluster snapshot	rds:cluster-snapshot-tag
rds:ModifyDBClusterSnapshot	arn:aws:rds: <i>region:account-id:cluster-snapshot:cluster-snapshot-name</i>	
ModifyDBInstance	Instance	rds:DatabaseClass
rds:ModifyDBInstance	arn:aws:rds: <i>region:account-id:db:db-instance-name</i>	rds:db-tag
RebootDBInstance	Instance	rds:db-tag
rds:RebootDBInstance	arn:aws:rds: <i>region:account-id:db:db-instance-name</i>	
RemoveTagsFromResource	Instance	rds:db-tag
rds:RemoveTagsFromResource	arn:aws:rds: <i>region:account-id:db:db-instance-name</i>	
	Subnet group arn:aws:rds: <i>region:account-id:subgrp:subnet-group-name</i>	rds:subgrp-tag
ResetDBClusterParameter	Cluster parameter group	rds:cluster-pg-tag
rds:ResetDBClusterParameter	arn:aws:rds: <i>region:account-id:cluster-pg:cluster-parameter-group-name</i>	
RestoreDBClusterFromSnapshot	Cluster	rds:cluster-tag
rds:RestoreDBClusterFromSnapshot	arn:aws:rds: <i>region:account-id:cluster:db-cluster-instance-name</i>	

Amazon DocumentDB API Operations and Actions	Resources	Condition Keys
	Cluster snapshot <code>arn:aws:rds:<i>region</i>:<i>account-id</i>:cluster-snapshot:<i>cluster-snapshot-name</i></code>	<code>rds:cluster-snapshot-tag</code>
RestoreDBClusterToPointInTime	<code>Cluster</code> <code>rds:RestoreDBClusterToPointInTime:<i>region</i>:<i>account-id</i>:cluster:<i>db-cluster-instance-name</i></code>	<code>rds:cluster-tag</code>
	Subnet group <code>arn:aws:rds:<i>region</i>:<i>account-id</i>:subgrp:<i>subnet-group-name</i></code>	<code>rds:subgrp-tag</code>

Amazon DocumentDB Actions That Don't Support Resource-Level Permissions

You can use all Amazon DocumentDB actions in an IAM policy to either grant or deny users permission to use that action. However, not all Amazon DocumentDB actions support resource-level permissions, which enable you to specify the resources on which an action can be performed. The following Amazon DocumentDB API actions currently don't support resource-level permissions. Therefore, to use these actions in an IAM policy, you must grant users permission to use all resources for the action by using a * wildcard for the Resource element in your statement.

- `rds:DescribeDBClusterSnapshots`
- `rds:DescribeDBInstances`

Managing Amazon DocumentDB Users

In Amazon DocumentDB, users authenticate to a cluster in conjunction with a password. Each cluster has a master user and password that is established during cluster creation.

Note

All new users created before **March 26, 2020** have been granted the `dbAdminAnyDatabase`, `readWriteAnyDatabase`, and `clusterAdmin` roles. It is recommended that you reevaluate all users and modify the roles as necessary to enforce least privilege for all users in your clusters. For more information, see [Restricting Database Access Using Role-Based Access Control \(p. 169\)](#).

Master and serviceadmin User

A newly created Amazon DocumentDB cluster has two users: the master user and the `serviceadmin` user.

The *master user* is a single, privileged user that can perform administrative tasks and create additional users with roles. When you connect to an Amazon DocumentDB cluster for the first time, you must authenticate using the master user name and password. The master user receives these administrative

permissions for an Amazon DocumentDB cluster when that cluster is created, and is granted the role of root.

The serviceadmin user is created implicitly when the cluster is created. Every Amazon DocumentDB cluster has a serviceadmin user that provides AWS the ability to manage your cluster. You cannot log in as, drop, rename, change the password, or change the permissions for serviceadmin. Any attempt to do so results in an error.

Note

The master and serviceadmin users for an Amazon DocumentDB cluster cannot be deleted and the role of root for the master user cannot be revoked.

If you forget your master user password, you can reset it using the AWS Management Console or the AWS CLI.

Creating Additional Users

After you connect as the master user (or any user that has the role `createUser`), you can create a new user, as shown below.

```
db.createUser(
{
    user: "sample-user-1",
    pwd: "password123",
    roles:
        [ {"db": "admin", "role": "dbAdminAnyDatabase"} ]
})
```

To view user details, you can use the `show users` command as follows. You can additionally remove users with the `dropUser` command. For more information, see [Common Commands \(p. 177\)](#).

```
show users
{
    "_id" : "serviceadmin",
    "user" : "serviceadmin",
    "db" : "admin",
    "roles" : [
        {
            "role" : "root",
            "db" : "admin"
        }
    ],
    {
        "_id" : "myMasterUser",
        "user" : "myMasterUser",
        "db" : "admin",
        "roles" : [
            {
                "role" : "root",
                "db" : "admin"
            }
        ],
        {
            "_id" : "sample-user-1",
            "user" : "sample-user-1",
            "db" : "admin",
        }
    ]
}
```

```
"roles" : [
  {
    "role" : "dbAdminAnyDatabase",
    "db" : "admin"
  }
]
```

In the example above, the new user `sample-user-1` is attributed to the `admin` database. This is always the case for a new user. Amazon DocumentDB does not have the concept of an `authenticationDatabase` and thus all authentication is performed in the context of the `admin` database.

When creating users, if you omit the `db` field when specifying the role, Amazon DocumentDB will implicitly attribute the role to the database in which the connection is being issued against. For example, if your connection is issued against the database `sample-database` and you run the following command, the user `sample-user-2` will be created in the `admin` database and will have `readWrite` permissions to the database `sample-database`.

```
db.createUser(
{
  user: "sample-user-2",
  pwd: "password123",
  roles:
    ["readWrite"]
})
```

Creating users with roles that are scoped across all databases (for example, `readInAnyDatabase`) require that you are either in the context of the `admin` database when creating the user or you explicitly state the database for the role when creating the user.

To switch the context of your database, you can use the following command.

```
use admin
```

To learn more about Role Based Access Control and enforcing least privilege amongst the users in your cluster, see [Restricting Database Access Using Role-Based Access Control \(p. 169\)](#).

Automatically Rotating Passwords for Amazon DocumentDB

With AWS Secrets Manager, you can replace hardcoded credentials in your code (including passwords) with an API call to Secrets Manager to retrieve the secret programmatically. This helps ensure that the secret can't be compromised by someone examining your code, because the secret simply isn't there. Also, you can configure Secrets Manager to automatically rotate the secret for you according to a schedule that you specify. This enables you to replace long-term secrets with short-term ones, which helps to significantly reduce the risk of compromise.

Using Secrets Manager, you can automatically rotate your Amazon DocumentDB passwords (that is, `secrets`) using an AWS Lambda function that Secrets Manager provides.

For more information about AWS Secrets Manager and native integration with Amazon DocumentDB, see the following:

- [Blog: How to rotate Amazon DocumentDB and Amazon Redshift credentials in AWS Secrets Manager](#)

- [What Is AWS Secrets Manager?](#)
- [Rotating Secrets for Amazon DocumentDB](#)

Restricting Database Access Using Role-Based Access Control

You can restrict access to the actions that users can perform on databases using *role-based access control* (RBAC) in Amazon DocumentDB (with MongoDB compatibility). RBAC works by granting one or more roles to a user. These roles determine the operations that a user can perform on database resources. Amazon DocumentDB currently supports both built-in roles that are scoped at the database level, such as `read`, `readWrite`, `readAnyDatabase`, `clusterAdmin`, and user-defined roles that can be scoped to specific actions and granular resources such as collections based on your requirements.

Common use cases for RBAC include enforcing least privileges by creating users with read-only access to the databases or collections in a cluster, and multi-tenant application designs that enable a single user to access a given database or collection in a cluster.

Note

All new users created before **March 26, 2020** have been granted the `dbAdminAnyDatabase`, `readWriteAnyDatabase`, and `clusterAdmin` roles. It is recommended that you reevaluate all existing users and modify the roles as necessary to enforce least privileges for your clusters.

Topics

- [RBAC Concepts \(p. 169\)](#)
- [Getting Started with RBAC built-in roles \(p. 170\)](#)
- [Getting Started with RBAC user-defined roles \(p. 173\)](#)
- [Connecting to Amazon DocumentDB as a User \(p. 176\)](#)
- [Common Commands \(p. 177\)](#)
- [Functional Differences \(p. 181\)](#)
- [Limits \(p. 181\)](#)
- [Restricting Database Access Using Role-Based Access Control \(p. 181\)](#)

RBAC Concepts

The following are important terms and concepts related to role-based access control. For more information on Amazon DocumentDB users, see [Managing Amazon DocumentDB Users \(p. 166\)](#).

- **User** — An individual entity that can authenticate to the database and perform operations.
- **Password** — A secret that is used to authenticate the user.
- **Role** — Authorizes a user to perform actions on one or more databases.
- **Admin Database** — The database in which users are stored and authorized against.
- **Database (db)** — The namespace within clusters that contains collections for storing documents.

The following command creates a user named `sample-user`.

```
db.createUser({user: "sample-user", pwd: "abc123", roles: [{role: "read", db: "sample-database"}]})
```

In this example:

- user: "sample-user" — Indicates the user name.
- pwd: "abc123" — Indicates the user password.
- role: "read", db: "sample-database" — Indicates that the user sample-user will have read permissions in sample-database.



The following example shows the output after you get the user sample-user with db.getUser(sample-user). In this example, the user sample-user resides in the admin database but has the read role for the database sample-database.

```
{
  "_id" : "sample-user", ← User ID
  "user" : "sample-user", ← Username
  "db" : "admin", ← All users created in
  "roles" : [           the admin database
    {
      "db" : "sample-database", ← User sample-user has
      "role" : "read"          read permissions in
                                database sample-database
    }
  ]
}
```

When creating users, if you omit the db field when specifying the role, Amazon DocumentDB will implicitly attribute the role to the database in which the connection is being issued against. For example, if your connection is issued against the database sample-database and you run the following command, the user sample-user will be created in the admin database and will have readWrite permissions to the database sample-database.

```
db.createUser({user: "sample-user", pwd: "abc123", roles: ["readWrite"]})
```

Output from this operation looks something like the following.

```
{
  "user": "sample-user",
  "roles": [
    {
      "db": "sample-database",
      "role": "readWrite"
    }
  ]
}
```

Creating users with roles that are scoped across all databases (for example, `readAnyDatabase`) require that you either be in the context of the admin database when creating the user, or you explicitly state the database for the role when creating the user. To issue commands against the admin database, you can use the command `use admin`. For more information, see [Common Commands \(p. 177\)](#).

Getting Started with RBAC built-in roles

To help you get started with role-based access control, this section walks you through an example scenario of enforcing least privilege by creating roles for three users with different job functions.

- user1 is a new manager that needs to be able to view and access all databases in a cluster.
- user2 is a new employee that needs access to only one database, sample-database-1, in that same cluster.
- user3 is an existing employee that needs to view and access a different database, sample-database-2 that they didn't have access to before, in the same cluster.

At a point later, both user1 and user2 leave the company and so their access must be revoked.

To create users and grant roles, the user that you authenticate to the cluster with must have an associated role that can perform actions for `createUser` and `grantRole`. For example, the roles `admin` and `userAdminAnyDatabase` can both grant such abilities, for example. For actions per role, see [Restricting Database Access Using Role-Based Access Control \(p. 181\)](#).

Note

In Amazon DocumentDB, all user and role operations (for example, `create`, `get`, `drop`, `grant`, `revoke`, etc.) are implicitly performed in the `admin` database whether or not you are issuing commands against the `admin` database.

First, to understand what the current users and roles are in the cluster, you can run the `show users` command, as in the following example. You will see two users, `serviceadmin` and the master user for the cluster. These two users always exist and cannot be deleted. For more information, see [Managing Amazon DocumentDB Users \(p. 166\)](#).

```
show users
```

For user1, create a role with read and write access to all databases in the entire cluster with the following command.

```
db.createUser({user: "user1", pwd: "abc123", roles: [{role: "readWriteAnyDatabase", db: "admin"}]})
```

Output from this operation looks something like the following.

```
{
  "user": "user1",
  "roles": [
    {
      "role": "readWriteAnyDatabase",
      "db": "admin"
    }
  ]
}
```

For user2, create a role with read-only access to the database sample-database-1 with the following command.

```
db.createUser({user: "user2", pwd: "abc123", roles: [{role: "read", db: "sample-database-1"}]})
```

Output from this operation looks something like the following.

```
{
  "user": "user2",
  "roles": [
    {
      "role": "read",
      "db": "sample-database-1"
    }
  ]
}
```

```

        "role":"read",
        "db":"sample-database-1"
    }
}

```

To simulate the scenario that `user3` is an existing user, first create the user `user3`, and then assign a new role to `user3`.

```
db.createUser({user: "user3", pwd: "abc123", roles: [{role: "readWrite", db: "sample-database-1"}]})
```

Output from this operation looks something like the following.

```
{
  "user": "user3",
  "roles": [
    {
      "role": "readWrite",
      "db": "sample-database-1"
    }
  ]
}
```

Now that the user `user3` has been created, assign `user3` the role `read` to `sample-database-2`.

```
db.grantRolesToUser("user3", [{role: "read", db: "sample-database-2"}])
```

Lastly, both `user1` and `user2` leave the company and need their access to the cluster revoked. You can do this by dropping the users, as follows.

```
db.dropUser("user1")
db.dropUser("user2")
```

To ensure that all users have the appropriate roles, you can list all users with the following command.

```
show users
```

Output from this operation looks something like the following.

```
{
  "_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"
        }
    ]
}

```

Getting Started with RBAC user-defined roles

To help you get started with user-defined roles, this section walks you through an example scenario of enforcing least privilege by creating roles for three users with different job functions.

In this example, the following applies:

- `user1` is a new manager that needs to be able to view and access all databases in a cluster.
- `user2` is a new employee that needs only the ‘find’ action to only one database, `sample-database-1`, in that same cluster.
- `user3` is an existing employee that needs to view and access a specific collection, `col2` in a different database, `sample-database-2` that they didn’t have access to before, in the same cluster.
- For `user1`, create a role with read and write access to all databases in the entire cluster with the following command.

```

db.createUser(
{
    user: "user1", pwd: "abc123",
    roles: [{role: "readWriteAnyDatabase", db: "admin"}]
}
)

```

Output from this operation looks something like the following.

```

{
    "user": "user1",
    "roles": [
        {
            "role": "readWriteAnyDatabase",
            "db": "admin"
        }
    ]
}

```

For `user2`, create a role with ‘find’ privileges to all collections in the database `sample-database-1` with the following command. Note that this role would ensure that any associated users can only run find queries.

```
db.createRole(
{
    role: "findRole",
    privileges: [
    {
        resource: {db: "sample-database-1", collection: ""}, actions: ["find"]
    }],
    roles: []
}
)
```

Output from this operation looks something like the following.

```
{
    "role": "findRole",
    "privileges": [
    {
        "resource": {
            "db": "sample-database-1",
            "collection": ""
        },
        "actions": [
            "find"
        ]
    }
],
    "roles": [
    ]
}
```

Next, create the user (`user2`) and attach the recently created role `findRole` to the user.

```
db.createUser(
{
    user: "user2",
    pwd: "abc123",
    roles: []
})

db.grantRolesToUser("user2",["findRole"])
```

To simulate the scenario that `user3` is an existing user, first create the user `user3`, and then create a new role called `collectionRole` which we will in the next step assing to `user3`.

Now you can assign a new role to `user3`. This new role will allow `user3` to be able to insert, update, delete and find access to one specific collection `col2` in `sample-database-2`.

```
db.createUser(
{
    user: "user3",
    pwd: "abc123",
    roles: []
})

db.createRole(
{
    role: "collectionRole",
    privileges: [
    {
```

```
        resource: {db: "sample-database-2", collection: "col2"}, actions: ["find",
"update", "insert", "remove"]
    ],
    roles: []
}
}
```

Output from this operation looks something like the following.

```
{
  "role": "collectionRole",
  "privileges": [
    {
      "resource": {
        "db": "sample-database-2",
        "collection": "col2"
      },
      "actions": [
        "find",
        "update",
        "insert",
        "remove"
      ]
    }
  ],
  "roles": [
  ]
}
```

Now that the user `user3` has been created, you can grant `user3` the role `collectionFind`.

```
db.grantRolesToUser("user3", ["collectionRole"])
```

Lastly, both `user1` and `user2` leave the company and need their access to the cluster revoked. You can do this by dropping the users, as follows.

```
db.dropUser("user1")
db.dropUser("user2")
```

To ensure that all users have the appropriate roles, you can list all users with the following command.

```
show users
```

Output from this operation looks something like the following.

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

Connecting to Amazon DocumentDB as a User

When connecting to an Amazon DocumentDB cluster, you connect in the context of a particular database. By default, if you don't specify a database in your connection string, you are automatically connected to the cluster in the context of the test database. All collection level commands like `insert` and `find` are issued against collections in the test database.

To see the database you are in the context of or — in other words — issuing commands against, use the `db` command in the mongo shell, as follows.

Query:

```
db
```

Output:

```
test
```

Although the default connection might be in the context of the test database, that does not necessarily mean that the user associated with the connection is authorized to perform actions on the test database. In the preceding example scenario, if you authenticate as the user `user3`, which has the `readWrite` role for the `sample-database-1` database, the default context of your connection is the test database. However, if you try to insert a document into a collection on the test database, you will receive an *Authorization failure* error message. This is because that user is not authorized to perform that command on that database, as shown below.

Query:

```
db
```

Output:

```
test
```

Query:

```
db.col.insert({x:1})
```

Output:

```
WriteCommandError({ "ok" : 0, "code" : 13, "errmsg" : "Authorization failure" })
```

If you change the context of your connection to the sample-database-1 database, you can write to the collection for which the user has the authorization to do so.

Query:

```
use sample-database-1
```

Output:

```
switched to db sample-database-1
```

Query:

```
db.col.insert({x:1})
```

Output:

```
WriteResult({ "nInserted" : 1})
```

When you authenticate to a cluster with a particular user, you can also specify the database in the connection string. Doing so removes the necessity to perform the use command after the user has been authenticated to the admin database.

The following connection string authenticates the user against the admin database, but the context of the connection will be against the sample-database-1 database.

```
mongo "mongodb://user3:abc123@sample-cluster.node.us-east-1.docdb.amazonaws.com:27017/  
sample-database-2"
```

Common Commands

This section provides examples of common commands using role-based access control in Amazon DocumentDB. You must be in the context of the admin database to create and modify users and roles. You can use the use admin command to switch to the admin database.

Note

Modifications to the users and roles will implicitly occur in the admin database. Creating users with roles that are scoped across all databases (for example, readAnyDatabase) requires that you are either in the context of the admin database (that is, use admin) when creating the user, or you explicitly state the database for the role when creating the user (as shown in Example 2 in this section).

Example 1: Create a user with read role for the database foo.

```
db.createUser({user: "readInFooBar", pwd: "abc123", roles: [{role: "read", db: "foo"}]})
```

Output from this operation looks something like the following.

```
{  
  "user": "readInFooBar",  
  "roles": [  
    {  
      "role": "read",  
      "db": "foo"  
    }  
  ]  
}
```

Example 2: Create a user with read access on all databases.

```
db.createUser({user: "readAllDBs", pwd: "abc123", roles: [{role: "readAnyDatabase", db: "admin"}]})
```

Output from this operation looks something like the following.

```
{  
  "user": "readAllDBs",  
  "roles": [  
    {  
      "role": "readAnyDatabase",  
      "db": "admin"  
    }  
  ]  
}
```

Example 3: Grant read role to an existing user on a new database.

```
db.grantRolesToUser("readInFooBar", [{role: "read", db: "bar"}])
```

Example 4: Update a user's role.

```
db.updateUser("readInFooBar", {roles: [{role: "read", db: "foo"}, {role: "read", db: "baz"}]})
```

Example 5: Revoke access to a database for a user.

```
db.revokeRolesFromUser("readInFooBar", [{role: "read", db: "baz"}])
```

Example 6: Describe a built-in role.

```
db.getRole("read", {showPrivileges:true})
```

Output from this operation looks something like the following.

```
{  
  "role": "read",  
  "db": "sample-database-1",  
  "privileges": [  
    {  
      "privilege": "r",  
      "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"
            ]
        }
    ]
}

```

Example 7: Drop a user from the cluster.

```
db.dropUser("readInFooBar")
```

Output from this operation looks something like the following.

```
true
```

Example 8: Create a role with read and write access to a specific collection

```

db.createRole(
{
    role: "collectionRole",
    privileges: [
        {
            resource: {db: "sample-database-2", collection: "col2"}, actions: ["find",
            "update", "insert", "remove"]
        },
        roles: []
    ]
}

```

```
)
```

Output from this operation looks something like the following.

```
{  
    "role": "collectionRole",  
    "privileges": [  
        {  
            "resource": {  
                "db": "sample-database-2",  
                "collection": "col2"  
            },  
            "actions": [  
                "find",  
                "update",  
                "insert",  
                "remove"  
            ]  
        }  
    ],  
    "roles": [  
    ]  
}
```

Example 9: Create a user and assign a user defined role

```
db.createUser(  
{  
    user: "user3",  
    pwd: "abc123",  
    roles: []  
})  
  
db.grantRolesToUser("user3", ["collectionRole"])
```

Example 10: Grant additional privileges to a user defined role

```
db.grantPrivilegesToRole(  
    "collectionRole",  
    [  
        {  
            resource: { db: "sample-database-1", collection: "col1" },  
            actions: ["find", "update", "insert", "remove"]  
        }  
    ]  
)
```

Example 11: Remove privileges from a user defined role

```
db.revokePrivilegesFromRole(  
    "collectionRole",  
    [  
        {  
            resource: { db: "sample-database-1", collection: "col2" },  
            actions: ["find", "update", "insert", "remove"]  
        }  
    ]  
)
```

Example 12: Update an existing user defined role

```
db.updateRole(  
  "collectionRole",  
  {  
    privileges: [  
      {  
        resource: {db: "sample-database-3", collection: "sample-collection-3"}, actions:  
        ["find", "update", "insert", "remove"]  
      }],  
      roles: []  
    }  
)
```

Functional Differences

In Amazon DocumentDB, user and role definitions are stored in the `admin` database and users are authenticated against the `admin` database. This functionality differs from the MongoDB Community Edition, but is consistent with MongoDB Atlas.

Amazon DocumentDB also supports change streams, which provide a time-ordered sequence of change events that occur within your cluster's collections. The `listChangeStreams` action is applied at the cluster level (that is, across all databases), and the `modifyChangeStreams` action can be applied at the database level and cluster level.

Limits

The following table contains the limits for Role-Based Access Control in Amazon DocumentDB.

Description	Limit
Number of users per cluster	1000
Number of roles associated with a user	1000
Number of user-defined roles	100
Number of resources associated with a privilege	100

Restricting Database Access Using Role-Based Access Control

With role-based access control, you can create a user and grant it one or more roles to determine what operations that user can perform in a database or cluster.

The following is a list of built-in roles that are currently supported in Amazon DocumentDB.

Note

In Amazon DocumentDB 4.0, the `ListCollection` and `ListDatabase` commands can optionally use the `authorizedCollections` and `authorizedDatabases` parameters to list the collections and databases that the user has permission to access with requiring the `listCollections` and `listDatabase` roles, respectively. Also, users now have the ability to kill their own cursors without requiring the `KillCursor` role.

Role Type	Role Name	Description	Actions
Database User	read	Grants a user read access to the specified database.	changeStreams collStats dbStats find killCursors listIndexes listCollections
	readWrite	Grants the user read and write access to the specified database.	All actions from read permissions. createCollection dropCollection createIndex dropIndex insert killCursors listIndexes listCollections remove update
Cluster User	readAnyDatabase	Grants a user read access to all databases in the cluster.	All actions from read permissions. listChangeStreams listDatabases
	readWriteAnyDatabase	Grants a user read and write access to all databases in the cluster.	All actions from readWrite permissions. listChangeStreams listDatabases
	userAdminAnyDatabase	Grants a user the ability to assign and modify the roles or privileges any user has to	changeCustomData changePassword createUser dropRole

Role Type	Role Name	Description	Actions
		the specified database.	dropUser grantRole listDatabases revokeRole viewRole viewUser
	dbAdminAnyDatabase	Grants a user the ability to perform database administration roles on any specified database.	All actions from dbAdmin permissions. dropCollection listDatabases listChangeStreams modifyChangeStreams
Superuser	root	Grants a user access to the resources and operations of all the following roles combined: <code>readWriteAnyDatabase</code> , <code>dbAdminAnyDatabase</code> , <code>userAdminAnyDatabase</code> , <code>clusterAdmin</code> , <code>restore</code> , and <code>backup</code> .	All actions from <code>readWriteAnyDatabase</code> , <code>dbAdminAnyDatabase</code> , <code>userAdminAnyDatabase</code> , <code>clusterAdmin</code> , <code>restore</code> , and <code>backup</code> .

Role Type	Role Name	Description	Actions
Database Administration	dbAdmin	Grants a user the ability to perform administrative tasks on the specified database.	collMod collStats createCollection createIndex dropCollection dropDatabase dropIndex dbStats find killCursors listIndexes listCollections modifyChangeStreams
	dbOwner	Grants a user the ability to perform any administrative tasks on the specified database by combining the roles dbAdmin and readWrite.	All actions from dbAdmin and readWrite.
Cluster Administration	clusterAdmin	Grants a user the greatest cluster management access by combining the clusterManager, clusterMonitor, and hostManager roles.	All actions from clusterManager, clusterMonitor, and hostManager. listChangeStreams dropDatabase modifyChangeStreams
	clusterManager	Grants a user the ability to take management and monitoring actions on the specified cluster.	listChangeStreams listSessions modifyChangeStreams replSetGetConfig

Role Type	Role Name	Description	Actions
	clusterMonitor	Grants a user the ability to have read-only access to monitoring tools.	collStats dbStats find getParameter hostInfo indexStats killCursors listChangeStreams listCollections listDatabases listIndexes listSessions replSetGetConfig serverStatus top
	hostManager	Grants a user the ability to monitor and manage servers.	killCursors killAnyCursor killAnySession killOp
Backup Administration	backup	Grants a user the access needed to back up data.	getParameter insert find listChangeStreams listCollections listDatabases listIndexes update

Role Type	Role Name	Description	Actions
	restore	Grants a user the access needed to restore data.	changeCustomData changePassword collMod createCollection createIndex createUser dropCollection dropRole dropUser getParameter grantRole find insert listCollections modifyChangeStreams revokeRole remove viewRole viewUser update

Logging and Monitoring in Amazon DocumentDB

Amazon DocumentDB (with MongoDB compatibility) provides a variety of Amazon CloudWatch metrics that you can monitor to determine the health and performance of your Amazon DocumentDB clusters and instances. You can view Amazon DocumentDB metrics using various tools, including the Amazon DocumentDB console, the AWS CLI, the Amazon CloudWatch console, and the CloudWatch API. For more information about monitoring, see [Monitoring Amazon DocumentDB \(p. 401\)](#).

In addition to Amazon CloudWatch metrics, you can use the profiler to log the execution time and details of operations that were performed on your cluster. Profiler is useful for monitoring the slowest operations on your cluster to help you improve individual query performance and overall cluster performance. When enabled, operations are logged to Amazon CloudWatch Logs and you can use CloudWatch Insight to analyze, monitor, and archive your Amazon DocumentDB profiling data. For more information, see [Profiling Amazon DocumentDB Operations \(p. 427\)](#).

Amazon DocumentDB also integrates with AWS CloudTrail, a service that provides a record of actions taken by IAM users, IAM roles, or an AWS service in Amazon DocumentDB (with MongoDB compatibility). CloudTrail captures all AWS CLI API calls for Amazon DocumentDB as events, including calls from the Amazon DocumentDB AWS Management Console and from code calls to the Amazon DocumentDB SDK. For more information, see [Logging Amazon DocumentDB API Calls with AWS CloudTrail \(p. 426\)](#).

With Amazon DocumentDB, you can audit events that were performed in your cluster. Examples of logged events include successful and failed authentication attempts, dropping a collection in a database, or creating an index. By default, auditing is disabled on Amazon DocumentDB and requires that you opt in to this feature. For more information, see [Auditing Amazon DocumentDB Events \(p. 205\)](#).

Updating Your Amazon DocumentDB TLS Certificates

The certificate authority (CA) certificate for Amazon DocumentDB (with MongoDB compatibility) clusters was updated on **March 5, 2020**. If you are using Amazon DocumentDB clusters with Transport Layer Security (TLS) enabled (the default setting) and you have not rotated your client application and server certificates, the following steps are required to mitigate connectivity issues between your application and your Amazon DocumentDB clusters.

- [Step 1: Download the New CA Certificate and Update Your Application \(p. 187\)](#)
- [Step 2: Update the Server Certificate \(p. 188\)](#)

The CA and server certificates were updated as part of standard maintenance and security best practices for Amazon DocumentDB. The previous CA certificate expired on March 5, 2020. Client applications must add the new CA certificates to their trust stores, and existing Amazon DocumentDB instances must be updated to use the new CA certificates before this expiration date.

Updating Your Application and Amazon DocumentDB Cluster

Follow the steps in this section to update your application's CA certificate bundle ([Step 1](#)) and your cluster's server certificates ([Step 2](#)). Before you apply the changes to your production environments, we strongly recommend testing these steps in a development or staging environment.

Note

You must complete Steps 1 and 2 in each AWS Region in which you have Amazon DocumentDB clusters.

Step 1: Download the New CA Certificate and Update Your Application

Download the new CA certificate and update your application to use the new CA certificate to create TLS connections to Amazon DocumentDB. Download the new CA certificate bundle from <https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem>. This operation downloads a file named `rds-combined-ca-bundle.pem`.

Note

If you are accessing the keystore that contains both the old CA certificate (`rds-ca-2015-root.pem`) and the new CA certificate (`rds-ca-2019-root.pem`), verify that the keystore selects CA-2019.

```
wget https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem
```

Next, update your applications to use the new certificate bundle. The new CA bundle contains both the old CA certificate and the new CA certificate (`rds-ca-2019-root.pem`). By having both CA certificates in the new CA bundle, you can update your application and cluster in two steps.

Any downloads of the CA certificate bundle after September 1, 2019 should use the new CA certificate bundle. To verify that your application is using the latest CA certificate bundle, see [How can I be sure that I'm using the newest CA bundle? \(p. 193\)](#) If you're already using the latest CA certificate bundle in your application, you can skip to Step 2.

For examples of using a CA bundle with your application, see [Encrypting Data in Transit \(p. 147\)](#) and [Connecting with TLS Enabled \(p. 471\)](#).

Note

Currently, the MongoDB Go Driver 1.2.1 only accepts one CA server certificate in `sslcertificateauthorityfile`. Please see [Connecting with TLS Enabled \(p. 471\)](#) for connecting to Amazon DocumentDB using Go when TLS is enabled.

Step 2: Update the Server Certificate

After the application has been updated to use the new CA bundle, the next step is to update the server certificate by modifying each instance in an Amazon DocumentDB cluster. To modify instances to use the new server certificate, see the following instructions.

Note

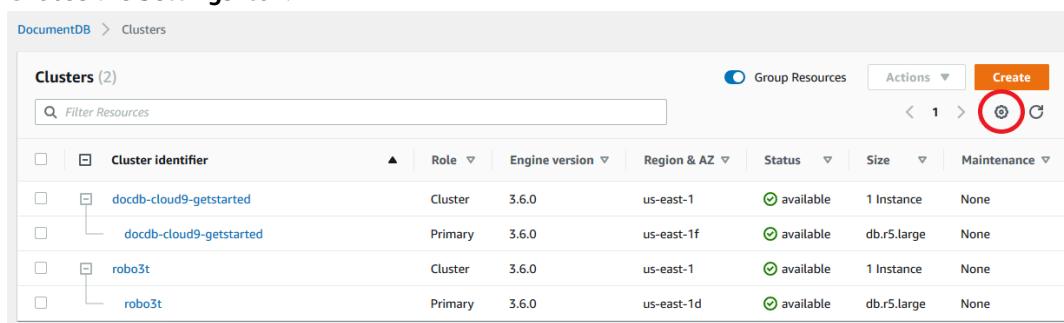
Updating your instances requires a reboot, which might cause service disruption. Before updating the server certificate, ensure that you have completed [Step 1](#).

Using the AWS Management Console

Complete the following steps to identify and rotate the old server certificate for your existing Amazon DocumentDB instances using the AWS Management Console.

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
 2. In the list of Regions in the upper-right corner of the screen, choose the AWS Region in which your clusters reside.
 3. wh
- In the navigation pane on the left side of the console, choose **Clusters**.
4. You may need to identify which instances are still on the old server certificate (`rds-ca-2015`). You can do this in the **Certificate authority** column which is hidden by default. To show the **Certificate authority column**, do the following:

- a. Choose the **Settings** icon.



Clusters (2)							
	Cluster identifier	Role	Engine version	Region & AZ	Status	Size	Maintenance
<input type="checkbox"/>	docdb-cloud9-getstarted	Cluster	3.6.0	us-east-1	available	1 Instance	None
<input type="checkbox"/>	docdb-cloud9-getstarted	Primary	3.6.0	us-east-1f	available	db.r5.large	None
<input type="checkbox"/>	robo3t	Cluster	3.6.0	us-east-1	available	1 Instance	None
<input type="checkbox"/>	robo3t	Primary	3.6.0	us-east-1d	available	db.r5.large	None

- b. Under the list of visible columns, choose the **Certificate authority** column.

- c. Choose **Confirm** to save your changes.
5. Now back in the Clusters navigation box, you'll see the column **Cluster Identifier**. Your instances are listed under clusters, similar to the screenshot below.

The screenshot shows the Amazon DocumentDB console interface. On the left, there's a sidebar with links: Dashboard, Clusters (which is selected and highlighted in orange), Snapshots, Subnet groups, Parameter groups, Events, What's New (with a blue badge showing '16'), Tutorials, and Blogs. The main area is titled 'Clusters (2)' and shows a table with two rows. The first row is for the cluster 'docdb-cloud9-getstarted', which has one primary instance also named 'docdb-cloud9-getstarted'. The second row is for the cluster 'robo3t', which has one primary instance named 'robo3t'. A red circle highlights the primary instance 'docdb-cloud9-getstarted' under the cluster 'docdb-cloud9-getstarted'.

	Cluster identifier	Role
<input type="checkbox"/>	docdb-cloud9-getstarted	Cluster
<input type="checkbox"/>	docdb-cloud9-getstarted	Primary
<input type="checkbox"/>	robo3t	Cluster
<input type="checkbox"/>	robo3t	Primary

6. Check the box to the left of the instance you are interested in.
7. Choose **Actions** and then choose **Modify**.
8. Under **Certificate authority**, select the new server certificate (`rds-ca-2019`) for this instance.
9. You can see a summary of the changes on the next page. Note that there is an extra alert to remind you to ensure that your application is using the latest certificate CA bundle before modifying the instance to avoid causing an interruption in connectivity.
10. You can choose to apply the modification during your next maintenance window or apply immediately. If your intention is to modify the server certificate immediately, use the **Apply Immediately** option.
11. Choose **Modify instance** to complete the update.

Using the AWS CLI

Complete the following steps to identify and rotate the old server certificate for your existing Amazon DocumentDB instances using the AWS CLI.

1. To modify the instances immediately, execute the following command for each instance in the cluster.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier> --ca-certificate-identifier rds-ca-2019 --apply-immediately
```

2. To modify the instances in your clusters to use the new CA certificate during your cluster's next maintenance window, execute the following command for each instance in the cluster.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier> --ca-certificate-identifier rds-ca-2019 --no-apply-immediately
```

Troubleshooting

If you are having issues connecting to your cluster as part of the certificate rotation, we suggest the following:

- **Reboot your instances.** Rotating the new certificate requires that you reboot each of your instances. If you applied the new certificate to one or more instances but did not reboot them, reboot your instances to apply the new certificate. For more information, see [Rebooting an Amazon DocumentDB Instance \(p. 329\)](#).
- **Verify that your clients are using the latest certificate bundle.** See [How can I be sure that I'm using the newest CA bundle? \(p. 193\)](#).
- **Verify that your instances are using the latest certificate.** See [How do I know which of my Amazon DocumentDB instances are using the old/new server certificate? \(p. 190\)](#).
- **Verify that the latest certificate CA is being utilized by your application.** Some drivers, like Java and Go, require extra code to import multiple certificates from a certificate bundle to the trust store. For more information on connecting to Amazon DocumentDB with TLS, see [Connecting Programmatically to Amazon DocumentDB \(p. 469\)](#).
- **Contact support.** If you have questions or issues, contact [AWS Support](#).

Frequently Asked Questions

The following are answers to some common questions about TLS certificates.

What if I have questions or issues?

If you have questions or issues, contact [AWS Support](#).

How do I know whether I'm using TLS to connect to my Amazon DocumentDB cluster?

You can determine whether your cluster is using TLS by examining the `tls` parameter for your cluster's cluster parameter group. If the `tls` parameter is set to `enabled`, you are using the TLS certificate to connect to your cluster. For more information, see [Managing Amazon DocumentDB Cluster Parameter Groups \(p. 355\)](#).

Why are you updating the CA and server certificates?

The Amazon DocumentDB CA and server certificates were updated as part of standard maintenance and security best practices for Amazon DocumentDB. The current CA and server certificates are expired on Thursday, March 5, 2020.

What happens if I don't take any action by the expiration date?

If you are using TLS to connect to your Amazon DocumentDB cluster and you do not make the change by March 5, 2020, your applications that connect via TLS will no longer be able to communicate with the Amazon DocumentDB cluster.

Amazon DocumentDB will not rotate your database certificates automatically before expiration. You must update your applications and clusters to use the new CA certificates before or after the expiration date.

How do I know which of my Amazon DocumentDB instances are using the old/new server certificate?

To identify the Amazon DocumentDB instances that still use the old server certificate, you can use either the Amazon DocumentDB AWS Management Console or the AWS CLI.

Using the AWS Management Console

To identify the instances in your clusters that are using the older certificate

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the list of Regions in the upper-right corner of the screen, choose the AWS Region in which your instances reside.
3. In the navigation pane on the left side of the console, choose **Instances**.
4. The **Certificate authority** column (hidden by default) shows which instances are still on the old server certificate (`rds-ca-2015`) and the new server certificate (`rds-ca-2019`). To show the **Certificate authority column**, do the following:
 - a. Choose the **Settings** icon.
 - b. Under the list of visible columns, choose the **Certificate authority** column.
 - c. Choose **Confirm** to save your changes.

Using the AWS CLI

To identify the instances in your clusters that are using the older server certificate, use the `describe-db-clusters` command with the following .

```
aws docdb describe-db-instances \
  --filters Name=engine,Values=docdb \
  --query 'DBInstances[*].
{CertificateVersion:CACertificateIdentifier,InstanceID:DBInstanceIdentifier}'
```

How do I modify individual instances in my Amazon DocumentDB cluster to update the server certificate?

We recommend that you update server certificates for all instances in a given cluster at the same time. To modify the instances in your cluster, you can use either the console or the AWS CLI.

Note

Updating your instances requires a reboot, which might cause service disruption. Before updating the server certificate, ensure that you have completed [Step 1](#).

Using the AWS Management Console

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the list of Regions in the upper-right corner of the screen, choose the AWS Region in which your clusters reside.
3. In the navigation pane on the left side of the console, choose **Instances**.
4. The **Certificate authority** column (hidden by default) shows which instances are still on the old server certificate (`rds-ca-2015`). To show the **Certificate authority column**, do the following:
 - a. Choose the **Settings** icon.
 - b. Under the list of visible columns, choose the **Certificate authority** column.
 - c. Choose **Confirm** to save your changes.
5. Select an instance to modify.

6. Choose **Actions** and then choose **Modify**.
7. Under **Certificate authority**, select the new server certificate (`rds-ca-2019`) for this instance.
8. You can see a summary of the changes on the next page. Note that there is an extra alert to remind you to ensure that your application is using the latest certificate CA bundle before modifying the instance to avoid causing an interruption in connectivity.
9. You can choose to apply the modification during your next maintenance window or apply immediately.
10. Choose **Modify instance** to complete the update.

Using the AWS CLI

Complete the following steps to identify and rotate the old server certificate for your existing Amazon DocumentDB instances using the AWS CLI.

1. To modify the instances immediately, execute the following command for each instance in the cluster.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier> --ca-certificate-identifier rds-ca-2019 --apply-immediately
```

2. To modify the instances in your clusters to use the new CA certificate during your cluster's next maintenance window, execute the following command for each instance in the cluster.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier> --ca-certificate-identifier rds-ca-2019 --no-apply-immediately
```

What happens if I add a new instance to an existing cluster?

All new instances that are created use the old server certificate and require TLS connections using the old CA certificate. Any new Amazon DocumentDB instances created after January 14, 2020 will default to using the new certificates.

What happens if there is an instance replacement or failover on my cluster?

If there is an instance replacement in your cluster, the new instance that is created continues to use the same server certificate that the instance was previously using. We recommend that you update server certificates for all instances at the same time. If a failover occurs in the cluster, the server certificate on the new primary is used.

If I'm not using TLS to connect to my cluster, do I still need to update each of my instances?

If you are not using TLS to connect to your Amazon DocumentDB clusters, no action is needed.

If I'm not using TLS to connect to my cluster but I plan to in the future, what should I do?

If you created a cluster before November 1, 2019, follow [Step 1](#) and [Step 2](#) in the previous section to ensure that your application is using the updated CA bundle, and that each Amazon DocumentDB instance is using the latest server certificate. If you create a cluster after January 14, 2020, your cluster

will already have the latest server certificate. To verify that your application is using the latest CA bundle, see [If I'm not using TLS to connect to my cluster, do I still need to update each of my instances? \(p. 192\)](#)

Can the deadline be extended beyond March 5, 2020?

If your applications are connecting via TLS, the deadline cannot be extended beyond March 5, 2020.

How can I be sure that I'm using the newest CA bundle?

For compatibility reasons, both old and new CA bundle files are named `rds-combined-ca-bundle.pem`. You can use both the size and the hash of the CA bundle to determine whether the CA bundle is the latest. You can also use tools like `openssl` or `keytool` to inspect the CA bundle. The old CA bundle file is 26016 bytes in size, and the SHA1 hash is `4cd5ba9e145006b17c400d5c778e1965b50172aa`.

To verify that you have the newest bundle, use the following commands.

macOS

Command:

```
ls -l rds-combined-ca-bundle.pem
```

Output:

```
-rw-r--r-- 1 user users 65484 Sep 25 14:49 rds-combined-ca-bundle.pem
```

Command:

```
shasum rds-combined-ca-bundle.pem
```

Output:

```
e12be0c71b499540ac6fe0c36a5050231616c9c4 rds-combined-ca-bundle.pem
```

Amazon Linux

Command:

```
ls -l rds-combined-ca-bundle.pem
```

Output:

```
-rw-rw-r-- 1 ec2-user ec2-user 65484 Sep 25 20:52 rds-combined-ca-bundle.pem
```

Command:

```
sha1sum rds-combined-ca-bundle.pem
```

Output:

```
e12be0c71b499540ac6fe0c36a5050231616c9c4 rds-combined-ca-bundle.pem
```

Windows

Command:

```
dir rds-combined-ca-bundle.pem
```

Output:

```
09/25/2019 02:53 PM      65,484 rds-combined-ca-bundle.pem
```

Command:

```
certutil -hashfile rds-combined-ca-bundle.pem
```

Output:

```
SHA1 hash of rds-combined-ca-bundle.pem:  
e12be0c71b499540ac6fe0c36a5050231616c9c4
```

Why do I see "RDS" in the name of the CA bundle?

For certain management features, such as certificate management, Amazon DocumentDB uses operational technology that is shared with Amazon Relational Database Service (Amazon RDS).

When will the new certificate expire?

The new server certificate will expire on August 22, 2024 GMT.

If I applied the new server certificate, can I revert it back to the old server certificate?

If you need to revert an instance to the old server certificate, we recommend that you do so for all instances in the cluster. You can revert the server certificate for each instance in a cluster by using the AWS Management Console or the AWS CLI.

Using the AWS Management Console

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the list of Regions in the upper-right corner of the screen, choose the AWS Region in which your clusters reside.
3. In the navigation pane on the left side of the console, choose **Instances**.
4. Select an instance to modify. Choose **Actions**, and then choose **Modify**.
5. Under **Certificate authority**, you can select the old server certificate (`rds-ca-2015`).
6. Choose **Continue** to view a summary of your modifications.
7. In this resulting page, you can choose to schedule your modifications to be applied in the next maintenance window or apply your modifications immediately. Make your selection, and choose **Modify instance**.

Note

If you choose to apply your modifications immediately, any changes in the pending modifications queue are also applied. If any of the pending modifications require downtime, choosing this option can cause unexpected downtime.

Using the AWS CLI

```
aws docdb modify-db-instance --db-instance-identifier <db_instance_name> ca-certificate-  
identifier rds-ca-2015 <--apply-immediately | --no-apply-immediately>
```

If you choose `--no-apply-immediately`, the changes will be applied during the cluster's next maintenance window.

If I restore from a snapshot or a point in time restore, will it have the new server certificate?

If you restore a snapshot or perform a point-in-time restore after January 14, 2020, the new cluster that is created will use the new CA certificate.

What if I'm having issues connecting directly to my Amazon DocumentDB cluster from Mac OS X Catalina?

Mac OS X Catalina has updated the requirements for trusted certificates. Trusted certificates must now be valid for 825 days or fewer (see <https://support.apple.com/en-us/HT210176>). Amazon DocumentDB instance certificates are valid for over four years, longer than the Mac OS X maximum. In order to connect directly to an Amazon DocumentDB cluster from a computer running Mac OS X Catalina, you must allow invalid certificates when creating the TLS connection. In this case, invalid certificates mean that the validity period is longer than 825 days. You should understand the risks before allowing invalid certificates when connecting to your Amazon DocumentDB cluster.

To connect to an Amazon DocumentDB cluster from OS X Catalina using the AWS CLI, use the `tlsAllowInvalidCertificates` parameter.

```
mongo --tls --host <hostname> --username <username> --password <password> --port 27017 --  
tlsAllowInvalidCertificates
```

Updating Your Amazon DocumentDB TLS Certificates — GovCloud (US-West)

Note

This information only applies to users in the GovCloud (US-West) region.

The certificate authority (CA) certificate for Amazon DocumentDB (with MongoDB compatibility) clusters will update on **May 18, 2022**. If you are using Amazon DocumentDB clusters with Transport Layer Security (TLS) enabled (the default setting) and you have not rotated your client application and server certificates, the following steps are required to mitigate connectivity issues between your application and your Amazon DocumentDB clusters.

- [Step 1: Download the New CA Certificate and Update Your Application \(p. 196\)](#)
- [Step 2: Update the Server Certificate \(p. 196\)](#)

The CA and server certificates were updated as part of standard maintenance and security best practices for Amazon DocumentDB. The previous CA certificate will expire on May 18, 2022. Client applications must add the new CA certificates to their trust stores, and existing Amazon DocumentDB instances must be updated to use the new CA certificates before this expiration date.

Updating Your Application and Amazon DocumentDB Cluster

Follow the steps in this section to update your application's CA certificate bundle ([Step 1](#)) and your cluster's server certificates ([Step 2](#)). Before you apply the changes to your production environments, we strongly recommend testing these steps in a development or staging environment.

Note

You must complete Steps 1 and 2 in each AWS Region in which you have Amazon DocumentDB clusters.

Step 1: Download the New CA Certificate and Update Your Application

Download the new CA certificate and update your application to use the new CA certificate to create TLS connections to Amazon DocumentDB. Download the new CA certificate bundle from <https://truststore.pki.us-gov-west-1.rds.amazonaws.com/us-gov-west-1/us-gov-west-1-bundle.pem>. This operation downloads a file named us-gov-west-1-bundle.pem.

Note

If you are accessing the keystore that contains both the old CA certificate (`rds-ca-2017-root.pem`) and the new CA certificate (`rds-ca-rsa4096-g1.pem`), verify that the keystore selects CA-RSA4096-G1.

```
wget https://truststore.pki.us-gov-west-1.rds.amazonaws.com/us-gov-west-1/us-gov-west-1-bundle.pem
```

Next, update your applications to use the new certificate bundle. The new CA bundle contains both the old CA certificate and the new CA certificate (`rds-ca-rsa4096-g1.pem`). By having both CA certificates in the new CA bundle, you can update your application and cluster in two steps.

Any downloads of the CA certificate bundle after December 21, 2021 should use the new CA certificate bundle. To verify that your application is using the latest CA certificate bundle, see [How can I be sure that I'm using the newest CA bundle? \(p. 201\)](#) If you're already using the latest CA certificate bundle in your application, you can skip to Step 2.

For examples of using a CA bundle with your application, see [Encrypting Data in Transit \(p. 147\)](#) and [Connecting with TLS Enabled \(p. 471\)](#).

Note

Currently, the MongoDB Go Driver 1.2.1 only accepts one CA server certificate in `sslcertificateauthorityfile`. Please see [Connecting with TLS Enabled \(p. 471\)](#) for connecting to Amazon DocumentDB using Go when TLS is enabled.

Step 2: Update the Server Certificate

After the application has been updated to use the new CA bundle, the next step is to update the server certificate by modifying each instance in an Amazon DocumentDB cluster. To modify instances to use the new server certificate, see the following instructions.

Note

Updating your instances requires a reboot, which might cause service disruption. Before updating the server certificate, ensure that you have completed [Step 1](#).

Using the AWS Management Console

Complete the following steps to identify and rotate the old server certificate for your existing Amazon DocumentDB instances using the AWS Management Console.

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the list of Regions in the upper-right corner of the screen, choose the AWS Region in which your clusters reside.
3. wh

In the navigation pane on the left side of the console, choose **Clusters**.

4. You may need to identify which instances are still on the old server certificate (rds-ca-2017). You can do this in the **Certificate authority** column which is hidden by default. To show the **Certificate authority** column, do the following:
 - a. Choose the **Settings** icon.

The screenshot shows the 'Clusters' page in the Amazon DocumentDB console. At the top right, there is a 'Create' button and a 'Actions' dropdown. Below the header, there is a search bar labeled 'Filter Resources'. The main area displays a table with two columns: 'Cluster identifier' and 'Role'. Under 'Cluster identifier', there are four entries: 'docdb-cloud9-getstarted' (Cluster, Primary), 'robo3t' (Cluster, Primary). The 'Role' column is currently collapsed. A red circle highlights the 'Actions' dropdown menu icon.

Cluster identifier	Role
docdb-cloud9-getstarted	Cluster
docdb-cloud9-getstarted	Primary
robo3t	Cluster
robo3t	Primary

- b. Under the list of visible columns, choose the **Certificate authority** column.
- c. Choose **Confirm** to save your changes.
5. Now back in the Clusters navigation box, you'll see the column **Cluster Identifier**. Your instances are listed under clusters, similar to the screenshot below.

The screenshot shows the 'Clusters' page in the Amazon DocumentDB console. On the left, a sidebar menu has 'Clusters' selected. The main area displays a table with two columns: 'Cluster identifier' and 'Role'. Under 'Cluster identifier', there are four entries: 'docdb-cloud9-getstarted' (Cluster, Primary), 'robo3t' (Cluster, Primary). The 'Role' column is currently collapsed. A red circle highlights the 'Cluster identifier' column header.

Cluster identifier	Role
docdb-cloud9-getstarted	Cluster
docdb-cloud9-getstarted	Primary
robo3t	Cluster
robo3t	Primary

6. Check the box to the left of the instance you are interested in.
7. Choose **Actions** and then choose **Modify**.
8. Under **Certificate authority**, select the new server certificate (rds-ca-rsa4096-g1) for this instance.
9. You can see a summary of the changes on the next page. Note that there is an extra alert to remind you to ensure that your application is using the latest certificate CA bundle before modifying the instance to avoid causing an interruption in connectivity.
10. You can choose to apply the modification during your next maintenance window or apply immediately. If your intention is to modify the server certificate immediately, use the **Apply Immediately** option.

11. Choose **Modify instance** to complete the update.

Using the AWS CLI

Complete the following steps to identify and rotate the old server certificate for your existing Amazon DocumentDB instances using the AWS CLI.

1. To modify the instances immediately, execute the following command for each instance in the cluster.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier> --ca-certificate-identifier rds-ca-rsa4096-g1 --apply-immediately
```

2. To modify the instances in your clusters to use the new CA certificate during your cluster's next maintenance window, execute the following command for each instance in the cluster.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier> --ca-certificate-identifier rds-ca-rsa4096-g1 --no-apply-immediately
```

Troubleshooting

If you are having issues connecting to your cluster as part of the certificate rotation, we suggest the following:

- **Reboot your instances.** Rotating the new certificate requires that you reboot each of your instances. If you applied the new certificate to one or more instances but did not reboot them, reboot your instances to apply the new certificate. For more information, see [Rebooting an Amazon DocumentDB Instance \(p. 329\)](#).
- **Verify that your clients are using the latest certificate bundle.** See [How can I be sure that I'm using the newest CA bundle? \(p. 201\)](#).
- **Verify that your instances are using the latest certificate.** See [How do I know which of my Amazon DocumentDB instances are using the old/new server certificate? \(p. 199\)](#).
- **Verify that the latest certificate CA is being utilized by your application.** Some drivers, like Java and Go, require extra code to import multiple certificates from a certificate bundle to the trust store. For more information on connecting to Amazon DocumentDB with TLS, see [Connecting Programmatically to Amazon DocumentDB \(p. 469\)](#).
- **Contact support.** If you have questions or issues, contact [AWS Support](#).

Frequently Asked Questions

The following are answers to some common questions about TLS certificates.

What if I have questions or issues?

If you have questions or issues, contact [AWS Support](#).

How do I know whether I'm using TLS to connect to my Amazon DocumentDB cluster?

You can determine whether your cluster is using TLS by examining the `tls` parameter for your cluster's cluster parameter group. If the `tls` parameter is set to `enabled`, you are using the TLS certificate to

connect to your cluster. For more information, see [Managing Amazon DocumentDB Cluster Parameter Groups \(p. 355\)](#).

Why are you updating the CA and server certificates?

The Amazon DocumentDB CA and server certificates were updated as part of standard maintenance and security best practices for Amazon DocumentDB. The current CA and server certificates will expire on Wednesday, May 18, 2022.

What happens if I don't take any action by the expiration date?

If you are using TLS to connect to your Amazon DocumentDB cluster and you do not make the change by May 18, 2022, your applications that connect via TLS will no longer be able to communicate with the Amazon DocumentDB cluster.

Amazon DocumentDB will not rotate your database certificates automatically before expiration. You must update your applications and clusters to use the new CA certificates before or after the expiration date.

How do I know which of my Amazon DocumentDB instances are using the old/new server certificate?

To identify the Amazon DocumentDB instances that still use the old server certificate, you can use either the Amazon DocumentDB AWS Management Console or the AWS CLI.

Using the AWS Management Console

To identify the instances in your clusters that are using the older certificate

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the list of Regions in the upper-right corner of the screen, choose the AWS Region in which your instances reside.
3. In the navigation pane on the left side of the console, choose **Instances**.
4. The **Certificate authority** column (hidden by default) shows which instances are still on the old server certificate (`rds-ca-2017`) and the new server certificate (`rds-ca-rsa4096-g1`). To show the **Certificate authority column**, do the following:
 - a. Choose the **Settings** icon.
 - b. Under the list of visible columns, choose the **Certificate authority** column.
 - c. Choose **Confirm** to save your changes.

Using the AWS CLI

To identify the instances in your clusters that are using the older server certificate, use the `describe-db-clusters` command with the following .

```
aws docdb describe-db-instances \
  --filters Name=engine,Values=docdb \
  --query 'DBInstances[*].
{CertificateVersion:CACertificateIdentifier,InstanceID:DBInstanceIdentifier}'
```

How do I modify individual instances in my Amazon DocumentDB cluster to update the server certificate?

We recommend that you update server certificates for all instances in a given cluster at the same time. To modify the instances in your cluster, you can use either the console or the AWS CLI.

Note

Updating your instances requires a reboot, which might cause service disruption. Before updating the server certificate, ensure that you have completed [Step 1](#).

Using the AWS Management Console

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the list of Regions in the upper-right corner of the screen, choose the AWS Region in which your clusters reside.
3. In the navigation pane on the left side of the console, choose **Instances**.
4. The **Certificate authority** column (hidden by default) shows which instances are still on the old server certificate (`rds-ca-2017`). To show the **Certificate authority column**, do the following:
 - a. Choose the **Settings** icon.
 - b. Under the list of visible columns, choose the **Certificate authority** column.
 - c. Choose **Confirm** to save your changes.
5. Select an instance to modify.
6. Choose **Actions** and then choose **Modify**.
7. Under **Certificate authority**, select the new server certificate (`rds-ca-rsa4096-g1`) for this instance.
8. You can see a summary of the changes on the next page. Note that there is an extra alert to remind you to ensure that your application is using the latest certificate CA bundle before modifying the instance to avoid causing an interruption in connectivity.
9. You can choose to apply the modification during your next maintenance window or apply immediately.
10. Choose **Modify instance** to complete the update.

Using the AWS CLI

Complete the following steps to identify and rotate the old server certificate for your existing Amazon DocumentDB instances using the AWS CLI.

1. To modify the instances immediately, execute the following command for each instance in the cluster.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier> --ca-certificate-identifier rds-ca-rsa4096-g1 --apply-immediately
```

2. To modify the instances in your clusters to use the new CA certificate during your cluster's next maintenance window, execute the following command for each instance in the cluster.

```
aws docdb modify-db-instance --db-instance-identifier <yourInstanceIdentifier> --ca-certificate-identifier rds-ca-rsa4096-g1 --no-apply-immediately
```

What happens if I add a new instance to an existing cluster?

All new instances that are created use the old server certificate and require TLS connections using the old CA certificate. Any new Amazon DocumentDB instances created after March 21, 2022 will default to using the new certificates.

What happens if there is an instance replacement or failover on my cluster?

If there is an instance replacement in your cluster, the new instance that is created continues to use the same server certificate that the instance was previously using. We recommend that you update server certificates for all instances at the same time. If a failover occurs in the cluster, the server certificate on the new primary is used.

If I'm not using TLS to connect to my cluster, do I still need to update each of my instances?

If you are not using TLS to connect to your Amazon DocumentDB clusters, no action is needed.

If I'm not using TLS to connect to my cluster but I plan to in the future, what should I do?

If you created a cluster before March 21, 2022, follow [Step 1](#) and [Step 2](#) in the previous section to ensure that your application is using the updated CA bundle, and that each Amazon DocumentDB instance is using the latest server certificate. If you create a cluster after March 21, 2022, your cluster will already have the latest server certificate. To verify that your application is using the latest CA bundle, see [If I'm not using TLS to connect to my cluster, do I still need to update each of my instances? \(p. 201\)](#)

Can the deadline be extended beyond May 18, 2022?

If your applications are connecting via TLS, the deadline cannot be extended beyond May 18, 2022.

How can I be sure that I'm using the newest CA bundle?

For compatibility reasons, both old and new CA bundle files are named `us-gov-west-1-bundle.pem`. You can also use tools like `openssl` or `keytool` to inspect the CA bundle.

Why do I see "RDS" in the name of the CA bundle?

For certain management features, such as certificate management, Amazon DocumentDB uses operational technology that is shared with Amazon Relational Database Service (Amazon RDS).

If I applied the new server certificate, can I revert it back to the old server certificate?

If you need to revert an instance to the old server certificate, we recommend that you do so for all instances in the cluster. You can revert the server certificate for each instance in a cluster by using the AWS Management Console or the AWS CLI.

Using the AWS Management Console

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.

2. In the list of Regions in the upper-right corner of the screen, choose the AWS Region in which your clusters reside.
3. In the navigation pane on the left side of the console, choose **Instances**.
4. Select an instance to modify. Choose **Actions**, and then choose **Modify**.
5. Under **Certificate authority**, you can select the old server certificate (`rds-ca-2017`).
6. Choose **Continue** to view a summary of your modifications.
7. In this resulting page, you can choose to schedule your modifications to be applied in the next maintenance window or apply your modifications immediately. Make your selection, and choose **Modify instance**.

Note

If you choose to apply your modifications immediately, any changes in the pending modifications queue are also applied. If any of the pending modifications require downtime, choosing this option can cause unexpected downtime.

Using the AWS CLI

```
aws docdb modify-db-instance --db-instance-identifier <db_instance_name> ca-certificate-  
identifier rds-ca-2017 <--apply-immediately | --no-apply-immediately>
```

If you choose `--no-apply-immediately`, the changes will be applied during the cluster's next maintenance window.

If I restore from a snapshot or a point in time restore, will it have the new server certificate ?

If you restore a snapshot or perform a point-in-time restore after March 21, 2022, the new cluster that is created will use the new CA certificate.

What if I'm having issues connecting directly to my Amazon DocumentDB cluster from Mac OS X Catalina?

Mac OS X Catalina has updated the requirements for trusted certificates. Trusted certificates must now be valid for 825 days or fewer (see <https://support.apple.com/en-us/HT210176>). Amazon DocumentDB instance certificates are valid for over four years, longer than the Mac OS X maximum. In order to connect directly to an Amazon DocumentDB cluster from a computer running Mac OS X Catalina, you must allow invalid certificates when creating the TLS connection. In this case, invalid certificates mean that the validity period is longer than 825 days. You should understand the risks before allowing invalid certificates when connecting to your Amazon DocumentDB cluster.

To connect to an Amazon DocumentDB cluster from OS X Catalina using the AWS CLI, use the `tlsAllowInvalidCertificates` parameter.

```
mongo --tls --host <hostname> --username <username> --password <password> --port 27017 --  
tlsAllowInvalidCertificates
```

Compliance Validation in Amazon DocumentDB

The security and compliance of Amazon DocumentDB (with MongoDB compatibility) is assessed by third-party auditors as part of multiple AWS compliance programs, including the following:

- System and Organization Controls (SOC) 1, 2, and 3. For more information, see [SOC](#).

- Payment Card Industry Data Security Standard (PCI DSS). For more information, see [PCI DSS](#).
- ISO 9001, 27001, 27017, and 27018. For more information, see [ISO Certified](#).
- Health Insurance Portability and Accountability Act Business Associate Agreement (HIPAA BAA). For more information, see [HIPAA Compliance](#)

AWS provides a frequently updated list of AWS services in scope of specific compliance programs at [AWS Services in Scope by Compliance Program](#).

Third-party audit reports are available for you to download using AWS Artifact. For more information, see [Downloading Reports in AWS Artifact](#).

For more information about AWS compliance programs, see [AWS Compliance Programs](#).

Your compliance responsibility when using Amazon DocumentDB is determined by the sensitivity of your data, your organization's compliance objectives, and applicable laws and regulations. If your use of Amazon DocumentDB is subject to compliance with standards like HIPAA or PCI, AWS provides resources to help:

- [AWS Compliance Resources](#) – A collection of workbooks and guides that might apply to your industry and location.
- [Security and Compliance Quick Start Guides](#) – Deployment guides that discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- [AWS Config](#) – A service that assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- [AWS Security Hub](#) – A comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.
- [Architecting for HIPAA Security and Compliance Whitepaper](#) – A whitepaper that describes how companies can use AWS to create HIPAA-compliant applications.

Resilience in Amazon DocumentDB

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between Availability Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

An Amazon DocumentDB cluster can only be created in an Amazon VPC that has at least two subnets in at least two Availability Zones. By distributing your cluster instances across at least two Availability Zones, Amazon DocumentDB helps ensure that there are instances available in your cluster in the unlikely event of an Availability Zone failure. The cluster volume for your Amazon DocumentDB cluster always spans three Availability Zones to provide durable storage with less possibility of data loss.

For more information about AWS Regions and Availability Zones, see [AWS Global Infrastructure](#).

In addition to the AWS global infrastructure, Amazon DocumentDB offers several features to help support your data resiliency and backup needs.

Fault-tolerant and self-healing storage

Each 10 GB portion of your storage volume is replicated six ways, across three Availability Zones. Amazon DocumentDB uses fault-tolerant storage that transparently handles the loss of up to two copies of data without affecting database write availability, and up to three copies without

affecting read availability. Amazon DocumentDB storage is also self-healing; data blocks and disks are continuously scanned for errors and replaced automatically.

Manual backups and restore

Amazon DocumentDB provides the capability to create full backups of your cluster for long-term retention and recovery. For more information, see [Backing Up and Restoring in Amazon DocumentDB \(p. 215\)](#).

Point-in-time recovery

Point-in-time recovery helps protect your Amazon DocumentDB clusters from accidental write or delete operations. With point-in-time recovery, you don't have to worry about creating, maintaining, or scheduling on-demand backups. For more information, see [Restoring to a Point in Time \(p. 241\)](#).

Infrastructure Security in Amazon DocumentDB

As a managed service, Amazon DocumentDB is protected by the AWS global network security procedures that are described in the [Amazon Web Services: Overview of Security Processes](#) whitepaper.

You use AWS published API calls to access Amazon DocumentDB through the network. Clients must support TLS (Transport Layer Security) 1.0. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes. Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the [AWS Security Token Service](#) (AWS STS) to generate temporary security credentials to sign requests.

You can call these API operations from any network location, but Amazon DocumentDB does support resource-based access policies, which can include restrictions based on the source IP address. You can also use Amazon DocumentDB policies to control access from specific Amazon Virtual Private Cloud (Amazon VPC) endpoints or specific VPCs. Effectively, this isolates network access to a given Amazon DocumentDB resource from only the specific VPC within the AWS network.

Security Best Practices for Amazon DocumentDB

For security best practices, you must use AWS Identity and Access Management (IAM) accounts to control access to Amazon DocumentDB API operations, especially operations that create, modify, or delete Amazon DocumentDB resources. Such resources include clusters, security groups, and parameter groups. You must also use IAM to control actions that perform common administrative actions such as backing up restoring clusters. When creating IAM roles, employ the principle of least privilege.

- Enforce least privilege with [role-based access control \(p. 169\)](#).
- Assign an individual IAM account to each person who manages Amazon DocumentDB resources. Do not use the AWS account root user to manage Amazon DocumentDB resources. Create an IAM user for everyone, including yourself.
- Grant each user the minimum set of permissions that are required to perform their duties.
- Use IAM groups to effectively manage permissions for multiple users. For more information about IAM, see the [IAM User Guide](#). For information about IAM best practices, see [IAM Best Practices](#).
- Regularly rotate your IAM credentials.
- Configure AWS Secrets Manager to automatically rotate the secrets for Amazon DocumentDB. For more information, see [Rotating Your AWS Secrets Manager Secrets](#) and [Rotating Secrets for Amazon DocumentDB](#) in the [AWS Secrets Manager User Guide](#).
- Use Transport Layer Security (TLS) and encryption at rest to encrypt your data.

Auditing Amazon DocumentDB Events

With Amazon DocumentDB (with MongoDB compatibility), you can audit events that were performed in your cluster. Examples of logged events include successful and failed authentication attempts, dropping a collection in a database, or creating an index. By default, auditing is disabled on Amazon DocumentDB and requires that you opt in to use this feature.

When auditing is enabled, Amazon DocumentDB records Data Definition Language (DDL), Data Manipulation Language (DML), authentication, authorization, and user management events to Amazon CloudWatch Logs. When auditing is enabled, Amazon DocumentDB exports your cluster's auditing records (JSON documents) to Amazon CloudWatch Logs. You can use Amazon CloudWatch Logs to analyze, monitor, and archive your Amazon DocumentDB auditing events.

Although Amazon DocumentDB does not charge an additional cost to enable auditing, you are charged standard rates for the usage of CloudWatch Logs. For information about CloudWatch Logs pricing, see [Amazon CloudWatch pricing](#).

The Amazon DocumentDB auditing feature is distinctly different from the service resource usage that is monitored with AWS CloudTrail. CloudTrail records operations that are performed with the AWS Command Line Interface (AWS CLI) or AWS Management Console on resources like clusters, instances, parameter groups, and snapshots. Auditing of AWS resources with CloudTrail is on by default and cannot be disabled. The Amazon DocumentDB auditing feature is an opt-in feature. It records operations that take place within your cluster on objects, such as databases, collections, indexes, and users.

Topics

- [Supported Events \(p. 205\)](#)
- [Enabling Auditing \(p. 208\)](#)
- [Disabling Auditing \(p. 212\)](#)
- [Accessing Your Audit Events \(p. 214\)](#)

Supported Events

Amazon DocumentDB auditing supports the following event categories:

- **Data Definition Language (DDL)** - includes database management operations, connections, user management, and authorization.
- **Data Manipulation Language read events (DML reads)** - includes `find()` and the various aggregation operators, arithmetic operators, boolean operators, and other read query operators.
- **Data Manipulation Language write events (DML writes)** - includes `insert()`, `update()`, `delete()`, and `bulkWrite()` operators

The event types are as follows.

Event Type	Category	Description
authCheck	Authorization	Unauthorized attempts to perform an operation.
authenticate	Connection	Successful or failed authentication attempts on a new connection.

Event Type	Category	Description
createDatabase	DDL	Creation of a new database.
createCollection	DDL	Creation of a new collection within a database.
createIndex	DDL	Creation of a new index within a collection.
dropCollection	DDL	Dropping of a collection within a database.
dropDatabase	DDL	Dropping of a database.
dropIndex	DDL	Dropping of an index within a collection.
modifyChangeStreams	DDL	Change stream was created.
renameCollection	DDL	Renaming of a collection within a database.
createRole	Role Management	Creating a role.
dropAllRolesFromDatabase	Role Management	Dropping all roles within a database.
dropRole	Role Management	Dropping a role.
grantPrivilegesToRole	Role Management	Granting privileges to a role.
grantRolesToRole	Role Management	Granting roles to a user-defined role.
revokePrivilegesFromRole	Role Management	Revoking privileges from a role.
revokeRolesFromRole	Role Management	Revoking roles from a user-defined role.
updateRole	Role Management	Updating a role.
createUser	User Management	Creation of a new user.
dropAllUsersFromDatabase	User Management	Dropping of all users within a database.
dropUser	User Management	Dropping of an existing user.
grantRolesToUser	User Management	Granting roles to a user.
revokeRolesFromUser	User Management	Revoking roles from a user.

Event Type	Category	Description
updateUser	UserManagement	Updating of an existing user.
insert	DML write	Inserts a document or documents into a collection.
delete	DML write	Deletes a document or documents from a collection.
update	DML write	Modifies an existing document or documents in a collection.
bulkWrite	DML write	Performs multiple write operations with controls for order of execution.
count	DML read	Returns the count of documents that would match a find() query for the collection or view.
countDocuments	DML read	Returns the count of documents that match the query for a collection or view.
find	DML read	Selects documents in a collection or view and returns a cursor to the selected documents.
findAndModify	DML read and DML write	Modifies and returns a single document.
findOneAndDelete	DML read and DML write	Deletes a single document based on the filter and sort criteria, returning the deleted document.
findOneAndReplace	DML read and DML write	Replaces a single document based on the specified filter.
findOneAndUpdate	DML read and DML write	Updates a single document based on the filter and sort criteria.
aggregate	DML read and DML write	Supports APIs in the aggregation pipeline.

Event Type	Category	Description
distinct	DML read	Finds the distinct values for a specified field across a single collection or view and returns the results in an array.

Note

TTL delete events are not audited at this time.

Enabling Auditing

Enabling auditing on a cluster is a two-step process. Ensure that both steps are completed, or audit logs will not be sent to CloudWatch Logs.

Step 1. Enable the audit_logs cluster parameter

To enable auditing, you need to modify the audit_logs parameter in the parameter group. audit_logs is a comma-delimited list of events to log. Events must be specified in lowercase and there should be no white space between the list elements.

You can set the following values for the parameter group:

Value	Description
ddl	Setting this will enable auditing for DDL events such as createDatabase, dropDatabase, createCollection, dropCollection, createIndex, dropIndex, authCheck, authenticate, createUser, dropUser, grantRolesToUser, revokeRolesFromUser, updateUser, and dropAllUsersFromDatabase
dml_read	Setting this will enable auditing for DML read events such as find, sort, count, distinct, group, projecta, unwind, geoNear, geoIntersects, geoWithin and other MongoDB read query operators.
dml_write	Setting this will enable auditing for DML write events such as insert(),

Value	Description
	update(), delete(), and bulkWrite()
all	Setting this will enable auditing for your database events, such as read queries, write queries, database actions and administrator actions.
none	Setting this will disable auditing
enabled (legacy)	This is a legacy parameter setting that is equivalent to 'ddl'. Setting this will enable auditing for DDL events such as createDatabase, dropDatabase, createCollection, dropCollection, createIndex, dropIndex, authCheck, authenticate, createUser, dropUser, grantRolesToUser, revokeRolesFromUser, updateUser, and dropAllUsersFromDatabase. We do not recommend using this setting because it is a legacy setting.
disabled (legacy)	This is a legacy parameter setting that is equivalent to 'none'. We do not recommend using this setting because it is a legacy setting.

You can also use the above mentioned values in combinations.

Value	Description
ddl, dml_read	Setting this will enable auditing for DDL events and DML read events.
ddl, dml_write	Setting this will enable auditing for DDL events and DML write

Value	Description
dml_read, dml_write	Setting this will enable auditing for all DML events

Note

You cannot modify a default parameter group.

For more information, see the following:

- [Creating Amazon DocumentDB Cluster Parameter Groups \(p. 360\)](#)

After creating a custom parameter group, modify it by changing the audit_logs parameter value to enabled.

- [Modifying Amazon DocumentDB Cluster Parameter Groups \(p. 362\)](#)

Step 2. Enable Amazon CloudWatch Logs Export

When the value of the audit_logs cluster parameter is enabled, you must also enable Amazon DocumentDB to export logs to Amazon CloudWatch. If you omit either of these steps, audit logs will not be sent to CloudWatch.

When creating a cluster, performing a point-in-time-restore, or restoring a snapshot, you can enable CloudWatch Logs by following these steps.

Using the AWS Management Console

To enable Amazon DocumentDB exporting logs to CloudWatch using the console, see the following topics:

- **When creating a cluster** — In [Creating a Cluster and Primary Instance Using the AWS Management Console \(p. 275\)](#), see **Create a Cluster: Additional Configurations** (step 5, **Log exports**)
- **When modifying an existing cluster** — [Using the AWS Management Console \(p. 287\)](#).
- **When performing a cluster snapshot restore** — [Restore from a Cluster Snapshot Using the AWS Management Console \(p. 237\)](#), step 9.
- **When performing a point-in-time restore** — [Restore to a Point in Time Using the AWS Management Console \(p. 241\)](#) (step 9).

Using the AWS CLI

To enable audit logs when creating a new cluster

The following code creates the cluster sample-cluster and enables CloudWatch audit logs.

Example

For Linux, macOS, or 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
```

For 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
```

To enable audit logs when modifying an existing cluster

The following code modifies the cluster *sample-cluster* and enables CloudWatch audit logs.

Example

For Linux, macOS, or Unix:

```
aws docdb modify-db-cluster \
--db-cluster-identifier sample-cluster \
--cloudwatch-logs-export-configuration '{"EnableLogTypes":["audit"]}'
```

For Windows:

```
aws docdb modify-db-cluster ^
--db-cluster-identifier sample-cluster ^
--cloudwatch-logs-export-configuration '{"EnableLogTypes":["audit"]}'
```

Output from these operations looks something like the following (JSON format).

```
{
    "DBCluster": {
        "HostedZoneId": "ZNKXH85TT8WVW",
        "StorageEncrypted": false,
        "DBClusterParameterGroup": "default.docdb4.0",
        "MasterUsername": "<user-name>",
        "BackupRetentionPeriod": 1,
        "Port": 27017,
        "VpcSecurityGroups": [
            {
                "Status": "active",
                "VpcSecurityGroupId": "sg-77186e0d"
            }
        ],
        "DBClusterArn": "arn:aws:rds:us-east-1:900083794985:cluster:sample-cluster",
        "Status": "creating",
        "Engine": "docdb",
        "EngineVersion": "4.0.0",
        "MultiAZ": false,
        "AvailabilityZones": [
            "us-east-1a",
            "us-east-1c",
            "us-east-1f"
        ],
        "ProcessorFeatures": []
    }
}
```

```
        "DBSubnetGroup": "default",
        "DBClusterMembers": [],
        "ReaderEndpoint": "sample-cluster.cluster-ro-corcj0zrlsfc.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-Y0S52CUXGDTNKDQ7DH72I4LED4",
        "Endpoint": "sample-cluster.cluster-corcj0zrlsfc.us-east-1.docdb.amazonaws.com",
        "PreferredBackupWindow": "07:16-07:46",
        "DBClusterIdentifier": "sample-cluster"
    }
}
```

Disabling Auditing

You can disable auditing by disabling CloudWatch Logs export and disabling the audit_logs parameter.

Disabling CloudWatch Logs Export

You can disable exporting audit logs using either the AWS Management Console or the AWS CLI.

Using the AWS Management Console

The following procedure uses the AWS Management Console to disable Amazon DocumentDB exporting logs to CloudWatch.

To disable audit logs

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Clusters**. Then choose the button to the left of the name of the cluster for which you want to disable exporting logs.
3. Choose **Actions**, and then choose **Modify**.
4. Scroll down to the **Log exports** section and choose **Disabled**.
5. Choose **Continue**.
6. Review your changes, and then choose when you want this change applied to your cluster.
 - **Apply during the next scheduled maintenance window**
 - **Apply immediately**
7. Choose **Modify cluster**.

Using the AWS CLI

The following code modifies the cluster sample-cluster and disables CloudWatch audit logs.

Example

For Linux, macOS, or Unix:

```
aws docdb modify-db-cluster \
```

```
--db-cluster-identifier sample-cluster \
--cloudwatch-logs-export-configuration '{"DisableLogTypes":["audit"]}'
```

For Windows:

```
aws docdb modify-db-cluster ^
--db-cluster-identifier sample-cluster ^
--cloudwatch-logs-export-configuration '{"DisableLogTypes":["audit"]}'
```

Output from this operation looks something like the following (JSON format).

```
{
  "DBCluster": {
    "DBClusterParameterGroup": "default.docdb4.0",
    "HostedZoneId": "ZNKXH85TT8WWV",
    "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-corcjoczrlsfc.us-east-1.docdb.amazonaws.com",
    "ReaderEndpoint": "sample-cluster2.cluster-ro-corcjoczrlsfc.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-Y0S52CUXGDTNKDQ7DH72I4LED4",
    "StorageEncrypted": false,
    "PreferredMaintenanceWindow": "wed:03:08-wed:03:38",
    "DBClusterMembers": [],
    "VpcSecurityGroups": [
      {
        "Status": "active",
        "VpcSecurityGroupId": "sg-77186e0d"
      }
    ]
  }
}
```

Disabling the audit_logs Parameter

To disable the audit_logs parameter for your cluster, you can modify the cluster so that it uses a parameter group where the audit_logs parameter value is disabled. Or you can modify the audit_logs parameter value in the cluster's parameter group so that it is disabled.

For more information, see the following topics:

- [Modifying an Amazon DocumentDB Cluster \(p. 287\)](#)
- [Modifying Amazon DocumentDB Cluster Parameter Groups \(p. 362\)](#)

Accessing Your Audit Events

Use following steps to access your audit events on Amazon CloudWatch.

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. Make sure that you are in the same Region as your Amazon DocumentDB cluster.
3. In the navigation pane, choose **Logs**.
4. To find the audit logs for your cluster, from the list locate and choose **/aws/docdb/*yourClusterName*/audit**.

The auditing events for each of your instances are available under each of the respective instance names.

Backing Up and Restoring in Amazon DocumentDB

Amazon DocumentDB (with MongoDB compatibility) continuously backs up your data to Amazon Simple Storage Service (Amazon S3) for 1–35 days so that you can quickly restore to any point within the backup retention period. Amazon DocumentDB also takes automatic snapshots of your data as part of this continuous backup process.

Note

These are service-managed Amazon S3 buckets and you will not have access to the backup files. If you want to control your own backups, follow the instructions on [Dumping, Restoring, Importing, and Exporting Data](#).

You can also retain backup data beyond the backup retention period by creating a manual snapshot of your cluster's data. The backup process does not impact your cluster's performance.

This section discusses the use cases for the backup capabilities in Amazon DocumentDB and shows you how to manage backups for your Amazon DocumentDB clusters.

Topics

- [Back Up and Restore: Concepts \(p. 215\)](#)
- [Understanding Backup Storage Usage \(p. 217\)](#)
- [Dumping, Restoring, Importing, and Exporting Data \(p. 218\)](#)
- [Cluster Snapshot Considerations \(p. 221\)](#)
- [Comparing Automatic and Manual Snapshots \(p. 223\)](#)
- [Creating a Manual Cluster Snapshot \(p. 224\)](#)
- [Copying Amazon DocumentDB Cluster Snapshots \(p. 226\)](#)
- [Sharing Amazon DocumentDB Cluster Snapshots \(p. 232\)](#)
- [Restoring from a Cluster Snapshot \(p. 236\)](#)
- [Restoring to a Point in Time \(p. 241\)](#)
- [Deleting a Cluster Snapshot \(p. 245\)](#)

Back Up and Restore: Concepts

Noun	Description	APIs (Verbs)
Backup retention period	A period of time between 1 and 35 days for which you can perform a point-in-time restore.	<code>create-db-cluster</code> <code>modify-db-cluster</code> <code>restore-db-cluster-to-point-in-time</code>
Amazon DocumentDB storage volume	Highly available and highly	<code>create-db-cluster</code> <code>delete-db-cluster</code>

Noun	Description	APIs (Verbs)
	durable storage volume that replicates data six ways across three Availability Zones. An Amazon DocumentDB cluster is highly durable regardless of the number of instances in the cluster.	
Backup window	Period of time in the day in which automatic snapshots are taken.	<code>create-db-cluster</code> <code>describe-db-cluster</code> <code>modify-db-cluster</code>
Automatic snapshot	Daily snapshots that are full backups of cluster and are automatically created by the continuous backup process in Amazon DocumentDB.	<code>restore-db-cluster-from-snapshot</code> <code>describe-db-cluster-snapshot-attributes</code> <code>describe-db-cluster-snapshots</code>
Manual snapshot	Snapshots you create manually to retain full backups of a cluster beyond the backup period.	<code>create-db-cluster-snapshot</code> <code>copy-db-cluster-snapshot</code> <code>delete-db-cluster-snapshot</code> <code>describe-db-cluster-snapshot-attributes</code> <code>describe-db-cluster-snapshots</code> <code>modify-db-cluster-snapshot-attribute</code>

Understanding Backup Storage Usage

Amazon DocumentDB backup storage consists of continuous backups within the backup retention period and manual snapshots outside the retention period. To control your backup storage usage, you can reduce the backup retention interval, remove old manual snapshots when they are no longer needed, or both. For general information about Amazon DocumentDB backups, see [Backing Up and Restoring in Amazon DocumentDB \(p. 215\)](#). For pricing information about Amazon DocumentDB backup storage, see [Amazon DocumentDB Pricing](#).

To control your costs, you can monitor the amount of storage consumed by continuous backups and manual snapshots that persist beyond the retention period. Then you can reduce the backup retention interval and remove manual snapshots when they are no longer needed.

You can use the Amazon CloudWatch metrics `TotalBackupStorageBilled`, `SnapshotStorageUsed`, and `BackupRetentionPeriodStorageUsed` to review and monitor the amount of storage used by your Amazon DocumentDB backups, as follows:

- `BackupRetentionPeriodStorageUsed` represents the amount of backup storage used for storing continuous backups at the current time. This metric value depends on the size of the cluster volume and the number of changes you make during the retention period. However, for billing purposes the metric does not exceed the cumulative cluster volume size during the retention period. For example, if your cluster size is 100 GiB and your retention period is two days, the maximum value for `BackupRetentionPeriodStorageUsed` is 200 GiB (100 GiB + 100 GiB).
- `SnapshotStorageUsed` represents the amount of backup storage used for storing manual snapshots beyond the backup retention period. Manual snapshots taken within the retention period do not count against your backup storage. Similarly, automatic snapshots do not count against your backup storage. The size of each snapshot is the size of the cluster volume at the time you take the snapshot. The `SnapshotStorageUsed` value depends on the number of snapshots you keep and the size of each snapshot. For example, suppose that you have one snapshot outside the retention period and cluster volume size was 100 GiB when that snapshot was taken. The amount of `SnapshotStorageUsed` is 100 GiB.
- `TotalBackupStorageBilled` represents the sum of `BackupRetentionPeriodStorageUsed` and `SnapshotStorageUsed`, minus an amount of free backup storage equal to the size of cluster volume for one day. For example, if your cluster size is 100 GiB, you have one retention day, and you have one snapshot outside the retention period, the `TotalBackupStorageBilled` is 100 GiB (100 GiB + 100 GiB - 100 GiB).
- These metrics are computed independently for each Amazon DocumentDB cluster.

You can monitor your Amazon DocumentDB clusters and build reports using CloudWatch metrics through the [CloudWatch console](#). For more information about how to use CloudWatch metrics, see [Monitoring Amazon DocumentDB \(p. 401\)](#).

Dumping, Restoring, Importing, and Exporting Data

You can use the `mongodump`, `mongorestore`, `mongoexport`, and `mongoimport` utilities to move data in and out of your Amazon DocumentDB cluster. This section discusses the purpose of each of these tools and configurations to help you achieve better performance.

Topics

- [mongodump \(p. 218\)](#)
- [mongorestore \(p. 218\)](#)
- [mongoexport \(p. 219\)](#)
- [mongoimport \(p. 219\)](#)
- [Tutorial \(p. 219\)](#)

[mongodump](#)

The `mongodump` utility creates a binary (BSON) backup of a MongoDB database. The `mongodump` tool is the preferred method of dumping data from your source MongoDB deployment when looking to restore it into your Amazon DocumentDB cluster due to the size efficiencies achieved by storing the data in a binary format.

Depending on the resources available on the instance or machine you are using to perform the command, you can speed up your `mongodump` by increasing the number of parallel connections dumped from the default 1 using the `--numParallelCollections` option. A good rule of thumb is to start with one worker per vCPU on your Amazon DocumentDB cluster's primary instance.

Example Usage

The following is an example usage of the `mongodump` utility in the Amazon DocumentDB cluster, `sample-cluster`.

```
mongodump --ssl \
    --host="sample-cluster.node.us-east-1.docdb.amazonaws.com:27017" \
    --collection=sample-collection \
    --db=sample-database \
    --out=sample-output-file \
    --numParallelCollections 4 \
    --username=sample-user \
    --password=abc0123 \
    --sslCAFile rds-combined-ca-bundle.pem
```

[mongorestore](#)

The `mongorestore` utility enables you to restore a binary (BSON) backup of a database that was created with the `mongodump` utility. You can improve restore performance by increasing the number of workers for each collection during the restore with the `--numInsertionWorkersPerCollection` option (the default is 1). A good rule of thumb is to start with one worker per vCPU on your Amazon DocumentDB cluster's primary instance.

Example Usage

The following is an example usage of the `mongorestore` utility in the Amazon DocumentDB cluster, `sample-cluster`.

```
mongorestore --ssl \
--host="sample-cluster.node.us-east-1.docdb.amazonaws.com:27017" \
--username=sample-user \
--password=abc0123 \
--sslCAFile rds-combined-ca-bundle.pem <fileToBeRestored>
```

mongoexport

The mongoexport tool exports data in Amazon DocumentDB to JSON, CSV, or TSV file formats. The mongoexport tool is the preferred method of exporting data that needs to be human or machine readable.

Note

mongoexport does not directly support parallel exports. However, it is possible to increase performance by executing multiple mongoexport jobs concurrently for different collections.

Example Usage

The following is an example usage of the mongoexport tool in the Amazon DocumentDB cluster, sample-cluster.

```
mongoexport --ssl \
--host="sample-cluster.node.us-east-1.docdb.amazonaws.com:27017" \
--collection=sample-collection \
--db=sample-database \
--out=sample-output-file \
--username=sample-user \
--password=abc0123 \
--sslCAFile rds-combined-ca-bundle.pem
```

mongoimport

The mongoimport tool imports the contents of JSON, CSV, or TSV files into an Amazon DocumentDB cluster. You can use the --numInsertionWorkers parameter to parallelize and speed up the import (the default is 1).

Example Usage

The following is an example usage of the mongoimport tool in the Amazon DocumentDB cluster, sample-cluster.

```
mongoimport --ssl \
--host="sample-cluster.node.us-east-1.docdb.amazonaws.com:27017" \
--collection=sample-collection \
--db=sample-database \
--file=<yourFile> \
--numInsertionWorkers 4 \
--username=sample-user \
--password=abc0123 \
--sslCAFile rds-combined-ca-bundle.pem
```

Tutorial

The following tutorial describes how to use the mongodump, mongorestore, mongoexport, and mongoimport utilities to move data in and out of an Amazon DocumentDB cluster.

1. **Prerequisites** — Before you begin, ensure that your Amazon DocumentDB cluster is provisioned and that you have access to an Amazon EC2 instance in the same VPC as your cluster. For more information, see [Connect Using Amazon EC2 \(p. 509\)](#).

To be able to use the mongo utility tools, you must have the mongodb-org-tools package installed in your EC2 instance, as follows.

```
sudo yum install mongodb-org-tools-4.0.18
```

Because Amazon DocumentDB uses Transport Layer Security (TLS) encryption by default, you must also download the Amazon RDS certificate authority (CA) file to use the mongo shell to connect, as follows.

```
wget https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem
```

2. **Download sample data** — For this tutorial, you will download some sample data that contains information about restaurants.

```
wget https://raw.githubusercontent.com/ozlerhakan/mongodb-json-files/master/datasets/restaurant.json
```

3. **Import the sample data into Amazon DocumentDB** — Since the data is in a logical JSON format, you will use the mongoimport utility to import the data into your Amazon DocumentDB cluster.

```
mongoimport --ssl \
--host="tutorialCluster.amazonaws.com:27017" \
--collection=restaurants \
--db=business \
--file=restaurant.json \
--numInsertionWorkers 4 \
--username=<yourUsername> \
--password=<yourPassword> \
--sslCAFile rds-combined-ca-bundle.pem
```

4. **Dump the data with mongodump** — Now that you have data in your Amazon DocumentDB cluster, you can take a binary dump of that data using the mongodump utility.

```
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 rds-combined-ca-bundle.pem
```

5. **Drop the restaurants collection** — Before you restore the restaurants collection in the business database, you have to first drop the collection that already exists in that database, as follows.

```
use business
```

```
db.restaurants.drop()
```

6. **Restore the data with mongorestore** — With the binary dump of the data from Step 3, you can now use the mongorestore utility to restore your data to your Amazon DocumentDB cluster.

```
mongorestore --ssl \
    --host="tutorialCluster.us-east-1.docdb.amazonaws.com:27017" \
    --numParallelCollections 4 \
    --username=<yourUsername> \
    --password=<yourPassword> \
    --sslCAFile rds-combined-ca-bundle.pem restaurantDump.bson
```

7. **Export the data using mongoexport** — To complete the tutorial, export the data from your cluster in the format of a JSON file, no different than the file you imported in Step 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 rds-combined-ca-bundle.pem
```

8. **Validation** — You can validate that the output of Step 5 yields the same result as Step 1 with the following commands.

```
wc -l restaurant.json
```

Output from this command:

```
2548 restaurant.json
```

```
wc -l restaurant2.json
```

Output from this command:

```
2548 restaurant2.json
```

Cluster Snapshot Considerations

Amazon DocumentDB creates daily automatic snapshots of your cluster during your cluster's backup window. Amazon DocumentDB saves the automatic snapshots of your cluster according to the backup retention period that you specify. If necessary, you can recover your cluster to any point in time during the backup retention period. Automatic snapshots don't occur while a copy operation is executing in the same Region for the same cluster.

In addition to automatic cluster snapshots, you can also manually create a cluster snapshot. You can copy both automatic and manual snapshots. For more information, see [Creating a Manual Cluster Snapshot \(p. 224\)](#) and [Copying Amazon DocumentDB Cluster Snapshots \(p. 226\)](#).

Note

Your cluster must be in the *available* state for an automatic snapshot to be taken.

You can't share an Amazon DocumentDB automated cluster snapshot. As a workaround, you can create a manual snapshot by copying the automated snapshot, and then share that copy.

For more information about copying a snapshot, see [Copying Amazon DocumentDB Cluster Snapshots \(p. 226\)](#). For more information about restoring a cluster from a snapshot, see [Restoring from a Cluster Snapshot \(p. 236\)](#).

Backup Storage

Your Amazon DocumentDB backup storage for each AWS Region is composed of the backup storage needed for your backup retention period, which includes automatic and manual cluster snapshots in that Region. The default backup retention period is 1 day. For more information about backup storage pricing, see [Amazon DocumentDB Pricing](#).

When you delete a cluster, all of its automatic snapshots are deleted and cannot be recovered. However, manual snapshots are not deleted when you delete a cluster. If you choose to have Amazon DocumentDB create a final snapshot (manual snapshot) before your cluster is deleted, you can use the final snapshot to recover your cluster.

For more information on snapshots and storage, see [Understanding Backup Storage Usage \(p. 217\)](#).

Backup Window

Automatic snapshots occur daily during the preferred backup window. If the snapshot requires more time than allotted to the backup window, the backup process continues until it finishes, even though the backup window has ended. The backup window can't overlap with the weekly maintenance window for the cluster.

If you don't specify a preferred backup window when you create the cluster, Amazon DocumentDB assigns a default 30-minute backup window. This window is chosen at random from an 8-hour block of time associated with your cluster's Region. You can change your preferred backup window by modifying the cluster. For more information, see [Modifying an Amazon DocumentDB Cluster \(p. 287\)](#).

Region	UTC Time Block
US East (Ohio)	03:00–11:00
US East (N. Virginia)	03:00–11:00
US West (Oregon)	06:00–14:00
Asia Pacific (Mumbai)	17:30–01:30
Asia Pacific (Seoul)	13:00–21:00
Asia Pacific (Singapore)	14:00–22:00
Asia Pacific (Sydney)	12:00–20:00
Asia Pacific (Tokyo)	13:00–21:00
Canada (Central)	22:00–06:00
Europe (Frankfurt)	23:00–07:00
Europe (Ireland)	22:00–06:00
Europe (Paris)	22:00–06:00
AWS GovCloud (US)	22:00–06:00

Backup Retention Period

The backup retention period is the number of days an automatic backup is retained before being automatically deleted. Amazon DocumentDB supports a backup retention period of 1–35 days.

You can set the backup retention period when you create a cluster. If you don't explicitly set the backup retention period, the default backup retention period of 1 day is assigned to your cluster. After you create a cluster, you can modify the backup retention period by modifying the cluster using either the AWS Management Console or the AWS CLI. For more information, see [Modifying an Amazon DocumentDB Cluster \(p. 287\)](#).

Comparing Automatic and Manual Snapshots

The following are key features of Amazon DocumentDB (with MongoDB compatibility) automatic and manual snapshots.

Amazon DocumentDB automatic snapshots have the following key features:

- **Automatic snapshot naming** — Automatic snapshot names follow the pattern `rds:<cluster-name>-yyyy-mm-dd-hh-mm`, with `yyyy-mm-dd-hh-mm` representing the date and time the snapshot was created.
- **Created automatically on a schedule** — When you create or modify a cluster, you can set the *backup retention period* to an integer value from 1 to 35 days. By default, new clusters have a backup retention period of 1 day. The backup retention period defines the number of days that automatic snapshots are kept before being automatically deleted. You can't disable automatic backups on Amazon DocumentDB clusters.

In addition to setting the backup retention period, you also set the *backup window*, the time of day during which automatic snapshots are created.

- **Deleting automatic snapshots** — Automatic snapshots are deleted when you delete the automatic snapshot's cluster. You can't manually delete an automatic snapshot.
- **Incremental** — During the backup retention period, database updates are recorded so that there is an incremental record of changes.
- **Restoring from an automatic snapshot** — You can restore from an automatic snapshot using the AWS Management Console or the AWS CLI. When you restore from a snapshot using the AWS CLI, you must add instances separately after the cluster is *available*.
- **Sharing** — You can't share an Amazon DocumentDB automated cluster snapshot. As a workaround, you can create a manual snapshot by copying the automated snapshot, and then share that copy. For more information about copying a snapshot, see [Copying Amazon DocumentDB Cluster Snapshots \(p. 226\)](#). For more information about restoring a cluster from a snapshot, see [Restoring from a Cluster Snapshot \(p. 236\)](#).
- **You can restore from any point within the backup retention period** — Because database updates are incrementally recorded, you can restore your cluster to any point in time within the backup retention period.

When you restore from an automatic snapshot or from a point-in-time restore using the AWS CLI, you must add instances separately after the cluster is *available*.

Amazon DocumentDB manual snapshots have the following key features:

- **Created on demand** — Amazon DocumentDB manual snapshots are created on demand using the Amazon DocumentDB Management Console or AWS CLI.
- **Deleting a manual snapshot** — A manual snapshot is deleted only when you explicitly delete it using either the Amazon DocumentDB console or AWS CLI. A manual snapshot is not deleted when you delete its cluster.
- **Full backups** — When a manual snapshot is taken, a full backup of your cluster's data is created and stored.

- **Manual snapshot naming** — You specify the manual snapshot name. Amazon DocumentDB does not add a datetime stamp to the name, so you must add that information if you want it included in the name.
- **Restoring from a manual snapshot** — You can restore from a manual snapshot using the console or the AWS CLI. When you restore from a snapshot using the AWS CLI, you must add instances separately after the cluster is *available*.
- **Service Quotas** — You are limited to a maximum of 100 manual snapshots per AWS Region.
- **Sharing** — You can share manual cluster snapshots, which can be copied by authorized AWS accounts. You can share encrypted or unencrypted manual snapshots. For more information about copying a snapshot, see [Copying Amazon DocumentDB Cluster Snapshots \(p. 226\)](#).
- **You restore to when the manual snapshot was taken** — When you restore from a manual snapshot, you restore to when the manual snapshot was taken.

When you restore from a snapshot using the AWS CLI, you must add instances separately after the cluster is *available*.

Creating a Manual Cluster Snapshot

You can create a manual snapshot using either the AWS Management Console or AWS CLI. The amount of time it takes to create a snapshot varies with the size of your databases. When you create a snapshot, you must do the following:

1. Identify which cluster to back up.
2. Give your snapshot a name. This allows you to restore from it later.

Create a Cluster Snapshot Using the AWS Management Console

To create a manual snapshot using the AWS Management Console, you can follow either method below.

1. Method 1:

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Snapshots**.

Tip

If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.

3. On the **Snapshots** page, choose **Create**.
4. On the **Create cluster snapshot** page:
 - a. **Cluster identifier** — From the drop-down list of clusters, choose the cluster that you want to create a snapshot of.
 - b. **Snapshot identifier** — Enter a name for your snapshot.

Snapshot naming constraints:

- Length is [1–255] letters, numbers, or hyphens.
- First character must be a letter.
- Cannot end with a hyphen or contain two consecutive hyphens.

- Must be unique for all clusters (across Amazon RDS, Amazon Neptune, and Amazon DocumentDB) per AWS account, per Region.
- c. Choose **Create**.
2. **Method 2:**
 1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
 2. In the navigation pane, choose **Clusters**.
- Tip**
If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.
3. On the **Clusters** page, choose the button to the left of the cluster that you want to snapshot.
 4. From the **Actions** menu, choose **Take snapshot**.
 5. On the **Create cluster snapshot** page:
 - a. **Snapshot identifier** — Enter a name for your snapshot.

Snapshot naming constraints:
 - Length is [1–63] letters, numbers, or hyphens.
 - First character must be a letter.
 - Cannot end with a hyphen or contain two consecutive hyphens.
 - Must be unique for all clusters (across Amazon RDS, Amazon Neptune, and Amazon DocumentDB) per AWS account, per Region.
 - b. Choose **Create**.

Create a Cluster Snapshot Using the AWS CLI

To create a cluster snapshot using the AWS CLI, use the `create-db-cluster-snapshot` operation with the following parameters.

Parameters

- **--db-cluster-identifier** — Required. The name of the cluster that you are taking a snapshot of. This cluster must exist and be *available*.
- **--db-cluster-snapshot-identifier** — Required. The name of the manual snapshot that you are creating.

The following example creates a snapshot named `sample-cluster-snapshot` for a cluster named `sample-cluster`.

For Linux, macOS, or Unix:

```
aws docdb create-db-cluster-snapshot \
--db-cluster-identifier sample-cluster \
--db-cluster-snapshot-identifier sample-cluster-snapshot
```

For Windows:

```
aws docdb create-db-cluster-snapshot ^
--db-cluster-identifier sample-cluster ^
--db-cluster-snapshot-identifier sample-cluster-snapshot
```

Output from this operation looks something like the following.

```
{  
    "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"  
    }  
}
```

Copying Amazon DocumentDB Cluster Snapshots

In Amazon DocumentDB, you can copy manual and automatic snapshots within the same AWS Region or to a different AWS Region within the same account. You can also share snapshots owned by other AWS accounts in the same AWS Region. However, you can't copy a cluster snapshot across AWS Regions and AWS account in a single step. These actions must be performed individually.

As an alternative to copying, you can also share manual snapshots with other AWS accounts. For more information, see [Sharing Amazon DocumentDB Cluster Snapshots \(p. 232\)](#).

Note

Amazon DocumentDB bills you based upon the amount of backup and snapshot data you keep and the period of time that you keep it. For more information about the storage associated with Amazon DocumentDB backups and snapshots, see [Understanding Backup Storage Usage \(p. 217\)](#). For pricing information about Amazon DocumentDB storage, see [Amazon DocumentDB Pricing](#).

Topics

- [Copying Shared Snapshots \(p. 227\)](#)
- [Copying Snapshots Across AWS Regions \(p. 227\)](#)
- [Limitations \(p. 227\)](#)
- [Handling Encryption \(p. 227\)](#)
- [Parameter Group Considerations \(p. 227\)](#)
- [Copying a Cluster Snapshot \(p. 228\)](#)

Copying Shared Snapshots

You can copy snapshots shared to you by other AWS accounts. If you are copying an encrypted snapshot that has been shared from another AWS account, you must have access to the AWS KMS encryption key that was used to encrypt the snapshot.

You can only copy a shared snapshot in the same AWS Region, whether the snapshot is encrypted or not. For more information, see [the section called "Handling Encryption" \(p. 227\)](#).

Copying Snapshots Across AWS Regions

When you copy a snapshot to an AWS Region that is different from the source snapshot's AWS Region, each copy is a full snapshot. A full snapshot copy contains all of the data and metadata required to restore the Amazon DocumentDB cluster.

Depending on the AWS Regions involved and the amount of data to be copied, a cross-region snapshot copy can take hours to complete. In some cases, there might be a large number of cross-region snapshot copy requests from a given source AWS Region. In these cases, Amazon DocumentDB might put new cross-region copy requests from that source AWS Region into a queue until some in-progress copies complete. No progress information is displayed about copy requests while they are in the queue. Progress information is displayed when the copy starts.

Limitations

The following are some limitations when you copy snapshots:

- If you delete a source snapshot before the target snapshot becomes available, the snapshot copy may fail. Verify that the target snapshot has a status of AVAILABLE before you delete a source snapshot.
- You can have up to five snapshot copy requests in progress to a single destination Region per account.
- Depending on the regions involved and the amount of data to be copied, a cross-region snapshot copy can take hours to complete. For more information, see [Copying Snapshots Across AWS Regions \(p. 227\)](#).

Handling Encryption

You can copy a snapshot that has been encrypted using an AWS KMS encryption key. If you copy an encrypted snapshot, the copy of the snapshot must also be encrypted. If you copy an encrypted snapshot within the same AWS Region, you can encrypt the copy with the same AWS KMS encryption key as the original snapshot, or you can specify a different AWS KMS encryption key. If you copy an encrypted snapshot across Regions, you can't use the same AWS KMS encryption key for the copy as used for the source snapshot, because AWS KMS keys are Region-specific. Instead, you must specify an AWS KMS key valid in the destination AWS Region.

The source snapshot remains encrypted throughout the copy process. For more information, see [Data Protection in Amazon DocumentDB \(p. 143\)](#).

Note

For Amazon DocumentDB cluster snapshots, you can't encrypt an unencrypted cluster snapshot when you copy the snapshot.

Parameter Group Considerations

When you copy a snapshot across Regions, the copy doesn't include the parameter group used by the original Amazon DocumentDB cluster. When you restore a snapshot to create a new cluster, that cluster

gets the default parameter group for the AWS Region it is created in. To give the new cluster the same parameters as the original, you must do the following:

1. In the destination AWS Region, [create an Amazon DocumentDB cluster parameter group](#) with the same settings as the original cluster. If one already exists in the new AWS Region, you can use that one.
2. After you restore the snapshot in the destination AWS Region, modify the new Amazon DocumentDB cluster and add the new or existing parameter group from the previous step. For more information, see [Modifying an Amazon DocumentDB Cluster \(p. 287\)](#).

Copying a Cluster Snapshot

You can copy an Amazon DocumentDB cluster using the AWS Management Console or the AWS CLI, as follows.

Copy a Cluster Snapshot Using the AWS Management Console

To make a copy of a cluster snapshot using the AWS Management Console, complete the following steps. This procedure works for copying encrypted or unencrypted cluster snapshots, in the same AWS Region or across Regions.

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Snapshots**, and then choose the button to the left of the snapshot that you want to copy.

Tip

If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.

3. From the **Actions** menu, choose **Copy**.
4. In the resulting **Make Copy of cluster snapshot** page, complete the **Settings** section.
 - a. **Destination Region** — Optional. To copy the cluster snapshot to a different AWS Region, choose that AWS Region for **Destination Region**.
 - b. **New snapshot identifier** — Enter a name for the new snapshot.

Target snapshot naming constraints:

- Cannot be the name of an existing snapshot.
- Length is [1—63] letters, numbers, or hyphens.
- First character must be a letter.
- Cannot end with a hyphen or contain two consecutive hyphens.
- Must be unique for all clusters across Amazon RDS, Neptune, and Amazon DocumentDB per AWS account, per Region.

- c. **Copy tags** — To copy any tags you have on your source snapshot to your snapshot copy, choose **Copy tags**.
5. Complete the **Encryption-at-rest** section.
 - a. **Encryption at rest** — If your snapshot is encrypted, these options are not available to you because you cannot create an unencrypted copy from an encrypted snapshot. If your snapshot is not encrypted, choose one of the following:
 - To encrypt all your cluster's data, choose **Enable encryption-at-rest**. If you choose this option, you must designate a **Master key**.

- To not encrypt your cluster's data, choose **Disable encryption-at-rest**. If you choose this option, your snapshot's copy data will not be encrypted, and you are finished with the encryption section.
- b. **Master key** — From the drop-down list, choose one of the following:
- **(default) aws/rds** — The account number and AWS KMS key ID are listed following this option.
 - **<some-key-name>** — If you created a key, it is listed and available for you to choose.
 - **Enter a key ARN** — In the **ARN** box, enter the Amazon Resource Name (ARN) for your AWS KMS key. The format of the ARN is `arn:aws:kms:<region>:<accountID>:key/<key-id>`.
6. To make a copy of the selected snapshot, choose **Copy snapshot**. Alternatively, you can choose **Cancel** to not make a copy of the snapshot.

Copy an Unencrypted Cluster Snapshot Using the AWS CLI

To make a copy of an unencrypted cluster snapshot using the AWS CLI, use the `copy-db-cluster-snapshot` operation with the following parameters. If you are copying the snapshot to another AWS Region, run the command in the AWS Region to which the snapshot will be copied.

- **--source-db-cluster-snapshot-identifier** — Required. The identifier of the cluster snapshot to make a copy of. The cluster snapshot must exist and be in the *available* state. If you are copying the snapshot to another AWS Region, this identifier must be in the ARN format for the source AWS Region. This parameter is not case sensitive.
- **--target-db-cluster-snapshot-identifier** — Required. The identifier of the new cluster snapshot to create from the source cluster snapshot. This parameter is not case sensitive.

Target snapshot naming constraints:

- Cannot be the name of an existing snapshot.
- Length is [1—63] letters, numbers, or hyphens.
- First character must be a letter.
- Cannot end with a hyphen or contain two consecutive hyphens.
- Must be unique for all clusters across Amazon RDS, Neptune, and Amazon DocumentDB per AWS account, per Region.
- **--source-region** — If you are copying the snapshot to another AWS Region, specify the AWS Region that the encrypted cluster snapshot will be copied from.

If you're copying the snapshot to another AWS Region and you don't specify `--source-region`, you must specify the `pre-signed-url` option instead. The `pre-signed-url` value must be a URL that contains a Signature Version 4 signed request for the `CopyDBClusterSnapshot` action to be called in the source AWS Region where the cluster snapshot is copied from. To learn more about the `pre-signed-url`, see [CopyDBClusterSnapshot](#).

- **--kms-key-id** — The KMS key identifier for the key to use to encrypt the copy of the cluster snapshot.

If you are copying an encrypted cluster snapshot to another AWS Region, this parameter is required. You must specify a KMS key for the destination AWS Region.

If you are copying an encrypted cluster snapshot in the same AWS Region, the AWS KMS key parameter is optional. The copy of the cluster snapshot is encrypted with the same AWS KMS key as the source cluster snapshot. If you want to specify a new AWS KMS encryption key to use to encrypt the copy, you can do so using this parameter.

- **--copy-tags** — **Optional. The tags and values to be copied over.**

To cancel a copy operation once it's in progress, you can delete the target cluster snapshot identified by `--target-db-cluster-snapshot-identifier` or `TargetDBClusterSnapshotIdentifier` while that cluster snapshot is in **copying** status.

Example

Example 1: Copy an unencrypted snapshot to the same Region

The following AWS CLI example creates a copy of `sample-cluster-snapshot` named `sample-cluster-snapshot-copy` in the same AWS Region as the source snapshot. When the copy is made, all tags on the original snapshot are copied to the snapshot copy.

For Linux, macOS, or 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
```

For 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 from this operation looks something like the following.

```
{
  "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

Example 2: Copy an unencrypted snapshot across AWS Regions

The following AWS CLI example creates a copy of sample-cluster-snapshot, which has the ARN arn:aws:rds:us-east-1:123456789012:cluster-snapshot:sample-cluster-snapshot. This copy is named sample-cluster-snapshot-copy and is in the AWS Region in which the command is run.

For Linux, macOS, or 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
```

For 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 from this operation looks something like the following.

```
{
    "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-1:111122223333:cluster-
snapshot:sample-cluster-snapshot-copy",
        "SourceDBClusterSnapshotArn": "arn:aws:rds:us-east-1:111122223333:cluster-
snapshot:sample-cluster-snapshot",
    }
}
```

Example

Example 3: Copy an encrypted snapshot across AWS Regions

The following AWS CLI example creates a copy of sample-cluster-snapshot from the us-west-2 Region to the us-east-1 Region. This command is called in the us-east-1 Region.

For Linux, macOS, or 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
```

For 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 from this operation looks something like the following.

```
{  
    "DBClusterSnapshot": {  
        "AvailabilityZones": [],  
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot-copy",  
        "DBClusterIdentifier": "ayhu-xrsc-test-ap-southeast-1-small-cluster-kms",  
        "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-1:111122223333:key/sample-key-id",  
        "DBClusterSnapshotArn": "arn:aws:rds:us-east-1:111122223333:cluster-  
snapshot:sample-cluster-snapshot-copy",  
        "SourceDBClusterSnapshotArn": "arn:aws:rds:us-west-2:111122223333:cluster-  
snapshot:sample-cluster-snapshot",  
    }  
}
```

Sharing Amazon DocumentDB Cluster Snapshots

In Amazon DocumentDB, you can share manual cluster snapshots, which can be copied by authorized AWS accounts. You can share encrypted or unencrypted manual snapshots. When sharing an unencrypted snapshot, authorized AWS accounts can restore the cluster directly from the snapshot instead of making a copy of it and restoring from that. However, you can't restore a cluster from a snapshot that is both shared and encrypted. Instead, you can make a copy of the cluster and restore the cluster from that copy. For more information about copying a snapshot, see [Copying Amazon DocumentDB Cluster Snapshots \(p. 226\)](#).

Note

You can't share an Amazon DocumentDB automated cluster snapshot. As a workaround, you can create a manual snapshot by copying the automated snapshot, and then share that copy. For more information about copying a snapshot, see [Copying Amazon DocumentDB Cluster Snapshots \(p. 226\)](#). For more information about restoring a cluster from a snapshot, see [Restoring from a Cluster Snapshot \(p. 236\)](#).

You can share a manual snapshot with up to 20 other AWS accounts. You can also share an unencrypted manual snapshot as public, which makes the snapshot available to all accounts. When sharing a snapshot as public, ensure that none of your private information is included in any of your public snapshots.

When sharing manual snapshots with other AWS accounts, and you restore a cluster from a shared snapshot using the AWS CLI or the Amazon DocumentDB API, you must specify the Amazon Resource Name (ARN) of the shared snapshot as the snapshot identifier.

Sharing an Encrypted Snapshot

The following restrictions apply to sharing encrypted snapshots:

- You can't share encrypted snapshots as public.
- You can't share a snapshot that has been encrypted using the default AWS KMS encryption key of the account that shared the snapshot.

Follow these steps to share encrypted snapshots.

1. Share the AWS Key Management Service (AWS KMS) encryption key that was used to encrypt the snapshot with any accounts that you want to be able to access the snapshot.

You can share AWS KMS encryption keys with another AWS accounts by adding the other accounts to the AWS KMS key policy. For details on updating a key policy, see [Using Key Policies in AWS KMS](#) in the *AWS Key Management Service Developer Guide*. For an example of creating a key policy, see [Creating an IAM Policy to Enable Copying of the Encrypted Snapshot \(p. 234\)](#) later in this topic.

2. Use the AWS CLI, as shown below (p. 235), to share the encrypted snapshot with the other accounts.

Allowing Access to an AWS KMS Encryption Key

For another AWS account to copy an encrypted snapshot shared from your account, the account that you share your snapshot with must have access to the AWS KMS key that encrypted the snapshot. To allow another account access to an AWS KMS key, update the key policy for the AWS KMS key with the ARN of the account that you are sharing to as a principal in the AWS KMS key policy. Then allow the kms:CreateGrant action.

After you give an account access to your AWS KMS encryption key, to copy your encrypted snapshot, that account must create an AWS Identity and Access Management (IAM) user if it doesn't already have one. In addition, that account must also attach an IAM policy to that IAM user that allows the user to copy an encrypted snapshot using your AWS KMS key. The account must be an IAM user and cannot be a root AWS account identity due to AWS KMS security restrictions.

In the following key policy example, user 123451234512 is the owner of the AWS KMS encryption key. User 123456789012 is the account that the key is being shared with. This updated key policy gives the account access to the AWS KMS key. It does this by including the ARN for the root AWS account identity for user 123456789012 as a principal for the policy, and by allowing the kms:CreateGrant action.

```
{  
    "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",  
            "Resource": "arn:aws:kms:us-east-1:123451234512:key/12345678-1234-1234-1234-123456789012"  
        }  
    ]  
}
```

```

        "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}}
    }
]
}

```

Creating an IAM Policy to Enable Copying of the Encrypted Snapshot

When the external AWS account has access to your AWS KMS key, the owner of that account can create a policy to allow an IAM user that is created for the account to copy an encrypted snapshot that is encrypted with that AWS KMS key.

The following example shows a policy that can be attached to an IAM user for AWS account 123456789012. The policy enables the IAM user to copy a shared snapshot from account 123451234512 that has been encrypted with the AWS KMS key c989c1dd-a3f2-4a5d-8d96-e793d082ab26 in the us-west-2 Region.

```
{
    "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
        }
    }
]
```

For details on updating a key policy, see [Using Key Policies in AWS KMS](#) in the *AWS Key Management Service Developer Guide*.

Sharing a Snapshot

To share a snapshot, use the Amazon DocumentDB `modify-db-snapshot-attribute` operation. Use the `--values-to-add` parameter to add a list of the IDs for the AWS accounts that are authorized to restore the manual snapshot.

The following example permits two AWS account identifiers, 123451234512 and 123456789012, to restore the snapshot named `manual-snapshot1`. It also removes the `all` attribute value to mark the snapshot as private.

For Linux, macOS, or Unix:

```
aws docdb modify-db-cluster-snapshot-attribute \
--db-cluster-snapshot-identifier sample-cluster-snapshot \
--attribute-name restore \
--values-to-add '[{"123451234512","123456789012"}'
```

For Windows:

```
aws docdb modify-db-cluster-snapshot-attribute ^
--db-cluster-snapshot-identifier sample-cluster-snapshot ^
--attribute-name restore ^
--values-to-add '[{"123451234512","123456789012"}']
```

Output from this operation looks something like the following.

```
{
    "DBClusterSnapshotAttributesResult": {
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot",
        "DBClusterSnapshotAttributes": [
            {
                "AttributeName": "restore",
                "AttributeValues": [
                    "123451234512",
                    "123456789012"
                ]
            }
        ]
    }
}
```

To remove an AWS account identifier from the list, use the `--values-to-remove` parameter. The following example prevents AWS account ID 123456789012 from restoring the snapshot.

For Linux, macOS, or Unix:

```
aws docdb modify-db-cluster-snapshot-attribute \
--db-cluster-snapshot-identifier sample-cluster-snapshot \
--attribute-name restore \
--values-to-remove '[\"123456789012\"]'
```

For Windows:

```
aws docdb modify-db-cluster-snapshot-attribute ^
--db-cluster-snapshot-identifier sample-cluster-snapshot ^
--attribute-name restore ^
--values-to-remove '[\"123456789012\"]'
```

Output from this operation looks something like the following.

```
{
    "DBClusterSnapshotAttributesResult": {
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot",
        "DBClusterSnapshotAttributes": [
            {
                "AttributeName": "restore",
                "AttributeValues": [
                    "123451234512"
                ]
            }
        ]
    }
}
```

Restoring from a Cluster Snapshot

Amazon DocumentDB (with MongoDB compatibility) creates a cluster snapshot of your storage volume. You can create a new cluster by restoring from a cluster snapshot. When you restore the cluster, you provide the name of the cluster snapshot to restore from and a name for the new cluster that is created by the restore. You can't restore from a snapshot to an existing cluster because a new cluster is created when you restore.

When you are restoring a cluster from a cluster snapshot:

- This action restores only the cluster, and not the instances for that cluster. You must invoke the `create-db-instance` action to create instances for the restored cluster, specifying the identifier of the restored cluster in `--db-cluster-identifier`. You can create instances only after the cluster is *available*.
- You cannot restore an encrypted snapshot to an unencrypted cluster. However, you can restore an unencrypted snapshot to an encrypted cluster by specifying the AWS KMS key.
- To restore a cluster from an encrypted snapshot, you must have access to the AWS KMS key.

Note

You cannot restore a 3.6 cluster to a 4.0 cluster but you can migrate from one cluster version to another. For more information, go to [Migrating to Amazon DocumentDB \(p. 121\)](#).

Restore from a Cluster Snapshot Using the AWS Management Console

The following procedure shows how to restore an Amazon DocumentDB cluster from a cluster snapshot using the Amazon DocumentDB Management Console.

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Snapshots**, and then choose the button to the left of the snapshot that you want to use to restore a cluster.

Tip

If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.

3. On the **Actions** menu, choose **Restore**.
4. On the **Restore snapshot** page, complete the **Configuration** section.
 - a. **Cluster identifier** — The name for the new cluster. You can accept the Amazon DocumentDB supplied name or type a name that you prefer. The Amazon DocumentDB supplied name is in the format of docdb- plus a UTC timestamp; for example, docdb-yyyy-mm-dd-hh-mm-ss.
 - b. **Instance class** — The instance class for the new cluster. You can accept the default instance class or choose an instance class from the drop-down list.
 - c. **Number of instances** — The number of instances you want created with this cluster. You can accept the default of 3 instances (1 primary read/write and 2 read-only replicas) or choose the number of instances from the drop-down list.
5. If you are satisfied with the cluster configuration, choose **Restore cluster** and wait while your cluster is restored.
6. If you prefer to change some configurations, such as specifying a non-default Amazon VPC or security group, choose **Show advanced settings** from the bottom left of the page, and then continue with the following steps.
 - a. Complete the **Network settings** section.
 - **Virtual Private Cloud (VPC)** — Accept the current VPC, or choose a VPC from the drop-down list.
 - **Subnet Group** — Accept the default subnet group, or choose one from the drop-down list.
 - **VPC Security Groups** — Accept the default (VPC) security group, or choose one from the list.
 - b. Complete the **Cluster options** section.
 - **Database port** — Accept the default port, 27017, or use the up or down arrow to set the port that you want to use for application connections.
 - c. Complete the **Encryption** section.
 - **Encryption at rest** — If your snapshot is encrypted, these options are not available to you. If it is not encrypted, you can choose one of the following:
 - To encrypt all your cluster's data, choose **Enable encryption-at-rest**. If you choose this option, you must designate a **Master key**.
 - To not encrypt your cluster's data, choose **Disable encryption-at-rest**. If you choose this option, you are finished with the encryption section.
 - **Master key** — Choose one of the following from the drop-down list:

- **(default) aws/rds** — The account number and AWS KMS key ID are listed following this option.
 - **Customer-managed key** — This option is available only if you created an IAM encryption key in the AWS Identity and Access Management (IAM) console. You can choose the key to encrypt your cluster.
 - **Enter a key ARN** — In the **ARN** box, enter the Amazon Resource Name (ARN) for your AWS KMS key. The format of the ARN is `arn:aws:kms:<region>:<accountID>:key/<key-id>`.
- d. Complete the **Log exports** section.
- **Select the log types to publish to CloudWatch** — Choose one of the following:
 - **Enabled** — Enables your cluster to export DDL logging to Amazon CloudWatch Logs.
 - **Disabled** — Prevents your cluster from exporting DDL logs to Amazon CloudWatch Logs. **Disabled** is the default.
 - **IAM role** — From the list, choose *RDS Service Linked Role*.
- e. Complete the **Tags** section.
- **Add Tag** — In the **Key** box, enter the name for the tag for your cluster. In the **Value** box, optionally enter the tag value. Tags are used with AWS Identity and Access Management (IAM) policies to manage access to Amazon DocumentDB resources and to control what actions can be applied to the resources.
- f. Complete the **Deletion protection** section.
- **Enable deletion protection** — Protects the cluster from being accidentally deleted. While this option is enabled, you can't delete the cluster.
7. Choose **Restore cluster**.

Restore from a Cluster Snapshot Using the AWS CLI

To restore a cluster from a snapshot using the AWS CLI, use the `restore-db-cluster-from-snapshot` operation with the following parameters. For more information, see [RestoreDBClusterFromSnapshot \(p. 690\)](#).

- **--db-cluster-identifier** — Required. The name of the cluster that is created by the operation. A cluster by this name cannot exist before this operation.

Cluster naming constraints:

 - Length is [1—63] letters, numbers, or hyphens.
 - First character must be a letter.
 - Cannot end with a hyphen or contain two consecutive hyphens.
 - Must be unique for all clusters across Amazon RDS, Neptune, and Amazon DocumentDB per AWS account, per Region.
- **--snapshot-identifier** — Required. The name of the snapshot used to restore from. A snapshot by this name must exist and be in the *available* state.
- **--engine** — Required. Must be docdb.
- **--kms-key-id** — Optional. The ARN of the AWS KMS key identifier to use when restoring an encrypted snapshot or encrypting a cluster when restoring from an unencrypted snapshot. Supplying the AWS KMS key ID results in the restored cluster being encrypted with the AWS KMS key, whether or not the snapshot was encrypted.

The format of the `--kms-key-id` is `arn:aws:kms:<region>:<accountID>:key/<key-id>`. If you do not specify a value for the `--kms-key-id` parameter, then the following occurs:

- If the snapshot in `--snapshot-identifier` is encrypted, then the restored cluster is encrypted using the same AWS KMS key that was used to encrypt the snapshot.
- If the snapshot in `--snapshot-identifier` is not encrypted, then the restored cluster is not encrypted.

For Linux, macOS, or 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
```

For 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 from this operation looks something like the following.

```
{
  "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-abcdefg",
        "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
  }
}
```

```
}
```

After the cluster status is *available*, create at least one instance for the cluster.

For Linux, macOS, or 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
```

For 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 from this operation looks something like the following.

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "sample-cluster-restore-instance",
        "DBInstanceClass": "db.r5.large",
        "Engine": "docdb",
        "DBInstanceState": "creating",
        "PreferredBackupWindow": "02:00-02:30",
        "BackupRetentionPeriod": 1,
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-abcdefg",
                "Status": "active"
            }
        ],
        "AvailabilityZone": "us-west-2b",
        "DBSubnetGroup": {
            "DBSubnetGroupName": "default",
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-6242c31a",
            "SubnetGroupStatus": "Complete",
            "Subnets": [
                {
                    "SubnetIdentifier": "subnet-abcdefg",
                    "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"  
}  
}
```

Restoring to a Point in Time

You can restore a cluster to any point in time that is within the cluster's backup retention period using the AWS Management Console or AWS Command Line Interface (AWS CLI).

Note

You cannot conduct a point-in-time restore of a 3.6 cluster to a 4.0 cluster but you can migrate from one cluster version to another. For more information, go to [Migrating to Amazon DocumentDB \(p. 121\)](#).

Keep the following in mind when restoring a cluster to a point in time.

- The new cluster is created with the same configuration as the source cluster, except that the new cluster is created with the default parameter group. To set the new cluster's parameter group to the source cluster's parameter group, modify the cluster after it is *available*. For more information on modifying a cluster, see [Modifying an Amazon DocumentDB Cluster \(p. 287\)](#).

Restore to a Point in Time Using the AWS Management Console

You can restore a cluster to a point-in-time within its backup retention period by completing the following using the AWS Management Console.

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Clusters**. In the list of clusters, choose the button to the left of the cluster that you want to restore.

Tip

If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.

3. On the **Actions** menu, choose **Restore to point in time**.
4. Complete the **Restore time** section, which specifies the date and time to restore to.
 - a. **Restore date**—Choose or enter a date that is between the **Earliest restore time** and the **Latest restore time**.
 - b. **Restore time**—Choose or enter the hour, minute, and seconds that are between the **Earliest restore time** and the **Latest restore time**.

5. Complete the **Configuration** section.
 - a. **Cluster identifier** — Accept the default identifier, or enter an identifier that you prefer.

Cluster naming constraints:

 - Length is [1—63] letters, numbers, or hyphens.
 - First character must be a letter.
 - Cannot end with a hyphen or contain two consecutive hyphens.
 - Must be unique for all clusters across Amazon RDS, Neptune and Amazon DocumentDB per AWS account, per Region.
 - b. **Instance class** — From the drop-down list, choose the instance class that you want for the cluster's instances.
 - c. **Number of instances** — From the drop-down list, choose the number of instances that you want created when the cluster is restored.
6. Optional. To configure the network settings, cluster options, and enable log exports, choose **Show advanced settings**, and then complete the following sections. Otherwise, continue with the next step.
 - **Network settings**
 1. **Virtual Private Cloud (VPC)** — From the drop-down list, choose the VPC that you want to use for this cluster.
 2. **Subnet group** — From the drop-down list, choose the subnet group for this cluster.
 3. **VPC security groups** — From the drop-down list, choose the VPC security groups for this cluster.
 - **Cluster options**
 1. **Port** — Accept the default port (27017), or use the up and down arrows to set the port for communicating with this cluster.
 - **Log exports**
 1. **Audit logs** — Select this option to enable exporting audit logs to Amazon CloudWatch Logs. If you select this option, you must enable `audit_logs` in the cluster's custom parameter group. For more information, see [Auditing Amazon DocumentDB Events \(p. 205\)](#).
 2. **Profiler logs** — Select this option to enable exporting operation profiler logs to Amazon CloudWatch Logs. If you select this option, you must also modify the following parameters in the cluster's custom parameter group:
 - `profiler` — Set to `enabled`.
 - `profiler_threshold_ms` — Set to a value [0-INT_MAX] to set the threshold for profiling operations.
 - `profiler_sampling_rate` — Set to a value [0.0-1.0] to set the percentage of slow operations to profile.For more information, see [Profiling Amazon DocumentDB Operations \(p. 427\)](#).
 3. **Profiler logs** — Export profiler logs to Amazon CloudWatch
 4. **IAM role** — From the drop-down list, choose *RDS Service Linked Role*.
 - **Tags**
 1. **Add Tag** — In the *Key* box, enter the name for the tag for your cluster. In the *Value* box, optionally enter the tag value. Tags are used with AWS Identity and Access Management (IAM) policies to manage access to Amazon DocumentDB resources and to control what actions can be applied to the resources.

- **Deletion protection**

1. **Enable deletion protection** — Protects the cluster from being accidentally deleted. While this option is enabled, you can't delete the cluster.
7. To restore the cluster, choose **Create cluster**. Alternatively, you can choose **Cancel** to cancel the operation.

Restore To a Point in Time Using the AWS CLI

To restore a cluster to a point in time using the snapshot's backup retention period, use the `restore-db-cluster-to-point-in-time` operation with the following parameters.

- **--db-cluster-identifier**— Required. The name of the new cluster to be created. This cluster cannot exist before the operation. The parameter value must meet the following constraints.

Cluster naming constraints:

- Length is [1–63] letters, numbers, or hyphens.
- First character must be a letter.
- Cannot end with a hyphen or contain two consecutive hyphens.
- Must be unique for all clusters across Amazon RDS, Neptune and Amazon DocumentDB per AWS account, per Region.
- **--restore-to-time** — The UTC date and time to restore the cluster to. For example, `2018-06-07T23:45:00Z`.

Time Constraints:

- Must be before the latest restorable time for the cluster.
- Must be specified if the `--use-latest-restorable-time` parameter is not provided.
- Cannot be specified if the `--use-latest-restorable-time` parameter is true.
- Cannot be specified if the `--restore-type` parameter value is `copy-on-write`.
- **--source-db-cluster-identifier** — The name of the source cluster from which to restore. This cluster must exist and be available.
- **--use-latest-restorable-time** or **--no-use-latest-restorable-time** — Whether to restore to the latest restorable backup time. Cannot be specified if the `--restore-to-time` parameter is provided.

The AWS CLI operation `restore-db-cluster-to-point-in-time` only restores the cluster, not the instances for that cluster. You must invoke the `create-db-instance` operation to create instances for the restored cluster, specifying the identifier of the restored cluster in `--db-cluster-identifier`. You can create instances only after the `restore-db-cluster-to-point-in-time` operation has completed and the restored cluster is *available*.

Example

The following example creates `sample-cluster-restored` from the snapshot `sample-cluster-snapshot` to the latest restorable time.

For Linux, macOS, or 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
```

For 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

The following example creates `sample-cluster-restored` from the snapshot `sample-cluster-snapshot` to 03:15 on December 11, 2018 (UTC), which is within the backup retention period of `sample-cluster`.

For Linux, macOS, or 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
```

For 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 from this operation looks something like the following.

```
{
    "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",
        "ProcessorFeatureEnabled": [
            "ProcessorFeatureEnabled"
        ]
    }
}
```

```
        "DbClusterResourceId": "cluster-ABCDEFGHIJKLMNPQRSTUVWXYZ",
        "DBClusterArn": "arn:aws:rds:us-east-1:<accountID>:cluster:sample-cluster-
restored",
        "AssociatedRoles": [],
        "ClusterCreateTime": "2020-04-24T20:14:36.713Z",
        "DeletionProtection": false
    }
}
```

Deleting a Cluster Snapshot

A manual snapshot is a full backup that is deleted only when you manually delete it using the AWS Management Console or AWS CLI. You cannot manually delete an automatic snapshot because automatic snapshots are deleted only when the snapshot's retention period expires or you delete the snapshot's cluster.

Delete a Cluster Snapshot Using the AWS Management Console

To delete a manual cluster snapshot using the AWS Management Console, complete the following steps.

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Snapshots**.

Tip
If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.
3. In the list of snapshots, choose the button to the left of the snapshot that you want to delete. The snapshot's type must be **manual**.
 1. You can verify that that the snapshot's type is **manual** by checking if it is listed as manual or automatic under the **Type** column.
 4. From the **Actions** menu, choose **Delete**. If the **Delete** option is unavailable, you probably chose an automatic snapshot.
 5. On the delete confirmation screen, to delete the snapshot, choose **Delete**. To keep the snapshot, choose **Cancel**.

Delete a Cluster Snapshot Using the AWS CLI

An Amazon DocumentDB manual cluster snapshot is a full backup that you can manually delete using the AWS CLI. You cannot manually delete an automatic snapshot.

To delete a manual cluster snapshot using the AWS CLI, use the `delete-db-cluster-snapshot` operation with the following parameters.

Parameters

- **--db-cluster-snapshot-identifier** — Required. The name of the manual snapshot to delete.

The following example deletes the cluster snapshot `sample-cluster-snapshot`.

For Linux, macOS, or Unix:

```
aws docdb delete-db-cluster-snapshot \
--db-cluster-snapshot-identifier sample-cluster-snapshot
```

For Windows:

```
aws docdb delete-db-cluster-snapshot ^
--db-cluster-snapshot-identifier sample-cluster-snapshot
```

Output from this operation lists the details of the cluster snapshot you deleted.

Managing Amazon DocumentDB Resources

These sections cover the various components and their related tasks for managing your Amazon DocumentDB (with MongoDB compatibility) implementation.

Topics

- [Amazon DocumentDB Operational Tasks Overview \(p. 247\)](#)
- [Overview of Amazon DocumentDB Global Clusters \(p. 252\)](#)
- [Managing Amazon DocumentDB Clusters \(p. 269\)](#)
- [Managing Amazon DocumentDB Instances \(p. 313\)](#)
- [Managing Amazon DocumentDB Subnet Groups \(p. 334\)](#)
- [Amazon DocumentDB High Availability and Replication \(p. 343\)](#)
- [Managing Amazon DocumentDB Events \(p. 349\)](#)
- [Choosing Regions and Availability Zones \(p. 353\)](#)
- [Managing Amazon DocumentDB Cluster Parameter Groups \(p. 355\)](#)
- [Understanding Amazon DocumentDB Endpoints \(p. 378\)](#)
- [Understanding Amazon DocumentDB Amazon Resource Names \(ARNs\) \(p. 384\)](#)
- [Tagging Amazon DocumentDB Resources \(p. 387\)](#)
- [Maintaining Amazon DocumentDB \(p. 392\)](#)
- [Understanding Service-Linked Roles \(p. 398\)](#)

Amazon DocumentDB Operational Tasks Overview

This section covers operational tasks for your Amazon DocumentDB (with MongoDB compatibility) cluster, and how to accomplish these tasks using the AWS CLI.

Topics

- [Adding a Replica to an Amazon DocumentDB Cluster \(p. 247\)](#)
- [Describing Clusters and Instances \(p. 248\)](#)
- [Creating a Cluster Snapshot \(p. 249\)](#)
- [Restoring from a Snapshot \(p. 250\)](#)
- [Removing an Instance from a Cluster \(p. 251\)](#)
- [Deleting a Cluster \(p. 251\)](#)

Adding a Replica to an Amazon DocumentDB Cluster

After you create the primary instance for your Amazon DocumentDB cluster, you can add one or more *replicas*. A replica is a read-only instance that serves two purposes:

- **Scalability** — If you have a large number of clients that require concurrent access, you can add more replicas for read-scaling.
- **High availability** — If the primary instance fails, Amazon DocumentDB automatically fails over to a replica instance and designates it as the new primary. If a replica fails, other instances in the cluster can still serve requests until the failed node can be recovered.

Each Amazon DocumentDB cluster can support up to 15 replicas.

Note

For maximum fault tolerance, you should deploy replicas in separate Availability Zones. This helps ensure that your Amazon DocumentDB cluster can continue to function, even if an entire Availability Zone becomes unavailable.

The following AWS CLI example shows how to add a new replica. The `--availability-zone` parameter places the replica in the specified Availability Zone.

```
aws docdb create-db-instance \
    --db-instance-identifier sample-instance \
    --db-cluster-identifier sample-cluster \
    --engine docdb \
    --db-instance-class db.r5.large \
    --availability-zone us-east-1a
```

Describing Clusters and Instances

The following AWS CLI example lists all Amazon DocumentDB clusters in a Region. For certain management features such as cluster and instance lifecycle management, Amazon DocumentDB leverages operational technology that is shared with Amazon RDS. The `filterName=engine,Values=docdb` filter parameter returns only Amazon DocumentDB clusters.

For more information on describing and modifying clusters, see the [Amazon DocumentDB Cluster Lifecycle \(p. 273\)](#).

```
aws docdb describe-db-clusters --filter Name=engine,Values=docdb
```

Output from this operation looks something like the following.

```
{
  "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",
      ...
    }
  ]
}
```

```
        "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",
    ...
}
]
```

The following AWS CLI example lists the instances in an Amazon DocumentDB cluster. For more information on describing and modifying clusters, see the [Amazon DocumentDB Instance Lifecycle \(p. 319\)](#).

```
aws docdb describe-db-clusters \
--db-cluster-identifier sample-cluster \
--query 'DBClusters[*].[DBClusterMembers]'
```

The output looks like something like below. In this output, there are two instances. The primary instance is sample-instance-1 ("IsClusterWriter": true). There is also a replica instance, 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
        }
      ]
    ]
]
```

Creating a Cluster Snapshot

A *cluster snapshot* is a complete backup of the data in your Amazon DocumentDB cluster. When the snapshot is being created, Amazon DocumentDB reads your data directly from the cluster volume.

Because of this, you can create a snapshot even if your cluster doesn't have any instances running at the time. The amount of time it takes to create a snapshot depends on the size of your cluster volume.

Amazon DocumentDB supports automatic backups, which occur daily during the preferred backup window — a 30-minute period of time during the day. The following AWS CLI example shows how to view the backup window for your cluster:

```
aws docdb describe-db-clusters \
--db-cluster-identifier sample-cluster \
--query 'DBClusters[*].PreferredBackupWindow'
```

The output shows the backup window (in UTC):

```
[  
    "00:18-00:48"  
]
```

You can define the backup window when you create your Amazon DocumentDB cluster. You can also change the backup window, as shown in the following example. If you don't define a backup window, Amazon DocumentDB automatically assigns one to your cluster.

```
aws docdb modify-db-cluster \
--db-cluster-identifier sample-cluster \
--preferred-backup-window "02:00-02:30"
```

In addition to automatic backups, you can manually create a cluster snapshot at any time. When you do this, you specify which cluster you want to back up, and a unique name for your snapshot so that you can restore from it later.

The following AWS CLI example shows how to create a snapshot of your data.

```
aws docdb create-db-cluster-snapshot \
--db-cluster-identifier sample-cluster \
--db-cluster-snapshot-identifier sample-cluster-snapshot
```

Restoring from a Snapshot

You can restore a cluster snapshot to a new Amazon DocumentDB cluster. To do this, you provide the name of the snapshot and the name of a new cluster. You can't restore from a snapshot to an existing cluster; instead, Amazon DocumentDB creates a new cluster when you restore and then populates it with your snapshot data.

The following example shows all the snapshots for the cluster sample-cluster.

```
aws docdb describe-db-cluster-snapshots \
--db-cluster-identifier sample-cluster \
--query 'DBClusterSnapshots[*].[DBClusterSnapshotIdentifier,SnapshotType,Status]'
```

The output looks something like the following. A manual snapshot is one that you created manually, whereas an automated snapshot is created by Amazon DocumentDB within the cluster backup window.

```
[  
    "sample-cluster-snapshot",  
    "manual",  
]
```

```
        "available"
    ],
[
    "rds:sample-cluster",
    "automated",
    "available"
]
]
```

The following example shows how to restore an Amazon DocumentDB cluster from a snapshot.

```
aws docdb restore-db-cluster-from-snapshot \
--engine docdb \
--db-cluster-identifier new-sample-cluster \
--snapshot-identifier sample-cluster-snapshot
```

The new cluster does not have any instances associated with it; so if you want to interact with the cluster, you must add an instance to it.

```
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
```

You can use the following AWS CLI operations to monitor the progress of cluster and instance creation. When the cluster and instance statuses are available, you can connect to the new cluster's endpoint and access your data.

```
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[*].[DBInstanceState]'
```

Removing an Instance from a Cluster

Amazon DocumentDB stores all of your data in the cluster volume. The data persists in that cluster volume, even if you remove all the instances from your cluster. If you need to access the data again, you can add an instance to the cluster at any time, and pick up where you left off.

The following example shows how to remove an instance from your Amazon DocumentDB cluster.

```
aws docdb delete-db-instance \
--db-instance-identifier sample-instance
```

Deleting a Cluster

Before you can delete an Amazon DocumentDB cluster, you must first remove all of its instances. The following AWS CLI example returns information about the instances in a cluster. If this operation returns any instance identifiers, you have to delete each of the instances. For more information, see [Removing an Instance from a Cluster \(p. 251\)](#).

```
aws docdb describe-db-clusters \
--db-cluster-identifier sample-cluster \
--query 'DBClusters[*].DBClusterMembers[*].DBInstanceIdentifier'
```

When there are no more instances remaining, you can delete the cluster. At that time, you must choose one of the following options:

- **Create a final snapshot** — Capture all the cluster data in a snapshot so that you can re-create a new instance with that data later. The following example shows how to do this:

```
aws docdb delete-db-cluster \
--db-cluster-identifier sample-cluster \
--final-db-snapshot-identifier sample-cluster-snapshot
```

- **Skip the final snapshot** — Permanently discard all the cluster data. This cannot be reversed. The following example shows how to do this:

```
aws docdb delete-db-cluster \
--db-cluster-identifier sample-cluster \
--skip-final-snapshot
```

Overview of Amazon DocumentDB Global Clusters

What is a global cluster?

A global cluster consists of one primary region and up to five read-only secondary regions. You issue write operations directly to the primary cluster in the primary region and Amazon DocumentDB automatically replicates the data to the secondary regions using dedicated infrastructure. Latency is typically under a second.

How are global clusters useful?

- **Recovery from region-wide outages** — In the event of a region-wide outage, you can promote one of the secondary clusters to a primary cluster within minutes, with a typical Recovery Time Objective (RTO) of under a minute. The Recovery Point Objective (RPO) is typically measured in seconds, but this depends on the lag across the network at the time of the failure.
- **Global reads with local latency** — If you have offices around the world, you can use a global cluster to keep your main sources of information updated in the primary region. Offices in your other regions can access the information in their own region, with local latency.
- **Scalable secondary clusters** — You can scale your secondary clusters by adding more read-only instances to a secondary region. The secondary cluster is read-only, so it can support up to 16 read-only replica instances rather than the usual limit of 15 for a single cluster.
- **Fast replication from primary to secondary clusters** — The replication performed by a global cluster has little performance impact on the primary database cluster. The resources of the DB instances are fully devoted to serve application read and write workloads.

What are the current limitations of global clusters?

- Global clusters are not supported on Amazon DocumentDB v3.6.
- Global clusters are not supported on t3, t4g, and r4 instance types.

- Global clusters are not available in the following regions: AWS GovCloud (US-West), South America (São Paulo), Europe (Milan), China (Beijing), and China (Ningxia).
- In the event of a regional failover, you must manually promote a secondary cluster to become the primary cluster, and modify your application to point to the new primary cluster.
- Only the primary cluster performs write operations. Clients that perform write operations connect to the cluster endpoint of the primary cluster.
- You can have a maximum of five secondary regions and one primary region for your cluster.
- A secondary cluster cannot be stopped. A primary cluster cannot be stopped if it has secondary clusters associated with it. Only a regional cluster that has no secondary clusters can be stopped.
- Replicas attached to the secondary cluster can restart under certain circumstances. If the primary region's instance restarts or fails over, replicas in the secondary region also restart. The cluster is then unavailable until all replicas are back in sync with the primary database cluster's writer instance. This behavior is expected. Be sure that you understand the impact to your global cluster before making changes to your primary cluster.
- You cannot use AWS CloudFormation to manage global clusters.
- You cannot use change streams on secondary clusters.

Topics

- [Quick Start Guide: Global Clusters \(p. 253\)](#)
- [Managing an Amazon DocumentDB global cluster \(p. 262\)](#)
- [Connect to an Amazon DocumentDB Global Clusters \(p. 267\)](#)
- [Monitoring Amazon DocumentDB Global Clusters \(p. 267\)](#)
- [Disaster Recovery and Amazon DocumentDB Global Clusters \(p. 268\)](#)

Quick Start Guide: Global Clusters

Topics

- [Configuration \(p. 253\)](#)
- [Creating an Amazon DocumentDB global cluster \(p. 254\)](#)
- [Adding an AWS Region to an Amazon DocumentDB global cluster \(p. 259\)](#)
- [Using a snapshot for your Amazon DocumentDB global cluster \(p. 262\)](#)

Configuration

Amazon DocumentDB global cluster spans at least two AWS Regions. The primary region supports a cluster that has one primary (writer) instance and up to fifteen replica instances, while a secondary region runs a read-only cluster made up entirely of up to sixteen replica instances. A global cluster can have up to five secondary regions. The table lists the maximum clusters, instances, and replicas allowed in a global cluster.

Description	Primary AWS Region	Secondary AWS Region
Clusters	1	5 (maximum)
Writer instances	1	0
Read-only instances (Amazon DocumentDB replicas), per cluster	15 (max)	16 (total)

Description	Primary AWS Region	Secondary AWS Region
Read-only instances (max allowed, given actual number of secondary regions)	15 - s	s = total number of secondary AWS Regions

The clusters have the following specific requirements:

- **Database instance class requirements** — You can only use the db.r5 and db.r6 instance classes.
- **AWS Region requirements** — The primary cluster must be in one region, and at least one secondary cluster must be in a different region of the same account. You can create up to five secondary (read-only) clusters, and each must be in a different region. In other words, no two clusters can be in the same region.
- **Naming requirements** — The names you choose for each of your clusters must be unique, across all regions. You can't use the same name for different clusters even though they're in different regions.

Creating an Amazon DocumentDB global cluster

Are you ready to build your first global cluster? In this section we will explain how to create a brand new global cluster with new database clusters and instances, using either the AWS Management Console or AWS CLI with the following instructions.

Using the AWS Management Console

1. In the AWS Management Console, navigate to **Amazon DocumentDB**.
2. When you get to the Amazon DocumentDB console, choose **Clusters**.

Cluster identifier	global-add-region-test	docdb-2us-east-2	global-bom-fra
global-add-region-test			
docdb-2us-east-2			
global-bom-fra			

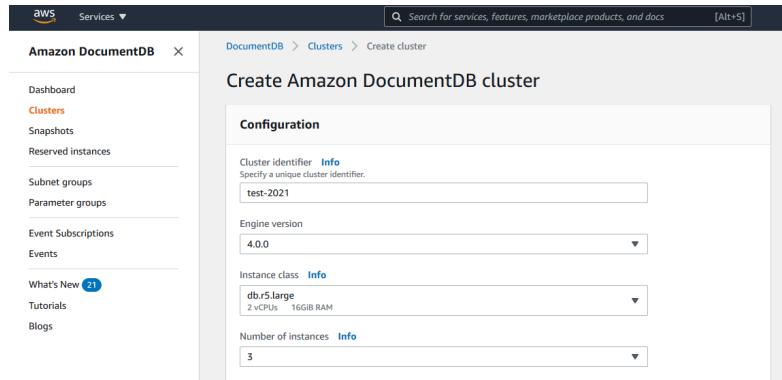
3. Choose **Create**.

Role	Engine version	Region & AZ	Status	Size	Maintenance
Global cluster	4.0.0	3 regions	available	3 clusters	-
Regional cluster	4.0.0	us-east-2	available	1 Instance	None
Global cluster	4.0.0	3 regions	available	3 clusters	-

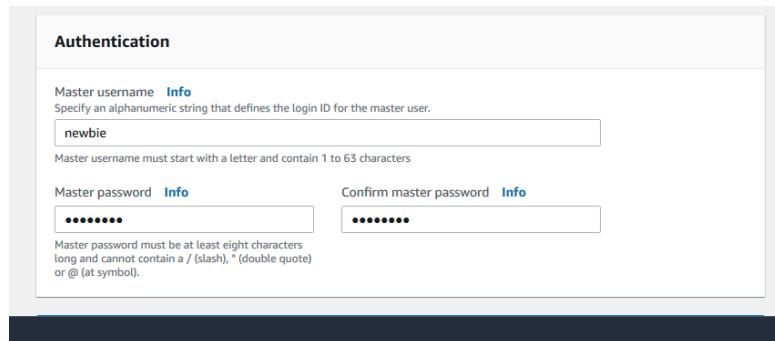
4. Fill out the **Configuration** section of the **Create Amazon DocumentDB Cluster** form accordingly:

- **Cluster identifier:** You can either enter a unique identifier for this instance or allow Amazon DocumentDB to provide the instance identifier based on the cluster identifier.
- **Engine version:** Choose **4.0.0**
- **Instance class:** Choose **db.r5.large**

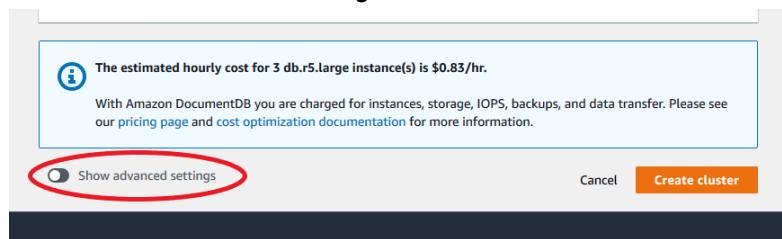
- Number of instances: Choose **3**.



5. In the **Authentication** section, fill in a master username and master password.

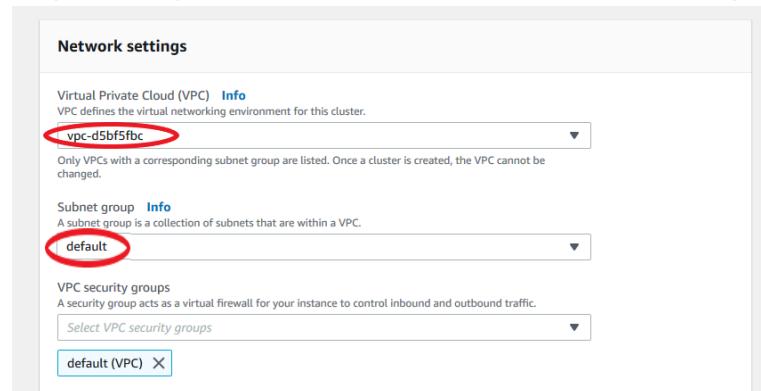


6. Choose **Show Advanced Settings**.

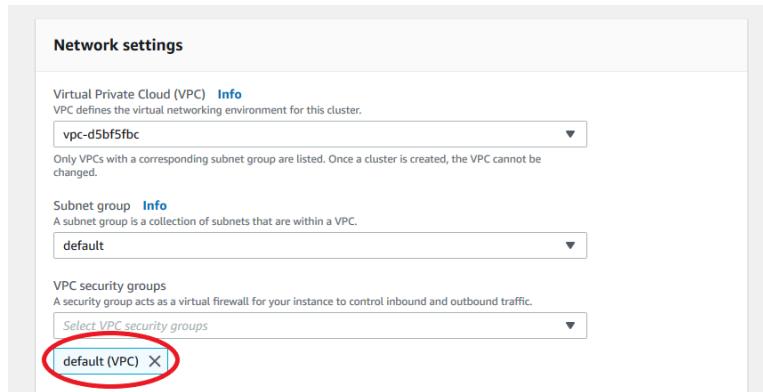


7. In the **Network Settings** section:

- Keep default options for **Virtual Private Cloud** and **Subnet Group**.



- For **VPC Security Groups**, **default VPC** should already be added.



Network settings

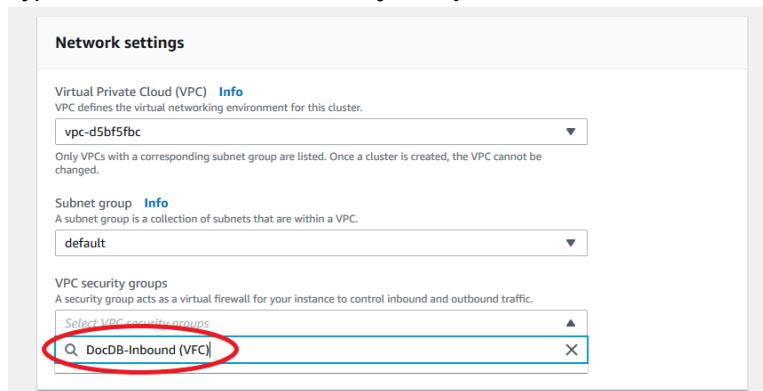
Virtual Private Cloud (VPC) [Info](#)
VPC defines the virtual networking environment for this cluster.

VPC security groups
A security group acts as a virtual firewall for your instance to control inbound and outbound traffic.

Select VPC security groups

default (VPC) X

- Type DocDB into the **VPC Security Groups** field and select **DocDB-Inbound (VPC)**.



Network settings

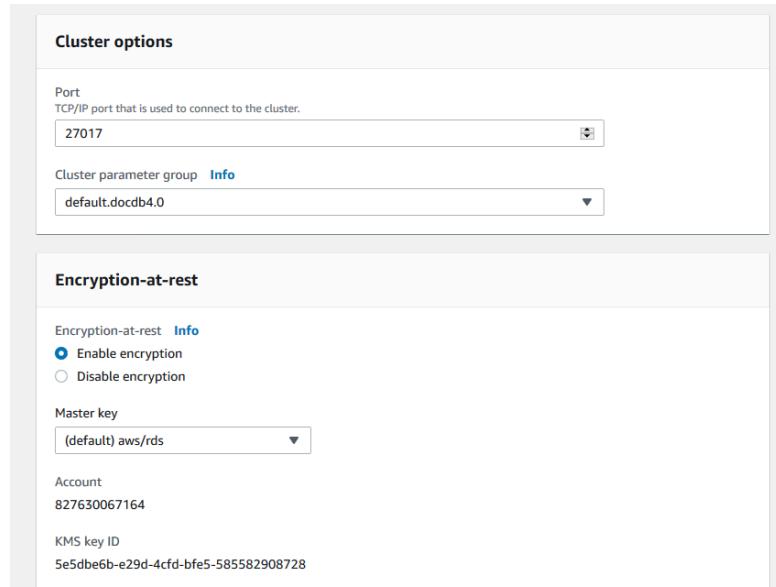
Virtual Private Cloud (VPC) [Info](#)
VPC defines the virtual networking environment for this cluster.

VPC security groups
A security group acts as a virtual firewall for your instance to control inbound and outbound traffic.

Select VPC security groups

Q DocDB-Inbound (VFC) X

8. For Cluster Options and Encryption-at-rest, leave at default selections.



Cluster options

Port
TCP/IP port that is used to connect to the cluster.
27017

Cluster parameter group [Info](#)
default.docdb4.0

Encryption-at-rest

Encryption-at-rest [Info](#)
 Enable encryption
 Disable encryption

Master key
(default) aws/rds

Account
827630067164

KMS key ID
5e5dbe6b-e29d-4cfb-bfe5-585582908728

9. For Backup and Log Exports, leave at default selections.

Backup

Backup retention period [Info](#)
A period between 1 and 35 days in which you can perform a point-in-time restore and for which automated backups are retained.

Backup window
The daily time range (in UTC) during which automated backups are created.

Start time	Duration
00 : 00 UTC	0.5 hours

Log exports

Select the log types to publish to Amazon CloudWatch Logs

- Audit logs
- Profiler logs

IAM role
The following service-linked role is used for publishing logs to CloudWatch Logs.

i To enable auditing, ensure that both exporting auditing logs to Amazon CloudWatch is enabled and the Cluster Parameter "Auditing" is enabled.
[Learn more](#)

10. For Maintenance, Tags and Deletion Protection, leave default selections.

Maintenance

Maintenance window [Info](#)
The period in which pending modifications or patches are applied to Instances in the cluster.

- Select window
- No preference

Tags

No tags

Deletion protection

Enable deletion protection
Protects the cluster from being accidentally deleted. While this option is enabled, you can't delete the cluster.

11. Now click the button that says Create.

The estimated hourly cost for 3 db.r5.large instance(s) is \$0.83/hr.

With Amazon DocumentDB you are charged for instances, storage, IOPS, backups, and data transfer. Please see our [pricing page](#) and [cost optimization documentation](#) for more information.

Show advanced settings

Using the AWS CLI

To create an Amazon DocumentDB regional cluster, call the `create-db-cluster` AWS CLI. The following AWS CLI command creates an Amazon DocumentDB cluster named `global-cluster-id`. For more information on deletion protection, see [Deleting an Amazon DocumentDB Cluster \(p. 296\)](#).

Also, `--engine-version` is an optional parameter that defaults to the latest major engine version. The current major engine version is `4.0.0`. When new major engine versions are released, the default engine version for `--engine-version` will be updated to reflect the lastest major engine version. As a result, for production workloads, and especially those that are dependent on scripting, automation, or AWS CloudFormation templates, we recommend that you explicitly specify the `--engine-version` to the intended major version.

If a `db-subnet-group-name` or `vpc-security-group-id` is not specified, Amazon DocumentDB will use the default subnet group and Amazon VPC security group for the given region.

In the following example, replace each *user input placeholder* with your own information.

For Linux, macOS, or 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
```

For 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 from this operation looks something like the following (JSON format).

```
{
    "DBCluster": {
        "StorageEncrypted": false,
        "DBClusterMembers": [],
        "Engine": "docdb",
        "DeletionProtection": "enabled",
        "ClusterCreateTime": "2018-11-26T17:15:19.885Z",
        "DBSubnetGroup": "default",
        "EngineVersion": "4.0.0",
        "MasterUsername": "masteruser",
        "BackupRetentionPeriod": 1,
        "DBClusterArn": "arn:aws:rds:us-east-1:123456789012:cluster:cluster-id",
        "DBClusterIdentifier": "cluster-id",
        "MultiAZ": false,
        "DBClusterParameterGroup": "default.docdb4.0",
        "PreferredBackupWindow": "09:12-09:42",
        "DbClusterResourceId": "cluster-KQSGI4MHU4NTDDRVNLNTU7XVAY",
        "PreferredMaintenanceWindow": "tue:04:17-tue:04:47",
        "Port": 27017,
        "Status": "creating",
        "ReaderEndpoint": "cluster-id.cluster-ro-sfcrlcjcoroz.us-east-1.docdb.amazonaws.com",
        "AssociatedRoles": [],
        "HostedZoneId": "ZNKXTT8WH85VW",
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-77186e0d",
                "Status": "active"
            }
        ],
        "AvailabilityZones": [
            "us-east-1a"
        ]
    }
}
```

```
        "us-east-1c",
        "us-east-1e"
    ],
    "Endpoint": "cluster-id.cluster-sfcrlcjcoroz.us-east-1.docdb.amazonaws.com"
}
```

It takes several minutes to create the cluster. You can use the AWS Management Console or AWS CLI to monitor the status of your cluster. For more information, see [Monitoring an Amazon DocumentDB Cluster's Status \(p. 402\)](#).

Important

When you use the AWS CLI to create an Amazon DocumentDB regional cluster, no instances are created. Consequently, you must explicitly create a primary instance and any replica instances that you need. You can use either the console or AWS CLI to create the instances. For more information, see [Adding an Amazon DocumentDB Instance to a Cluster \(p. 319\)](#) and [CreateDBCluster \(p. 584\)](#) in the Amazon DocumentDB API Reference.

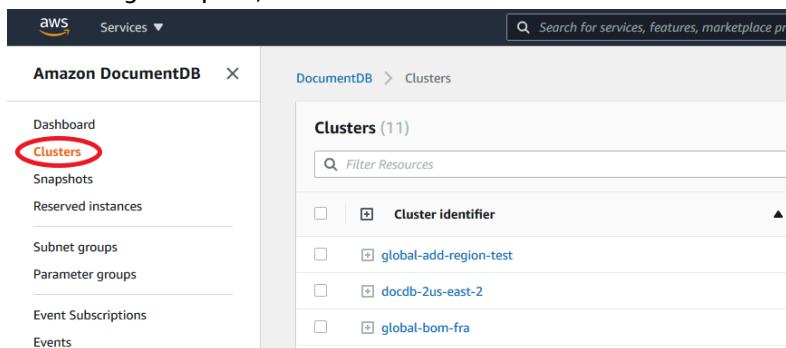
Once your regional cluster is available, you can add a secondary cluster in another region with the following instructions: [Adding an AWS Region to an Amazon DocumentDB global cluster \(p. 259\)](#). When you add a region, your regional cluster becomes your primary cluster, and you have a new secondary cluster in the region you chose.

Adding an AWS Region to an Amazon DocumentDB global cluster

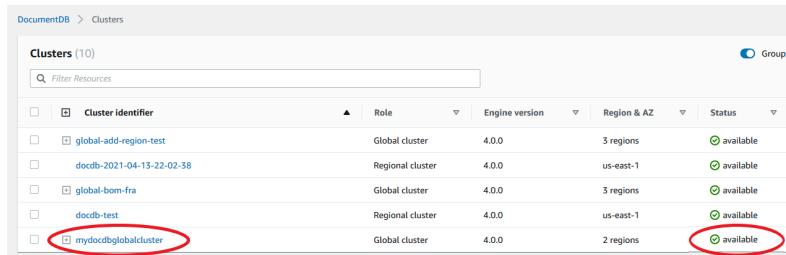
A global cluster needs at least one secondary cluster in a different region than the primary cluster, and you can add up to five secondary clusters. Note that for each secondary cluster that you add, you must reduce the number of replicas allowed in the primary cluster by one. For example, if your global cluster has five secondary regions, your primary cluster can have only ten (rather than fifteen) replicas. For more information, see [Configuration requirements of an Amazon DocumentDB global cluster](#).

Using the AWS Management Console

1. Sign in to the AWS Management Console and open the Amazon DocumentDB console.
2. In the navigation pane, choose **Clusters**.

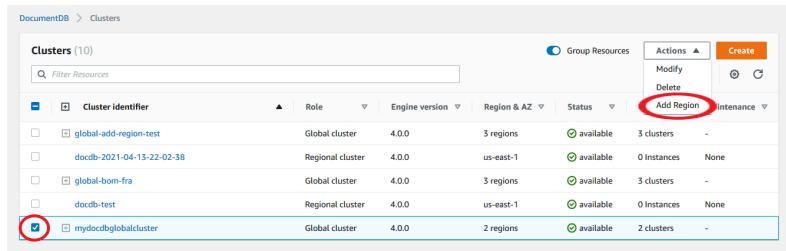


3. Choose the cluster that you would like to add a secondary cluster to. Ensure that the cluster is Available.



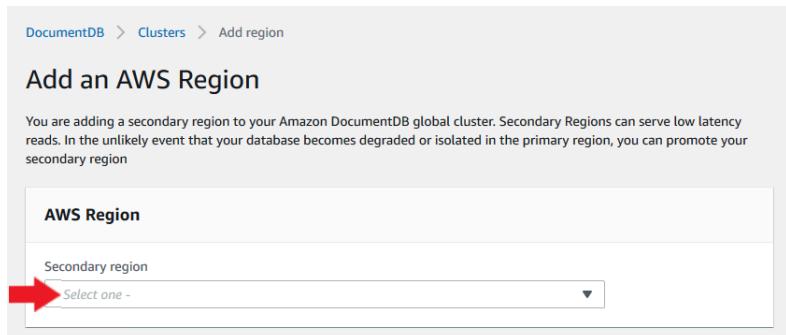
Cluster identifier	Role	Engine version	Region & AZ	Status
global-add-region-test	Global cluster	4.0.0	3 regions	available
docdb-2021-04-13-22-02-38	Regional cluster	4.0.0	us-east-1	available
global-bon-fra	Global cluster	4.0.0	3 regions	available
docdb-test	Regional cluster	4.0.0	us-east-1	available
mydocdbglobalcluster	Global cluster	4.0.0	2 regions	available

4. Select the drop down menu for **Actions** and then choose **Add region**.



Cluster identifier	Role	Engine version	Region & AZ	Status	Actions	Create
global-add-region-test	Global cluster	4.0.0	3 regions	available	Modify	
docdb-2021-04-13-22-02-38	Regional cluster	4.0.0	us-east-1	available	Delete	
global-bon-fra	Global cluster	4.0.0	3 regions	available	Add Region	
docdb-test	Regional cluster	4.0.0	us-east-1	available	Instances	
mydocdbglobalcluster	Global cluster	4.0.0	2 regions	available	-	

5. On the **Add a region** page, choose the secondary region. Note that you can't choose a region that already has a secondary cluster for the same global cluster. Also, it can't be the same region as the primary cluster. If this is the first region you are adding, you will also have to specify a global cluster identifier of your choice.



Add an AWS Region

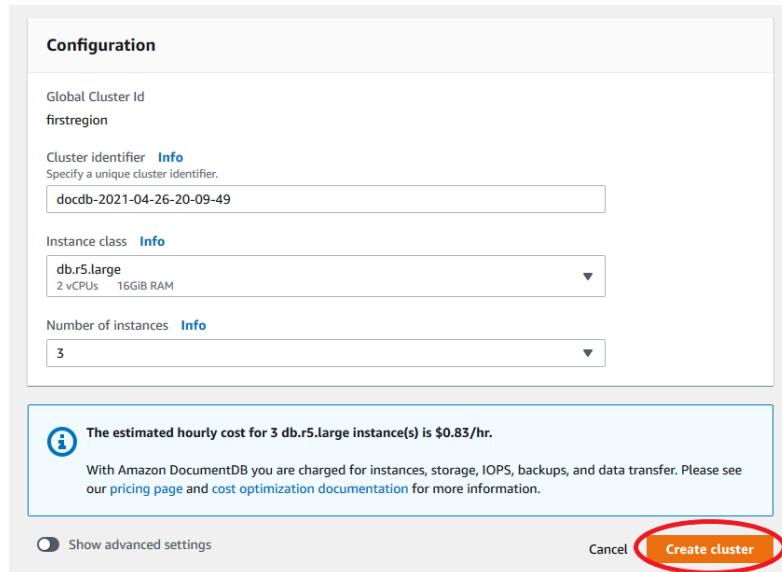
You are adding a secondary region to your Amazon DocumentDB global cluster. Secondary Regions can serve low latency reads. In the unlikely event that your database becomes degraded or isolated in the primary region, you can promote your secondary region

AWS Region

Secondary region

Select one -

6. Complete the remaining fields for the secondary cluster in the new region then select **Create Cluster**. After you finish adding the region, you can see it in the list of **Clusters** in the AWS Management Console.



Using the AWS CLI

- Use the `create-db-cluster` CLI command with the name (`--global-cluster-identifier`) of your global cluster. For other parameters, do the following:
 - For `--region`, choose a different AWS Region than that of your primary region.
 - Choose specific values for the `--engine` and `--engine-version` parameters.
 - For an encrypted cluster, specify your primary AWS Region as the `--source-region` for encryption.

The following example creates a new Amazon DocumentDB cluster and attaches it to the global cluster as a read-only secondary cluster. In the last step, the instance is added to the new cluster.

In the following example, replace each *user input placeholder* with your own information.

For Linux, macOS, or Unix:

```
aws docdb --region secondary-region-id \
  create-db-cluster \
    --db-cluster-identifier cluster-id \
    --global-cluster-identifier global-cluster-id \
    --engine-version version

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
```

For Windows:

```
aws docdb --region secondary-region-id ^
  create-db-cluster ^
    --db-cluster-identifier cluster-id ^
    --global-cluster-identifier global-cluster-id ^
```

```
--engine-version version
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
```

Using a snapshot for your Amazon DocumentDB global cluster

You can restore a snapshot of an Amazon DocumentDB cluster to use as the starting point for your global cluster. To do this, you must restore the snapshot and create a new cluster. This will serve as the primary cluster of your global cluster. You can then add another region to the restored cluster, thus converting it into a global cluster.

Managing an Amazon DocumentDB global cluster

You perform most management operations on the individual clusters that make up a global cluster. When you choose **Group related resources** on the **Clusters** page in the console, you see the primary cluster and secondary clusters grouped under the associated global cluster.

The **Configuration** tab for a global cluster shows the AWS Regions where the clusters are running, the version, and the global cluster identifier.

Topics

- [Modifying an Amazon DocumentDB global cluster \(p. 262\)](#)
- [Modifying parameters an Amazon DocumentDB global cluster \(p. 262\)](#)
- [Removing a cluster from an Amazon DocumentDB global cluster \(p. 262\)](#)
- [Deleting a cluster from an Amazon DocumentDB global cluster \(p. 264\)](#)
- [Creating a headless Amazon DocumentDB cluster in a secondary region \(p. 266\)](#)

Modifying an Amazon DocumentDB global cluster

The **Clusters** page in the AWS Management Console lists all your global clusters, showing the primary cluster and secondary clusters for each one. The global cluster has its own configuration settings. Specifically, it has regions associated with its primary and secondary clusters.

When you make changes to the global cluster, you have a chance to cancel changes.

When you choose **Continue**, you confirm the changes.

Modifying parameters an Amazon DocumentDB global cluster

You can configure the cluster parameter groups independently for each cluster within the global cluster. Most parameters work the same as for other kinds of Amazon DocumentDB clusters. We recommend that you keep settings consistent among all the clusters in a global database. Doing this helps to avoid unexpected behavior changes if you promote a secondary cluster to be the primary.

For example, use the same settings for time zones and character sets to avoid inconsistent behavior if a different cluster takes over as the primary cluster.

Removing a cluster from an Amazon DocumentDB global cluster

There are several situations when you may want to remove clusters from your global cluster. For example, you might want to remove a cluster from a global cluster if the primary cluster becomes

degraded or isolated. It then becomes a standalone provisioned cluster that could be used to create a new global cluster. To learn more, see [Manually recovering a global cluster from an unplanned outage](#).

You also might want to remove clusters because you want to delete a global cluster that you no longer need. You can't delete the global cluster until after you detach all associated clusters, leaving the primary for last. For more information, see [Deleting an Amazon DocumentDB global cluster](#).

Note

When a cluster is detached from the global cluster, it's no longer synchronized with the primary. It becomes a standalone provisioned cluster with full read/write capabilities. Additionally, it is no longer visible in the Amazon DocumentDB console. It is only visible when you select the region in the console that the cluster was located in.

You can remove clusters from your global cluster using the AWS Management Console, the AWS CLI, or the RDS API.

Using the AWS Management Console

1. Sign in to the AWS Management Console and navigate to the Amazon DocumentDB console.
2. Choose **Clusters** on the left side navigation.

3. Expand the global cluster so you can see all the secondary clusters. Select the secondary clusters you wish to remove. Choose **Actions**, and in the menu that drops down, choose **Remove from Global**.

4. A prompt will appear, asking you to confirm that you want to detach the secondary from the global cluster. Choose **Remove and promote** to remove the cluster from the global cluster.

Now that cluster is no longer serving as a secondary and no longer synchronized with the primary cluster. It is a standalone cluster with full read/write capability.

After you remove or delete all secondary clusters, then you can remove the primary cluster the same way. You can't detach or remove the primary cluster from the global cluster until after you have removed all secondary clusters. The global cluster might remain in the Clusters list, with zero regions and AZs. You can delete if you no longer want to use this global cluster.

Using the AWS CLI

To remove a cluster from a global cluster, run the `remove-from-global-cluster` CLI command with the following parameters:

- `--global-cluster-identifier` — The name (identifier) of your global cluster.
- `--db-cluster-identifier` — The name of each cluster to remove from the global cluster.

The following examples first remove a secondary cluster and then the primary cluster from a global cluster.

For Linux, macOS, or 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
```

Repeat the `remove-from-global-cluster --db-cluster-identifier secondary_cluster_ARN` command for each secondary region in your global cluster.

For 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
```

Repeat the `remove-from-global-cluster --db-cluster-identifier secondary_cluster_ARN` command for each secondary region in your global cluster.

Deleting a cluster from an Amazon DocumentDB global cluster

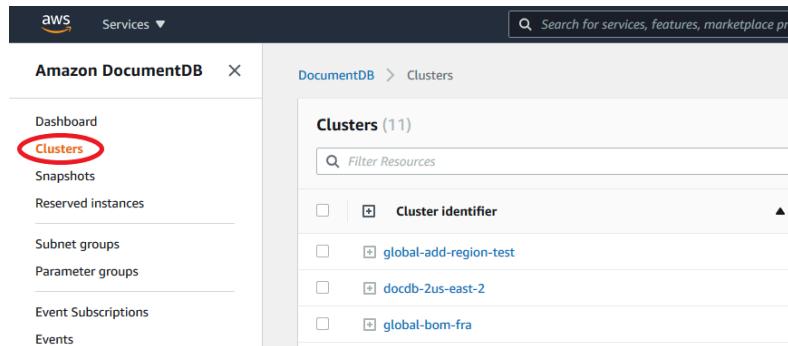
To delete an global cluster, do the following:

- Remove all secondary clusters from the global cluster. Each cluster becomes a standalone cluster. See the previous section, Removing Global Clusters.
- From each standalone cluster, delete all replicas.

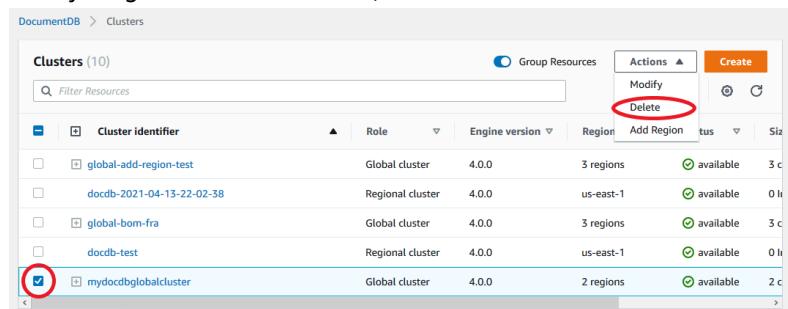
- Remove the primary cluster from the global cluster. This becomes a standalone cluster.
- From the primary cluster, first delete all replicas, then delete the primary instance. Deleting the primary instance from the newly standalone cluster also typically removes both the cluster and the global cluster.

Using the AWS Management Console

1. Sign in to the AWS Management Console and navigate to the Amazon DocumentDB console.
2. Choose **Clusters** and find the global cluster you want to delete.



3. With your global cluster selected, choose **Delete** from the **Actions** menu.



Confirm that all clusters are removed from the global cluster. The global cluster should show zero regions and AZs and a size of zero clusters. If the global cluster contains any clusters, you can't delete it yet. You'll first have to follow the instructions in the previous step, [Removing Global Clusters](#).

Using the AWS CLI

To delete a global cluster, run the `delete-global-cluster` CLI command with the name of the AWS Region and the global cluster identifier, as shown in the following example.

For Linux, macOS, or Unix:

```
aws docdb --region primary_region delete-global-cluster \
--global-cluster-identifier global_cluster_id
```

For Windows:

```
aws docdb --region primary_region delete-global-cluster ^
--global-cluster-identifier global_cluster_id
```

Creating a headless Amazon DocumentDB cluster in a secondary region

Although an Amazon DocumentDB global cluster requires at least one secondary cluster in a different AWS Region than the primary, you can use a headless configuration for the secondary cluster. A headless secondary Amazon DocumentDB cluster is one without an instance. This type of configuration can lower expenses for a global cluster. In an Amazon DocumentDB cluster, compute and storage are decoupled. Without the instance, you're not charged for compute, only for storage. If it's set up correctly, a headless secondary's storage volume is kept in sync with the primary cluster.

You add the secondary cluster as you normally do when creating an Amazon DocumentDB global cluster. However, after the primary cluster begins replication to the secondary, you delete the read-only instance from the secondary cluster. This secondary cluster is now considered "headless" because it no longer has a Instance. Yet, the storage volume is kept in sync with the primary Amazon DocumentDB cluster.

Important

We only recommend headless clusters for customers who can tolerate region-wide failures for 15+ minutes. This is because recovering from a region-wide failure with a headless secondary cluster will require the user to create a new instance after failing over. A new instance can take ~10-15 minutes to become available.

How to Add a Headless Secondary Cluster to Your Global Cluster

1. Sign in to the AWS Management Console and open the [Amazon DocumentDB console](#).
2. Choose **Clusters** on the left side navigation.
3. Choose the global cluster that needs a secondary cluster. Ensure that the primary cluster is Available.
4. For **Actions**, choose **Add region**.
5. On the **Add a region** page, choose the secondary region.

Note

You can't choose a region that already has a secondary cluster for the same global cluster. Also, it can't be the same region as the primary cluster.

6. Complete the remaining fields for the secondary cluster in the new region. These are the same configuration options as for any cluster instance.
7. Add a region. After you finish adding the region to your global cluster, you will see it in the list of Clusters in the AWS Management Console.
8. Check the status of the secondary cluster and its reader instance before continuing, by using the AWS Management Console or the AWS CLI. Here is a sample command if you use the AWS CLI:

```
$ aws docdb describe-db-clusters --db-cluster-identifier secondary-cluster-id --query '*[].[Status]' --output text
```

It can take several minutes for the status of a newly added secondary cluster to change from creating to available. When the cluster is available, you can delete the reader instance.

9. Select the reader instance in the secondary cluster, and then choose **Delete**.
10. After deleting the reader instance, the secondary cluster remains part of the global cluster. It should have no instance associated with it.

Note

You can use this headless secondary Amazon DocumentDB cluster to manually recover your Amazon DocumentDB global cluster from an unplanned outage in the primary region if such an outage occurs.

Connect to an Amazon DocumentDB Global Clusters

How you connect to a global cluster depends on whether you need to write to the cluster or read from the cluster:

- For read-only requests or queries, you connect to the reader endpoint for the cluster in your AWS Region.
- To run data manipulation language (DML) or data definition language (DDL) statements, you connect to the cluster endpoint for the primary cluster. This endpoint might be in a different AWS Region than your application.

When you view a global cluster in the console, you can see all the general-purpose endpoints associated with all of its clusters.

How you connect to a global cluster depends on whether you need to write to the database or read from the database. For DDL, DML and read operations that you would like to serve from the primary region, you should connect to your primary cluster. We recommend that you connect to your primary cluster using the cluster endpoint in replica set mode, with a read preference of `secondaryPreferred=true`. This will route write traffic to your primary cluster's writer instance and read traffic to your primary cluster's replica instance.

For cross region, read only traffic, you should connect to one of your secondary clusters. We recommend that you connect to your secondary cluster using the cluster endpoint in replica set mode. Since all instances are read-only replica instances, you do not need to specify a read preference. To minimize latency, choose whichever reader endpoint is in your region or the region closest to you.

Monitoring Amazon DocumentDB Global Clusters

Amazon DocumentDB (with MongoDB compatibility) integrates with CloudWatch so that you can gather and analyze operational metrics for your clusters. You can monitor these metrics using the CloudWatch console, the Amazon DocumentDB console, the AWS Command Line Interface (AWS CLI), or the CloudWatch API.

To monitor a global cluster, use the following CloudWatch metrics.

Metric	Description
<code>GlobalClusterReplicatedWriteIO</code>	The average number of billed write I/O operations replicated from the cluster volume in the primary AWS Region to the cluster volume in a secondary AWS Region, reported at 5-minute intervals. The number of replicated <code>ReplicatedWriteIOs</code> to each secondary region is the same as the number of in-region <code>VolumeWriteIOPs</code> performed by the primary region.
<code>GlobalClusterDataTransferBytes</code>	The amount of data transferred from the primary cluster's AWS Region to a secondary cluster's AWS Region, measured in bytes.
<code>GlobalClusterReplicationLag</code>	The amount of lag, in milliseconds, when replicating change events from the primary cluster's AWS Region to a secondary cluster's AWS Region

For more information on how to view these metrics, please see [Viewing CloudWatch data](#).

Disaster Recovery and Amazon DocumentDB Global Clusters

By using a global cluster, you can recover from disasters such as region failures quickly. Recovery from disaster is typically measured using values for RTO and RPO.

- **Recovery time objective (RTO)** — The time it takes a system to return to a working state after a disaster. In other words, RTO measures downtime. For a global cluster, RTO can be in the order of minutes.
- **Recovery point objective (RPO)** — The amount of data that can be lost (measured in time). For a global cluster, RPO is typically measured in seconds.
- To recover from an unplanned outage, you can perform a cross-region failover to one of the secondaries in your global cluster. When your global cluster has multiple secondary regions, make sure that you detach all the secondary regions if the primary AWS Region experiences an outage. Then, you promote one of those secondary regions to be the new primary AWS Region. Finally, you create new clusters in each of the other secondary regions and attach those clusters to your global cluster.
- When you promote a secondary cluster to be the primary cluster, you also need to update the endpoints that your applications use to connect to the global cluster. To get a new writer endpoint from a newly promoted cluster, you can convert a former reader endpoint by removing `-ro` from the endpoint string. For example, if a former reader endpoint is `global-16rr-test-cluster-1.cluster-ro-12345678901.us-west-2.docdb.amazonaws.com`, then the new promoted writer endpoint is `global-16rr-test-cluster-1.cluster-cps2igpwyrlwa.us-west-2.rds.amazonaws.com`.

Failover for Amazon DocumentDB Global Clusters

If an entire cluster in one AWS Region becomes unavailable, you can promote another cluster in the global cluster to have read/write capability.

You can manually activate the failover mechanism if a cluster in a different AWS Region is a better choice to be the primary cluster. For example, you might increase the capacity of one of the secondary clusters and then promote it to be the primary cluster. Or the balance of activity among the AWS Regions might change, so that switching the primary cluster to a different AWS Region might give lower latency for write operations.

The following procedure outlines what to do to promote one of the secondary clusters in an DocumentDB global cluster.

To promote a secondary cluster:

1. Stop issuing DML statements and other write operations to the primary cluster in the AWS Region with the outage.
2. Identify a cluster from a secondary AWS Region to use as a new primary cluster. If you have two (or more) secondary AWS Regions in your global cluster, choose the secondary cluster that has the least lag time.
3. Detach your chosen secondary cluster from the global cluster.

Removing a secondary cluster from a global cluster immediately stops the replication from the primary to this secondary and promotes it to standalone provisioned cluster cluster with full read/write capabilities. Any other secondary cluster associated with the primary cluster in the region with the outage are still available and can accept calls from your application. They also consume

resources. Since you are recreating the global cluster, to avoid split-brain and other issues, remove the other secondary clusters before creating the new global cluster in the steps that follow.

For detailed steps for detaching, see [Removing a cluster from an Amazon DocumentDB global cluster \(p. 262\)](#).

4. Reconfigure your application to send all write operations to this now standalone cluster using its new endpoint. If you accepted the provided names when you created the global cluster, you can change the endpoint by removing the -ro from the cluster's endpoint string in your application.

For example, the secondary cluster's endpoint `my-global.cluster-ro-aaaaabbbbb.us-west-1.docdb.amazonaws.com` becomes `my-global.cluster-aaaaabbbbb.us-west-1.docdb.amazonaws.com` when that cluster is detached from the global cluster.

This cluster becomes the primary cluster of a new global cluster when you start adding Regions to it, in the next step.

5. Add an AWS Region to the cluster. When you do this, the replication process from primary to secondary begins.
6. Add more AWS Regions as needed to re-create the topology needed to support your application. Make sure that application writes are sent to the correct cluster before, during, and after making changes such as these, to avoid data inconsistencies among the clusters in the global cluster (split-brain issues).
7. When the outage is resolved and you're ready to assign your original AWS Region as the primary cluster again, perform the same steps in reverse.
8. Remove one of the secondary clusters from the global cluster. This will enable it to serve read/write traffic.
9. Redirect all the write traffic to the primary cluster in the original AWS Region.
10. Add an AWS Region to set up one or more secondary clusters in the same AWS Region as before.

Amazon DocumentDB global clusters can be managed using AWS SDKs, enabling you to create solutions to automate global cluster failover process for Disaster Recovery and Business Continuity Planning use cases. One such solution is made available for our customers under Apache 2.0 licensing and can be accessed from our tools repository [here](#). This solution leverages Amazon Route53 for endpoint management and provides AWS Lambda functions that can be triggered based appropriate events.

Managing Amazon DocumentDB Clusters

To manage an Amazon DocumentDB cluster, you must have an IAM policy with the appropriate Amazon DocumentDB control plane permissions. These permissions allow you to create, modify, and delete clusters and instances. The `AmazonDocDBFullAccess` policy provides all the required permissions for administering an Amazon DocumentDB cluster.

The following topics show how to perform various tasks when working with Amazon DocumentDB clusters, including creating, deleting, modifying, connecting to, and viewing clusters.

Topics

- [Understanding Clusters \(p. 270\)](#)
- [Amazon DocumentDB Cluster Settings \(p. 271\)](#)
- [Determining a Cluster's Status \(p. 272\)](#)
- [Amazon DocumentDB Cluster Lifecycle \(p. 273\)](#)
- [Scaling Amazon DocumentDB Clusters \(p. 300\)](#)
- [Cloning a volume for an Amazon DocumentDB cluster \(p. 302\)](#)

- [Understanding Amazon DocumentDB Cluster Fault Tolerance \(p. 312\)](#)

Understanding Clusters

Amazon DocumentDB separates compute and storage, and offloads data replication and backup to the cluster volume. A cluster volume provides a durable, reliable, and highly available storage layer that replicates data six ways across three Availability Zones. Replicas enable higher data availability and read scaling. Each cluster can scale up to 15 replicas.

Noun	Description	API Operations (Verbs)
Cluster	Consists of one or more instances and a cluster storage volume that manages the data for those instances.	<code>create-db-cluster</code> <code>delete-db-cluster</code> <code>describe-db-clusters</code> <code>modify-db-cluster</code>
Instance	Reading and writing data to the cluster storage volume is done via instances. In a given cluster, there are two types of instances: primary and replica. A cluster always has one primary instance and can have 0–15 replicas.	<code>create-db-instance</code> <code>delete-db-instance</code> <code>describe-db-instances</code> <code>modify-db-instance</code> <code>describe-orderable-db-instance-options</code> <code>reboot-db-instance</code>
Cluster volume	A virtual database storage volume that spans three Availability Zones, with each Availability Zone having two copies of the cluster data.	N/A
Primary instance	Supports both read and write operations, and performs all data modifications to the cluster volume. Each cluster has one primary instance.	N/A
Replica instance	Supports only read operations. Each Amazon DocumentDB cluster can have up to 15 replica instances in addition to the primary instance. Multiple replicas distribute the read workload. By locating replicas in separate Availability Zones, you can also increase database availability.	N/A
Cluster endpoint	An endpoint for an Amazon DocumentDB cluster that connects to the current primary	N/A

Noun	Description	API Operations (Verbs)
	instance for the cluster. Each Amazon DocumentDB cluster has a cluster endpoint and one primary instance.	
Reader endpoint	An endpoint for an Amazon DocumentDB cluster that connects to one of the available replicas for that cluster. Each Amazon DocumentDB cluster has a reader endpoint. If there is more than one replica, the reader endpoint directs each connection request to one of the Amazon DocumentDB replicas.	N/A
Instance endpoint	An endpoint for an instance in an Amazon DocumentDB cluster that connects to a specific instance. Each instance in a cluster, regardless of instance type, has its own unique instance endpoint.	N/A

Amazon DocumentDB Cluster Settings

When you create or modify a cluster, it is important to understand which parameters are immutable and which are modifiable after the cluster has been created. The following table lists all the settings, or parameters, that are specific to a cluster. As specified in the table, some are modifiable, others are not.

Note

These settings should not be confused with Amazon DocumentDB cluster parameter groups and their parameters. For more information about cluster parameter groups, see [Managing Amazon DocumentDB Cluster Parameter Groups \(p. 355\)](#).

Parameter	Modifiable	Notes
DBClusterIdentifier	Yes	<p>Naming constraints:</p> <ul style="list-style-type: none"> Length is [1—63] letters, numbers, or hyphens. First character must be a letter. Cannot end with a hyphen or contain two consecutive hyphens. Must be unique for all clusters across Amazon RDS, Amazon Neptune, and Amazon DocumentDB per AWS account, per Region.
Engine	No	Must be docdb.
BackupRetentionPeriod	Yes	Must be between [1-35] days.
DBClusterParameterGroupName	Yes	<p>Naming constraints:</p> <ul style="list-style-type: none"> Length is [1—255] alphanumeric characters.

Parameter	Modifiable	Notes
		<ul style="list-style-type: none"> • First character must be a letter. • Cannot end with a hyphen or contain two consecutive hyphens.
DBSubnetGroupName	No	After a cluster has been created, you cannot modify the cluster's subnet.
EngineVersion	No	Must be 4.0.0.
KmsKeyId	No	If you choose to encrypt your cluster, you cannot change the AWS KMS key that you used to encrypt your cluster.
MasterUsername	No	<p>After a cluster has been created, you cannot modify the MasterUsername.</p> <p>Naming constraints:</p> <ul style="list-style-type: none"> • Length is [1—63] alphanumeric characters. • First character must be a letter. • Cannot be a word reserved by the database engine.
MasterUserPassword	Yes	<p>Constraints:</p> <ul style="list-style-type: none"> • Length is [8—100] printable ASCII characters. • Can use any printable ASCII characters except for the following: <ul style="list-style-type: none"> • / (forward slash) • " (double quotation mark) • @ (at symbol)
Port	Yes	The port number applies to all instances in the cluster.
PreferredBackupWindow	Yes	
PreferredMaintenanceWindow	Yes	
StorageEncrypted	No	If you choose to encrypt your cluster, it cannot be unencrypted.
Tags	Yes	
VpcSecurityGroupIds	No	After a cluster has been created, you cannot modify the VPC that the cluster resides in.

Determining a Cluster's Status

You can determine a cluster's status using the AWS Management Console or AWS CLI.

Using the AWS Management Console

Use the following procedure to see the status of your Amazon DocumentDB cluster using the AWS Management Console

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Clusters**.
3. In the **Cluster identifier** column, find the name of the cluster that you are interested in. Then, to find the status of the cluster, read across that row to the **Status** column, as shown below.

Clusters (1)			
<input type="text"/> Filter clusters			
Cluster identifier	Engine version	Status	Instances
docdb-2020-10-23-22-23-28	docdb 3.6.0	available	1

Using the AWS CLI

Use the `describe-db-clusters` operation to see the the status of your Amazon DocumentDB cluster using the AWS CLI.

The following code finds the status of the cluster `sample-cluster`.

For Linux, macOS, or Unix:

```
aws docdb describe-db-clusters \
--db-cluster-identifier sample-cluster \
--query 'DBClusters[*].[DBClusterIdentifier,Status]'
```

For Windows:

```
aws docdb describe-db-clusters ^
--db-cluster-identifier sample-cluster ^
--query 'DBClusters[*].[DBClusterIdentifier,Status]'
```

Output from this operation looks something like the following (JSON format).

```
[  
  [  
    "sample-cluster",  
    "available"  
  ]  
]
```

Amazon DocumentDB Cluster Lifecycle

The lifecycle of an Amazon DocumentDB cluster includes creating, describing, modifying, and deleting the cluster. This section provides information about how to complete these processes.

Topics

- [Creating an Amazon DocumentDB Cluster \(p. 274\)](#)
- [Describing Amazon DocumentDB Clusters \(p. 283\)](#)
- [Modifying an Amazon DocumentDB Cluster \(p. 287\)](#)
- [Determining Pending Maintenance \(p. 290\)](#)
- [Updating a Cluster's Engine Version \(p. 291\)](#)

- [Stopping and Starting an Amazon DocumentDB Cluster \(p. 293\)](#)
- [Deleting an Amazon DocumentDB Cluster \(p. 296\)](#)

Creating an Amazon DocumentDB Cluster

An Amazon DocumentDB cluster consists of instances and a cluster volume that represents the data for the cluster. The cluster volume is replicated six ways across three Availability Zones as a single, virtual volume. The cluster contains a primary instance and, optionally, up to 15 replica instances.

The following sections show how to create an Amazon DocumentDB cluster using either the AWS Management Console or the AWS CLI. You can then add additional replica instances for that cluster. When you use the console to create your Amazon DocumentDB cluster, a primary instance is automatically created for you at the same time. If you use the AWS CLI to create your Amazon DocumentDB cluster, after the cluster's status is *available*, you must then create the primary instance for that cluster.

Prerequisites

The following are prerequisites for creating an Amazon DocumentDB cluster.

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open <https://portal.aws.amazon.com/billing/signup>.
2. Follow the online instructions.

Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, [assign administrative access to an administrative user](#), and use only the root user to perform [tasks that require root user access](#).

VPC Prerequisites

You can only create an Amazon DocumentDB cluster in an Amazon Virtual Private Cloud (Amazon VPC). Your Amazon VPC must have at least one subnet in each of at least two Availability Zones in order for you to use it with an Amazon DocumentDB cluster. By distributing your cluster instances across Availability Zones, you ensure that instances are available in your cluster in the unlikely case of an Availability Zone failure.

Subnet Prerequisites

When creating an Amazon DocumentDB cluster, you must choose a VPC and corresponding subnet group within that VPC to launch your cluster. Subnets determine the Availability Zone and IP range within that Availability Zone that you want to use to launch an instance. For the purposes of this discussion, the terms *subnet* and *Availability Zone* are used interchangeably. A subnet group is a named set of subnets (or Availability Zones). What a subnet group allows you to do is specify the Availability Zones that you want to use to for launching Amazon DocumentDB instances. For example, in a cluster with three instances, it is recommended for high availability that each of those instances is provisioned in separate Availability Zones. Thus, if a single Availability Zone goes down, it only affects a single instance.

Amazon DocumentDB instances can currently be provisioned in up to three Availability Zones. Even if a subnet group has more than three subnets, you can only use three of those subnets to create an Amazon DocumentDB cluster. As a result, it is suggested that when you create a subnet group, only choose the three subnets that you want to deploy your instances to. In US East (N. Virginia), your subnet group can

have six subnets (or Availability Zones). However, when an Amazon DocumentDB cluster is provisioned, Amazon DocumentDB chooses three of those Availability Zones that it uses to provision instances.

For example, suppose that when you are creating a cluster, Amazon DocumentDB chooses the Availability Zones {1A, 1B, and 1C}. If you try to create an instance in Availability Zone {1D}, the API call fails. However, if you choose to create an instance without specifying a particular Availability Zone, then Amazon DocumentDB chooses an Availability Zone on your behalf. Amazon DocumentDB uses an algorithm to load balance the instances across Availability Zones to help you achieve high availability. For example, if three instances are provisioned, by default, they are provisioned across three Availability Zones and are not provisioned all in a single Availability Zone.

Recommendations:

- Unless you have a specific reason, always create a subnet group with three subnets. Doing so helps ensure that clusters with three or more instances can achieve higher availability as instances are provisioned across three Availability Zones.
- Always spread instances across multiple Availability Zones to achieve high availability. Never place all instances for a cluster in a single Availability Zone.
- Because failover events can happen at any time, you should not assume that a primary instance or replica instances are always in a particular Availability Zone.

Additional Prerequisites

The following are some additional prerequisites for creating an Amazon DocumentDB cluster:

- If you are connecting to AWS using AWS Identity and Access Management (IAM) credentials, your IAM account must have IAM policies that grant the permissions that are required to perform Amazon DocumentDB operations.

If you are using an IAM account to access the Amazon DocumentDB console, you must first sign in to the AWS Management Console with your IAM account. Then go to the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.

- If you want to tailor the configuration parameters for your cluster, you must specify a cluster parameter group and parameter group with the required parameter settings. For information about creating or modifying a cluster parameter group or parameter group, see [Managing Amazon DocumentDB Cluster Parameter Groups \(p. 355\)](#).
- You must determine the TCP/IP port number that you want to specify for your cluster. The firewalls at some companies block connections to the default ports for Amazon DocumentDB. If your company firewall blocks the default port, choose another port for your cluster. All instances in a cluster use the same port.

Creating a Cluster and Primary Instance Using the AWS Management Console

The following procedures describe how to use the console to launch an Amazon DocumentDB cluster with one or more instances.

Create a Cluster: Using Default Settings

To create a cluster with instances using the default settings using the AWS Management Console

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. If you want to create your cluster in an AWS Region other than the US East (N. Virginia) Region, choose the Region from the list in the upper-right section of the console.
3. In the navigation pane, choose **Clusters**, and then choose **Create**.

Tip

If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.

4. On the **Create Amazon DocumentDB cluster** page, complete the **Configuration** pane.
 - a. **Cluster identifier**—Accept the Amazon DocumentDB provided name, or enter a name for your cluster; for example, **sample-cluster**.

Cluster naming constraints:

 - Length is [1—63] letters, numbers, or hyphens.
 - First character must be a letter.
 - Cannot end with a hyphen or contain two consecutive hyphens.
 - Must be unique for all clusters across Amazon RDS, Neptune, and Amazon DocumentDB per AWS account, per Region.
 - b. **Engine version**—Accept the default engine version of 4.0.0, or optionally choose 3.6.0.
 - c. **Instance class**—Accept the default db.r5.large, or choose the instance class that you want from the list.
 - d. **Number of instances**—In the list, choose the number of instances that you want to be created with this cluster. The first instance is the primary instance, and all other instances are read-only replica instances. You can add and delete instances later if you need to. By default, an Amazon DocumentDB cluster launches with three instances (one primary and two replicas).
5. Complete the **Authentication** pane.
 - a. **Master username**—Enter a name for the master user. To log in to your cluster, you must use the master user name.

Master user naming constraints:

 - Length is [1—63] alphanumeric characters.
 - First character must be a letter.
 - Cannot be a word reserved by the database engine.
 - b. **Master password**—Enter a password for the master user, and then confirm it. To log in to your cluster, you must use the master password.

Master password constraints:

 - Length is [8-100] printable ASCII characters.
 - Can use any printable ASCII characters except for the following:
 - / (forward slash)
 - " (double quotation mark)
 - @ (at symbol)
6. At the bottom of the screen, choose one of the following:
 - To create the cluster now, choose **Create cluster**.
 - To not create the cluster, choose **Cancel**.
 - To further configure the cluster before creating, choose **Show additional configurations**, and then continue at [Create a Cluster: Additional Configurations \(p. 278\)](#).

The configurations covered in the **Additional Configurations** section are as follows:

- **Network settings**—The default is to use the default VPC security group.
- **Cluster options**—The default is to use port 27017 and the default parameter group.

- **Encryption**—The default is to enable encryption using the (default) aws/rds key.

Important

After a cluster is encrypted, it cannot be unencrypted.

- **Backup**—The default is to retain backups for 1 day and let Amazon DocumentDB choose the backup window.
- **Log exports**—The default is to not export audit logs to CloudWatch Logs.
- **Maintenance**—The default is to let Amazon DocumentDB choose the maintenance window.
- **Deletion protection**—Protect your cluster from accidental deletion. Default for cluster created using the console is *enabled*.

If you accept the default settings now, you can change most of them later by modifying the cluster.

7. Enable inbound connection for your cluster's security group.

If you did not change the defaults settings for your cluster, you created a cluster using the default security group for the default VPC in the given region. To connect to Amazon DocumentDB, you must enable inbound connections on port 27017 (or the port of your choice) for your cluster's security group.

To add an inbound connection to your cluster's security group

- Sign in to the AWS Management Console and open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
- In the **Resources** section of the main window, choose **Security groups**.

Resources					
You are using the following Amazon EC2 resources in the EU West (Ireland) region:					
0 Running Instances	0 Elastic IPs				
0 Dedicated Hosts	0 Snapshots				
0 Volumes	0 Load Balancers				
0 Key Pairs	1 Security Groups				
0 Placement Groups					

- From the list of security groups locate the security group you used when creating your cluster (it is most likely the *default* security group) and choose the box to the left of the security group's name.

Name	Group ID	Group Name	VPC ID
<input checked="" type="checkbox"/> sg-06b2ad61	sg-06b2ad61	default	vpc-d833a4bc
<input type="checkbox"/> sg-07443a112c70a5282	sg-07443a112c70a5282	test-sg	vpc-d833a4bc

- From the **Actions** menu, choose **Edit inbound rules** then choose or enter the rule constraints.
 - Type**—From the list, choose the protocol to open to network traffic.
 - Protocol**—From the list, choose the type of protocol.
 - Port Range**—For a custom rule, enter a port number or port range. Be sure that the port number or range includes the port you specified when you created your cluster (default: 27017).
 - Source**—Specifies the traffic that can reach your instance. From the list, choose the traffic source. If you choose **Custom**, specify a single IP address or an IP address range in CIDR notation (e.g., 203.0.113.5/32).
 - Description**—Enter a description for this rule.
 - When finished creating the rule, choose **Save**.

Create a Cluster: Additional Configurations

If you want to accept the default settings for your cluster, you can skip the following steps and choose **Create cluster**.

1. Complete the **Network settings** pane.

Network settings

a. **Virtual Private Cloud (VPC)**—In the list, choose the Amazon VPC that you want to launch this cluster in.

b. **Subnet group**—In the list, choose the subnet group you want to use for this cluster.

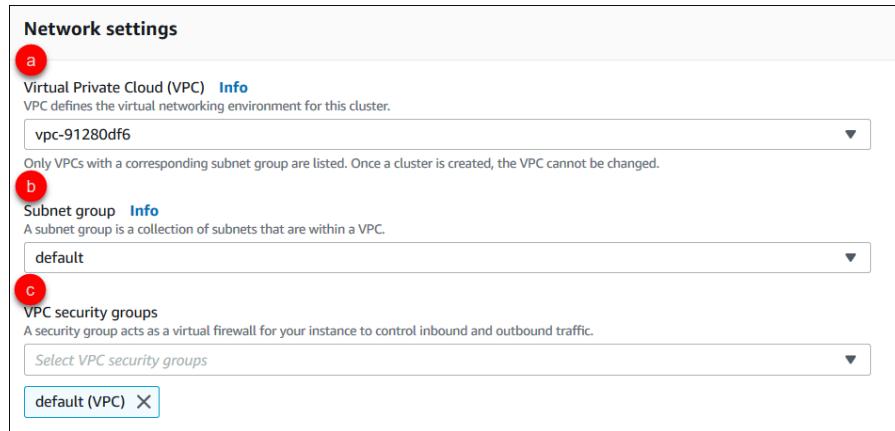
c. **VPC security groups**—In the list, choose the VPC security group for this cluster.

vpc-91280df6

default

Select VPC security groups

default (VPC) X



- a. **Virtual Private Cloud (VPC)**—In the list, choose the Amazon VPC that you want to launch this cluster in.

- b. **Subnet group**—In the list, choose the subnet group you want to use for this cluster.

- c. **VPC security groups**—In the list, choose the VPC security group for this cluster.

2. Complete the **Cluster options** pane.

Cluster options

Port
TCP/IP port that is used to connect to the cluster.
27017

Cluster parameter group **Info**
default.docdb4.0



- a. **Data base port**—Use the up and down arrows to set the TCP/IP port that applications will use to connect to your instance.

- b. **Cluster parameter group**—In the list of parameter groups, choose the cluster parameter group for this cluster.

3. Complete the **Encryption** pane.

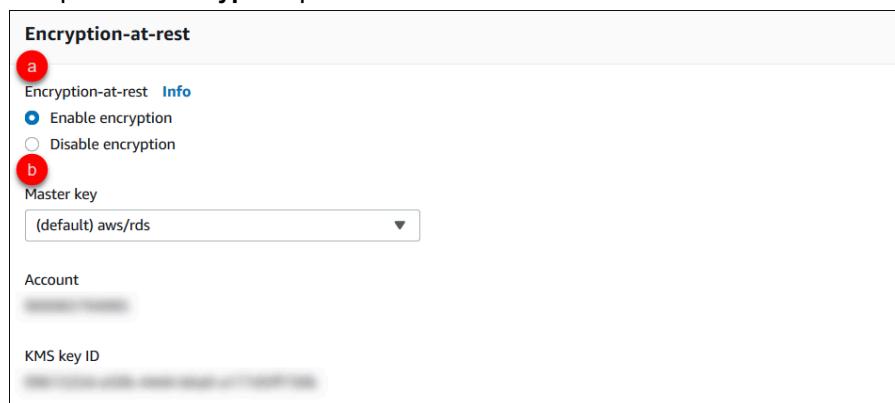
Encryption-at-rest

a. **Encryption-at-rest** **Info**
 Enable encryption
 Disable encryption

b. **Master key**
(default) aws/rds

Account

KMS key ID

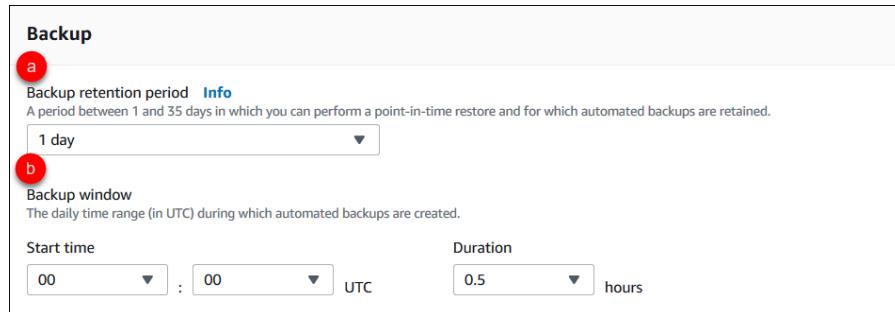


- a. **Encryption-at-rest**—Choose one of the following:
 - **Enable encryption**—Default. All data at rest is encrypted. If you choose to encrypt your data, you cannot undo this action.
 - **Disable encryption**—Your data is not encrypted.
- b. **Master key**—This is only available if you are encrypting your data. In the list, choose the key that you want to use for encrypting the data in this cluster. The default is (default) aws/rds.

If you chose **Enter a key ARN**, you must enter an Amazon Resource Name (ARN) for the key.

Backup retention period—In the list, choose the number of days that you want automatic backups retained.

4. Complete the **Backup** pane.



The screenshot shows the 'Backup' pane with two main sections highlighted by red circles:

- a.** **Backup retention period**: A dropdown menu set to "1 day".
- b.** **Backup window**: Fields for "Start time" (00:00 UTC) and "Duration" (0.5 hours).

- a. **Backup retention period**—In the list, choose the number of days to keep automatic backups of this cluster before deleting them.
- b. **Backup window**—Set the daily time and duration during which Amazon DocumentDB is to make backups of this cluster.
 - i. **Start time**—In the first list, choose the start time hour (UTC) for starting your automatic backups. In the second list, choose the minute of the hour that you want automatic backups to begin.
 - ii. **Duration**—In the list, choose the number of hours to be allocated to creating automatic backups.

5. Complete the **Log exports** pane by selecting the types of logs you want to export to CloudWatch Logs.



The screenshot shows the 'Log exports' pane with two checkboxes listed:

- Audit logs**
- Profiler logs**

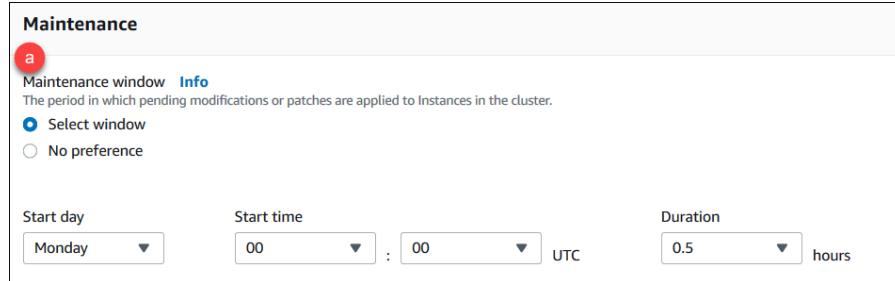
 The 'Audit logs' checkbox is highlighted with a red box.

- **Audit logs**—Select this option to enable exporting audit logs to Amazon CloudWatch Logs. If you select **Audit logs**, you must enable audit_logs in the cluster's custom parameter group. For more information, see [Auditing Amazon DocumentDB Events \(p. 205\)](#).
- **Profiler logs**—Select this option to enable exporting operation profiler logs to Amazon CloudWatch Logs. If you select **Profiler logs**, you must also modify the following parameters in the cluster's custom parameter group:
 - **profiler**—Set to enabled.
 - **profiler_threshold_ms**—Set to a value [0-INT_MAX] to set the threshold for profiling operations.

- **profiler_sampling_rate**—Set to a value [0.0-1.0] to set the percentage of slow operations to profile.

For more information, see [Profiling Amazon DocumentDB Operations \(p. 427\)](#).

6. Complete the **Maintenance** pane.

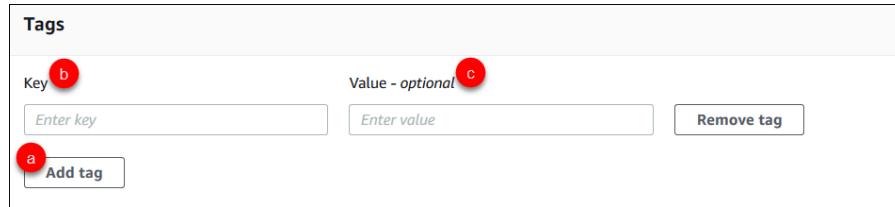


The screenshot shows the 'Maintenance' pane with the following configuration:

- Maintenance window**: Set to **Select window**.
- Start day**: Monday
- Start time**: 00 : 00 UTC
- Duration**: 0.5 hours

- Choose one of the following
 - Select window**—You can specify the day of the week, UTC start time, and duration for Amazon DocumentDB to perform maintenance on your cluster.
 - Start day**—In the list, choose the day of the week to start cluster maintenance.
 - Start time**—In the lists, choose the hour and minute (UTC) to start maintenance.
 - Duration**—In the list, choose how much time to allocate for cluster maintenance. If maintenance cannot be finished in the specified time, the maintenance process will continue past the specified time until finished.
 - No preference**—Amazon DocumentDB chooses the day of the week, start time, and duration for performing maintenance.

7. If you want to add one or more tags to this cluster, complete the **Tags** pane.



The screenshot shows the 'Tags' pane with the following fields:

- Key**: (labeled 'a')
- Value - optional**: (labeled 'c')
- Add tag** button (labeled 'b')
- Remove tag** button

For each tag you want to add to the cluster, repeat the following steps. You may have up to 10 on a cluster.

- Choose **Add tags**.
- Type the tag's **Key**.
- Optionally type the tag's **Value**.

To remove a tag, choose **Remove tag**.

8. **Deletion Protection** is enabled by default when you create a cluster using the console. To disable deletion protection, clear **Enable deletion protection**. When enabled, deletion protection prevents a cluster from being deleted. To delete a deletion protected cluster, you must first modify the cluster to disable deletion protection.



The screenshot shows the 'Deletion protection' pane with the following setting:

- Enable deletion protection**:

A note below the checkbox states: "Protects the cluster from being accidentally deleted. While this option is enabled, you can't delete the cluster."

For more information about deletion protection, see [Deleting an Amazon DocumentDB Cluster \(p. 296\)](#).

9. To create the cluster, choose **Create cluster**. Otherwise, choose **Cancel**.

Creating a Cluster Using the AWS CLI

The following procedures describe how to use the AWS CLI to launch an Amazon DocumentDB cluster and create an Amazon DocumentDB replica.

Parameters

- **--db-cluster-identifier**—Required. A lowercase string that identifies this cluster.

Cluster Naming Constraints:

- Length is [1–63] letters, numbers, or hyphens.
- First character must be a letter.
- Cannot end with a hyphen or contain two consecutive hyphens.
- Must be unique for all clusters (across Amazon RDS, Amazon Neptune, and Amazon DocumentDB) per AWS account, per Region.

- **--engine**—Required. Must be **docdb**.

- **--deletion-protection | --no-deletion-protection**—Optional. When deletion protection is enabled, it prevents a cluster from being deleted. When you use the AWS CLI, the default setting is to have deletion protection disabled.

For more information about deletion protection, see [Deleting an Amazon DocumentDB Cluster \(p. 296\)](#).

- **--master-username**—Required. The user name used to authenticate the user.

Master User Naming Constraints:

- Length is [1-63] alphanumeric characters.
- First character must be a letter.
- Cannot be a word reserved by the database engine.

- **--master-user-password**—Required. The user's password used to authenticate the user.

Master Password Constraints:

- Length is [8-100] printable ASCII characters.
- Can use any printable ASCII characters except for the following:
 - / (forward slash)
 - " (double quotation mark)
 - @ (at symbol)

For additional parameters, see [CreateDBCluster \(p. 584\)](#).

To launch an Amazon DocumentDB cluster using the AWS CLI

To create an Amazon DocumentDB cluster, call the `create-db-cluster` AWS CLI. The following AWS CLI command creates an Amazon DocumentDB cluster named `sample-cluster` with deletion protection enabled. For more information on deletion protection, see [Deleting an Amazon DocumentDB Cluster \(p. 296\)](#).

Also, `--engine-version` is an optional parameter that defaults to the latest major engine version. The current major engine version is 4.0.0. When new major engine versions are released, the default engine version for `--engine-version` will be updated to reflect the lastest major engine version. As a result, for production workloads, and especially those that are dependent on scripting, automation, or AWS CloudFormation templates, we recommend that you explicitly specify the `--engine-version` to the intended major version.

Note

If a `db-subnet-group-name` or `vpc-security-group-id` is not specified, Amazon DocumentDB will use the default subnet group and Amazon VPC security group for the given region.

For Linux, macOS, or Unix:

```
aws docdb create-db-cluster \
    --db-cluster-identifier sample-cluster \
    --engine docdb \
    --engine-version 4.0.0 \
    --deletion-protection \
    --master-username masteruser \
    --master-user-password password
```

For Windows:

```
aws docdb create-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --engine docdb ^
    --engine-version 4.0.0 ^
    --deletion-protection ^
    --master-username masteruser ^
    --master-user-password password
```

Output from this operation looks something like the following (JSON format).

```
{  
    "DBCluster": {  
        "StorageEncrypted": false,  
        "DBClusterMembers": [],  
        "Engine": "docdb",  
        "DeletionProtection" : "enabled",  
        "ClusterCreateTime": "2018-11-26T17:15:19.885Z",  
        "DBSubnetGroup": "default",  
        "EngineVersion": "4.0.0",  
        "MasterUsername": "masteruser",  
        "BackupRetentionPeriod": 1,  
        "DBClusterArn": "arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster",  
        "DBClusterIdentifier": "sample-cluster",  
        "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": "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"
}
```

It takes several minutes to create the cluster. You can use the AWS Management Console or AWS CLI to monitor the status of your cluster. For more information, see [Monitoring an Amazon DocumentDB Cluster's Status \(p. 402\)](#).

Important

When you use the AWS CLI to create an Amazon DocumentDB cluster, no instances are created. Consequently, you must explicitly create a primary instance and any replica instances that you need. You can use either the console or AWS CLI to create the instances. For more information, see [Adding an Amazon DocumentDB Instance to a Cluster \(p. 319\)](#).

For more information, see [CreateDBCluster](#) in the *Amazon DocumentDB API Reference*.

Describing Amazon DocumentDB Clusters

You can use either Amazon DocumentDB Management Console or the AWS CLI to see details such as connection endpoints, security groups, VPCs, and parameter groups pertaining to your Amazon DocumentDB clusters.

For more information, see the following:

- [Monitoring an Amazon DocumentDB Cluster's Status \(p. 402\)](#)
- [Finding a Cluster's Endpoints \(p. 379\)](#)

Using the AWS Management Console

Use the following procedure to view the details of a specified Amazon DocumentDB cluster using the console.

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Clusters**.

Tip

If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.

3. In the list of clusters, choose the name of the cluster that you want to see the details of. The information about the cluster is organized into the following groupings:
 - **Summary** — General information about the cluster, including the engine version, cluster status, pending maintenance, and the status of its parameter group.

- **Connectivity & Security** —The **Connect** section lists connection endpoints to connect to this cluster with the mongo shell or with an application. The **Security Groups** section lists the security groups associated with this cluster and their VPC ID and descriptions.
- **Configuration** — The **Cluster details** section lists details about the cluster, including the cluster's Amazon Resource Name (ARN), endpoint, and parameter group. It also lists the cluster's backup information, maintenance details, and security and network settings. The **Cluster instances** section lists the instances that belong to this cluster with each instance's role and cluster parameter group status.
- **Monitoring** — The Amazon CloudWatch Logs metrics for this cluster. For more information, see [Monitoring Amazon DocumentDB with CloudWatch \(p. 414\)](#).
- **Events & tags** — The **Recent events** section lists the recent events for this cluster. Amazon DocumentDB keeps a record of events that relate to your clusters, instances, snapshots, security groups, and cluster parameter groups. This information includes the date, time, and message associated with each event. The **Tags** section lists the tags attached to this cluster.

Using the AWS CLI

To view the details of your Amazon DocumentDB clusters using the AWS CLI, use the `describe-db-clusters` command as shown in the examples below. For more information, see [DescribeDBClusters](#) in the *Amazon DocumentDB Resource Management API Reference*.

Note

For certain management features such as cluster and instance lifecycle management, Amazon DocumentDB leverages operational technology that is shared with Amazon RDS. The `filterName=engine,Values=docdb` filter parameter returns only Amazon DocumentDB clusters.

Example

Example 1: List all Amazon DocumentDB clusters

The following AWS CLI code lists the details for all Amazon DocumentDB clusters in a region.

```
aws docdb describe-db-clusters --filter Name=engine,Values=docdb
```

Output from this operation looks something like the following.

```
{  
    "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

Example 2: List all details for a specified Amazon DocumentDB cluster

The following AWS CLI code lists the details for the cluster sample-cluster.

For Linux, macOS, or Unix:

```
aws docdb describe-db-clusters \
--filter Name=engine,Values=docdb \
--db-cluster-identifier sample-cluster
```

For Windows:

```
aws docdb describe-db-clusters ^
--filter Name=engine,Values=docdb ^
--db-cluster-identifier sample-cluster
```

Output from this operation looks something like the following.

```
{
    "DBClusters": [
        {
            "AvailabilityZones": [
                "us-east-1c",
                "us-east-1b",
                "us-east-1a"
            ],
            "BackupRetentionPeriod": 1,
            "DBClusterIdentifier": "sample-cluster",
            "DBClusterParameterGroup": "sample-parameter-group",
            "DBSubnetGroup": "default",
            "Status": "available",
            "EarliestRestorableTime": "2020-03-09T00:04:26.844Z",
            "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": "4.0.0",
            "LatestRestorableTime": "2020-03-10T20:38:17.456Z",
            "Port": 8453
        }
    ]
}
```

```

    "Port": 27017,
    "MasterUsername": "<user-name>",
    "PreferredBackupWindow": "00:00-00:30",
    "PreferredMaintenanceWindow": "tue:09:50-tue:10:20",
    "DBClusterMembers": [
        {
            "DBInstanceIdentifier": "sample-instance-1",
            "IsClusterWriter": true,
            "DBClusterParameterGroupStatus": "in-sync",
            "PromotionTier": 1
        },
        {
            "DBInstanceIdentifier": "sample-instance-2",
            "IsClusterWriter": false,
            "DBClusterParameterGroupStatus": "in-sync",
            "PromotionTier": 1
        },
    ],
    "VpcSecurityGroups": [
        {
            "VpcSecurityGroupId": "sg-abcd0123",
            "Status": "active"
        }
    ],
    "HostedZoneId": "ABCDEFGHIJKLMN",
    "StorageEncrypted": true,
    "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "DbClusterResourceId": "cluster-ABCDEFGHIJKLMN0PQRSTUVWXYZ",
    "DBClusterArn": "arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster",
    "AssociatedRoles": [],
    "ClusterCreateTime": "2020-01-10T22:13:38.261Z",
    "EnabledCloudwatchLogsExports": [
        "profiler"
    ],
    "DeletionProtection": true
]
}

```

Example

Example 3: List specific details for a Amazon DocumentDB cluster

To list a subset of the clusters' details using the AWS CLI, add a `--query` that specifies which cluster members the `describe-db-clusters` operation is to list. The `--db-cluster-identifier` parameter is the identifier for the particular cluster that you want to display the details of. For more information on queries, see [How to Filter the Output with the `--query` Option](#) in the *AWS Command Line Interface User Guide*.

The following example lists the instances in an Amazon DocumentDB cluster.

For Linux, macOS, or Unix:

```
aws docdb describe-db-clusters \
--filter Name=engine,Values=docdb \
--db-cluster-identifier sample-cluster \
--query 'DBClusters[*].[DBClusterMembers]'
```

For Windows:

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

```
--filter Name=engine,Values=docdb ^
--db-cluster-identifier sample-cluster ^
--query 'DBClusters[*].[DBClusterMembers]'
```

Output from this operation looks something like the following.

```
[  
  [  
    [  
      {  
        "DBInstanceIdentifier": "sample-instance-1",  
        "IsClusterWriter": true,  
        "DBClusterParameterGroupStatus": "in-sync",  
        "PromotionTier": 1  
      },  
      {  
        "DBInstanceIdentifier": "sample-instance-2",  
        "IsClusterWriter": false,  
        "DBClusterParameterGroupStatus": "in-sync",  
        "PromotionTier": 1  
      }  
    ]  
  ]  
]
```

Modifying an Amazon DocumentDB Cluster

To modify a cluster, the cluster must be in the *available* state. You cannot modify a cluster that is stopped. If the cluster is stopped, first start the cluster, wait for the cluster to become *available*, and then make the desired modifications. For more information, see [Stopping and Starting an Amazon DocumentDB Cluster \(p. 293\)](#).

Using the AWS Management Console

Use the following procedure to modify a specific Amazon DocumentDB cluster using the console.

To modify an Amazon DocumentDB cluster

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Clusters**.

Tip

If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.

3. Specify the cluster that you want to modify by choosing the button to the left of the cluster's name.
4. Choose **Actions**, and then choose **Modify**.
5. In the **Modify Cluster: <cluster-name>** pane, make the changes that you want. You can make changes in the following areas:
 - **Cluster specifications**—The cluster's name, security groups, and password.
 - **Cluster options**—The cluster's port and parameter group.
 - **Backup**—The cluster's backup retention period and backup window.
 - **Log exports**—Enable or disable exporting audit or profiler logs.
 - **Maintenance**—Set the cluster's maintenance window.

- **Deletion protection**—Enable or disable deletion protection on the cluster. Deletion protection is enabled by default.
6. When you're finished, choose **Continue** to view a summary of your changes.
 7. If you are satisfied with your changes, you can choose **Modify cluster** to modify your cluster. Alternatively, you can choose **Back** or **Cancel** to edit or cancel your changes, respectively.

It takes a few minutes for your changes to be applied. You can use the cluster only when its status is *available*. You can monitor the cluster's status using the console or AWS CLI. For more information, see [Monitoring an Amazon DocumentDB Cluster's Status \(p. 402\)](#).

Using the AWS CLI

Use the `modify-db-cluster` operation to modify the specified cluster using the AWS CLI. For more information, see [ModifyDBCluster](#) in the *Amazon DocumentDB API Reference*.

Parameters

- **--db-cluster-identifier**—Required. The identifier of the Amazon DocumentDB cluster that you are going to modify.
- **--backup-retention-period**—Optional. The number of days for which automated backups are retained. Valid values are 1–35.
- **--db-cluster-parameter-group-name**—Optional. The name of the cluster parameter group to use for the cluster.
- **--master-user-password**—Optional. The new password for the master database user.

Password constraints:

- Length is [8–100] printable ASCII characters.
- Can use any printable ASCII characters except for the following:
 - / (forward slash)
 - " (double quotation mark)
 - @ (at symbol)
- **--new-db-cluster-identifier**—Optional. The new cluster identifier for the cluster when renaming a cluster. This value is stored as a lowercase string.

Naming constraints:

- Length is [1–63] letters, numbers, or hyphens.
- First character must be a letter.
- Cannot end with a hyphen or contain two consecutive hyphens.
- Must be unique for all clusters across Amazon RDS, Amazon Neptune, and Amazon DocumentDB per AWS account, per Region.
- **--preferred-backup-window**—Optional. The daily time range during which automated backups are created, in Universal Coordinated Time (UTC).
 - Format: hh24:mm–hh24:mm
- **--preferred-maintenance-window**—Optional. The weekly time range during which system maintenance can occur, in UTC.
 - Format: ddd:hh24:mm–ddd:hh24:mm
 - Valid days: Sun, Mon, Tue, Wed, Thu, Fri, and Sat.
- **--deletion-protection** or **--no-deletion-protection**—Optional. Whether deletion protection should be enabled on this cluster. Deletion protection prevents a cluster from being accidentally deleted until the cluster is modified to disable deletion protection. For more information, see [Deleting an Amazon DocumentDB Cluster \(p. 296\)](#).

- **--apply-immediately or --no-apply-immediately**—Use `--apply-immediately` to make the change immediately. Use `--no-apply-immediately` to make the change during your cluster's next maintenance window.

Example

The following code changes the backup retention period for the cluster sample-cluster.

For Linux, macOS, or Unix:

```
aws docdb modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --apply-immediately \
    --backup-retention-period 7
```

For Windows:

```
aws docdb modify-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --apply-immediately ^
    --backup-retention-period 7
```

Output from this operation looks something like the following.

```
{
    "DBCluster": {
        "BackupRetentionPeriod": 7,
        "DbClusterResourceId": "cluster-VDP53QEWEST7YHM36TTXOPJT5YE",
        "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,
        "ClusterCreateTime": "2018-06-17T00:06:19.374Z",
        "DeletionProtection": false,
        "DeletionProtectionEnabledAt": null,
        "DeletionProtectionStatus": "disabled"
    }
}
```

```
        "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
    }
}
```

It takes a few minutes for your changes to be applied. You can use the cluster only when its status is *available*. You can monitor the cluster's status using the console or AWS CLI. For more information, see [Monitoring an Amazon DocumentDB Cluster's Status \(p. 402\)](#).

Determining Pending Maintenance

You can determine whether you have the latest Amazon DocumentDB engine version by determining whether you have pending cluster maintenance.

Using the AWS Management Console

You can use the AWS Management Console to determine whether a cluster has pending maintenance.

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Clusters**.

Tip

If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.

3. Locate the **Maintenance** column to determine whether a cluster has pending maintenance.

The screenshot shows three panels from the AWS Management Console:

- Clusters (3)**: A list of clusters with their identifiers: docdb-2019-01-09-23-55-38, sample-cluster, and sample-cluster2.
- Engine version**: A table showing the engine version for each cluster: docdb 3.6.0, docdb 3.6.0, and docdb 3.6.0.
- Maintenance**: A table showing the maintenance status for each cluster: Available, None, and None. The "None" row for sample-cluster2 is highlighted with a red box.

None indicates that the cluster is running the latest engine version. **Available** indicates that the cluster has pending maintenance, which might mean that an engine upgrade is needed.

4. If your cluster has pending maintenance, continue with the steps at [Updating a Cluster's Engine Version \(p. 291\)](#).

Using the AWS CLI

You can use the AWS CLI to determine whether a cluster has the latest engine version by using the `describe-pending-maintenance-actions` operation with the following parameters.

Parameters

- **--resource-identifier**—Optional. The ARN for the resource (cluster). If this parameter is omitted, pending maintenance actions for all clusters are listed.

- **--region**—Optional. The AWS Region that you want to run this operation in, for example, us-east-1.

Example

For Linux, macOS, or Unix:

```
aws docdb describe-pending-maintenance-actions \
--resource-identifier arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster \
--region us-east-1
```

For Windows:

```
aws docdb describe-pending-maintenance-actions ^
--resource-identifier arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster ^
--region us-east-1
```

Output from this operation looks something like the following.

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

If your cluster has pending maintenance, continue with the steps at [Updating a Cluster's Engine Version \(p. 291\)](#).

Updating a Cluster's Engine Version

In this section, we will explain how to deploy a patch update using the AWS Management Console or the AWS CLI. A patch update is an update within the same engine version (for example, updating a 3.6 engine version to a newer 3.6 engine version). You can update it immediately or during your cluster's next maintenance window. To determine whether your engine needs an update, see [Determining Pending Maintenance \(p. 290\)](#). Please note that when you apply the update, your cluster will experience some downtime.

Note

If you are trying to upgrade from a major engine version to another, such as 3.6 to 4.0, please see [Upgrading your Amazon DocumentDB cluster from 3.6 to 4.0 using AWS Database Migration Service \(p. 121\)](#).

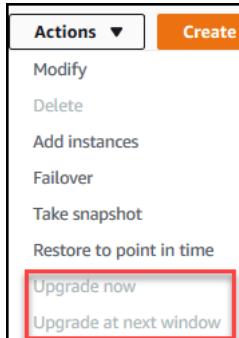
There are two configuration requirements to get the latest patch updates for a cluster's engine version:

- The cluster's status must be *available*.
- The cluster must be running an earlier engine version.

Using the AWS Management Console

The following procedure applies patch updates to your cluster's engine version using the console. You have the option to update immediately or during your cluster's next maintenance window.

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
 2. In the navigation pane, choose **Clusters**. In the list of clusters, choose the button to the left of the cluster that you want to upgrade. The status of the cluster must be *available*.
- Tip**
If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.
3. From the **Actions** menu, choose one of the following options. These menu options are selectable only if the cluster you chose is not running the latest engine version.



- **Upgrade now**—Immediately initiates the upgrade process. Your cluster will be offline for a time while the cluster is upgraded to the latest engine version.
 - **Upgrade at next window**—Initiates the upgrade process during the cluster's next maintenance window. Your cluster will be offline for a time while it is upgraded to the latest engine version.
4. When the confirmation window opens, choose one of the following:
 - **Upgrade**—To upgrade your cluster to the latest engine version according to the schedule chosen in the previous step.
 - **Cancel**—To cancel the cluster's engine upgrade and continue with the cluster's current engine version.

Using the AWS CLI

You can apply patch updates to your cluster using the AWS CLI and the `apply-pending-maintenance-action` operation with the following parameters.

Parameters

- **--resource-identifier**—Required. The ARN of the Amazon DocumentDB cluster that you are going to upgrade.
- **--apply-action**—Required. The following values are permitted. To upgrade your cluster engine version, use `db-upgrade`.
 - **db-upgrade**

- **system-update**
- **--opt-in-type**—Required. The following values are permitted.
 - immediate—Apply the maintenance action immediately.
 - next-maintenance—Apply the maintenance action during the next maintenance window.
 - undo-opt-in—Cancel any existing next-maintenance opt-in requests.

Example

The following example patch updates the engine version of sample-cluster to version 4.0.0.

For Linux, macOS, or 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
```

For 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 from this operation looks like the following.

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

Stopping and Starting an Amazon DocumentDB Cluster

Stopping and starting Amazon DocumentDB clusters can help you manage costs for development and test environments. Instead of creating and deleting clusters and instances each time you use Amazon DocumentDB, you can temporarily stop all the instances in your cluster when they aren't needed. You can then start them again when you resume your testing.

Topics

- [Overview of Stopping and Starting a Cluster \(p. 294\)](#)
- [Using the AWS Management Console \(p. 294\)](#)
- [Using the AWS CLI \(p. 295\)](#)

- Operations You Can Perform on a Stopped Cluster (p. 296)

Overview of Stopping and Starting a Cluster

During periods where you don't need an Amazon DocumentDB cluster, you can stop all instances in that cluster at once. You can then start the cluster again anytime you need to use it. Starting and stopping simplifies the setup and teardown processes for clusters that are used for development, testing, or similar activities that don't require continuous availability. You can stop and start a cluster using the AWS Management Console or the AWS CLI with a single action, regardless of how many instances are in the cluster.

While your cluster is stopped, the cluster storage volume remains unchanged. You are charged only for storage, manual snapshots, and automated backup storage within your specified retention window. You aren't charged for any instance hours. Amazon DocumentDB automatically starts your cluster after seven days so that it doesn't fall behind any required maintenance updates. When your cluster starts after seven days, you will begin to be charged for the instances in the cluster again. While your cluster is stopped, you can't query your storage volume because querying requires that instances are in the available state.

When an Amazon DocumentDB cluster is stopped, neither the cluster nor its instances can be modified in any way. This includes adding or removing instances, or deleting the cluster.

Using the AWS Management Console

The following procedure shows you how to stop a cluster with one or more instances in the available state, or start a stopped cluster.

To stop or start an Amazon DocumentDB cluster

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Clusters**.

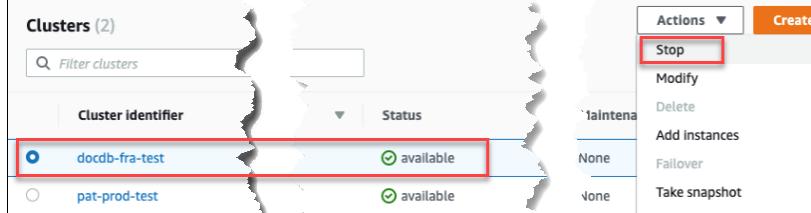
Tip

If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.

3. In the list of clusters, choose the button to the left of the name of the cluster that you want to stop or start.
4. Choose **Actions**, and then choose the action that you want to perform on the cluster.

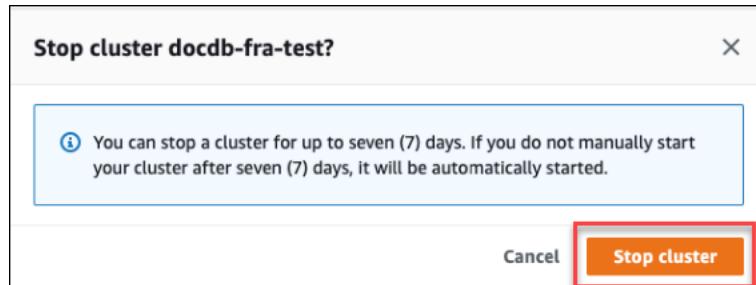
- If you want to stop the cluster and the cluster is available:

- a. Choose **Stop**.



To avoid activating the failover mechanism, the stop operation stops the replica instances first, and then the primary instance.

- b. On the confirmation dialog, confirm that you want to stop the cluster by choosing **Stop cluster**, or to keep the cluster running, choose **Cancel**.



- If you want to start the cluster, and the cluster is stopped, choose **Start**.

Cluster identifier	Status
greg-demo	available
greg-demo-second	stopped

5. Monitor the status of the cluster and its instances. If you started the cluster, you can resume using the cluster when the cluster and its instances are *available*. For more information, see [Determining a Cluster's Status \(p. 272\)](#).

Cluster identifier	Status
greg-demo	available
greg-demo-second	starting

Using the AWS CLI

The following code examples show you how to stop a cluster with one or more instances in the available state, or start a stopped cluster.

To stop a cluster with one or more available instances using the AWS CLI, use the `stop-db-cluster` operation. To start a stopped cluster, use the `start-db-cluster` operation. Both operations use the `--db-cluster-identifier` parameter.

Parameter:

- `--db-cluster-identifier`—Required. The name of the cluster to stop or start.

Example — To stop a cluster using the AWS CLI

The following code stops the cluster `sample-cluster`. The cluster must have one or more instances in the available state.

For Linux, macOS, or Unix:

```
aws docdb stop-db-cluster \
--db-cluster-identifier sample-cluster
```

For Windows:

```
aws docdb stop-db-cluster ^
--db-cluster-identifier sample-cluster
```

Example — To start a cluster using the AWS CLI

The following code starts the cluster sample-cluster. The cluster must currently be stopped.

For Linux, macOS, or Unix:

```
aws docdb start-db-cluster \
--db-cluster-identifier sample-cluster
```

For Windows:

```
aws docdb start-db-cluster ^
--db-cluster-identifier sample-cluster
```

Operations You Can Perform on a Stopped Cluster

While an Amazon DocumentDB cluster is stopped, you can do a point-in-time restore to any point within your specified automated backup retention window. For details about doing a point-in-time restore, see [Restoring to a Point in Time \(p. 241\)](#).

You can't modify the configuration of an Amazon DocumentDB cluster, or any of its instances, while the cluster is stopped. You also can't add or remove instances from the cluster, or delete the cluster if it still has any associated instances. You must start the cluster before performing any such administrative actions.

Amazon DocumentDB applies any scheduled maintenance to your stopped cluster only after it's started again. After seven days, Amazon DocumentDB automatically starts a stopped cluster so that it doesn't fall too far behind in its maintenance status. When the cluster restarts, you will begin to be charged for the instances in the cluster again.

While a cluster is stopped, Amazon DocumentDB does not perform any automated backups nor does it extend the backup retention period.

Deleting an Amazon DocumentDB Cluster

You can delete an Amazon DocumentDB cluster using the AWS Management Console or the AWS CLI. To delete a cluster, the cluster must be in the *available* state and must not have any instances associated with it. If the cluster is stopped, first start the cluster, wait for the cluster to become *available*, and then delete the cluster. For more information, see [Stopping and Starting an Amazon DocumentDB Cluster \(p. 293\)](#).

Deletion Protection

To protect your cluster from accidental deletion, you can enable *deletion protection*. Deletion protection is enabled by default when you create a cluster using the console. However, deletion protection is disabled by default if you create a cluster using the AWS CLI.

Amazon DocumentDB enforces deletion protection for a cluster whether you perform the delete operation using the console or the AWS CLI. If deletion protection is enabled, you can't delete a cluster. To delete a cluster that has deletion protection enabled, you must first modify the cluster and disable deletion protection.

When using the console with deletion protection enabled on a cluster, you can't delete the cluster's last instance because doing so also deletes the cluster. You can delete the last instance of a deletion protected cluster using the AWS CLI. However, the cluster itself still exists, and your data is preserved. You can access the data by creating new instances for the cluster. For more information about enabling and disabling deletion protection, see:

- [Creating an Amazon DocumentDB Cluster \(p. 274\)](#)
- [Modifying an Amazon DocumentDB Cluster \(p. 287\)](#)

Deleting an Amazon DocumentDB Cluster Using the AWS Management Console

To delete a cluster using the AWS Management Console, deletion protection must be disabled.

To determine whether a cluster has deletion protection enabled:

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
 2. In the navigation pane, choose **Clusters**.
- Tip**
If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.
3. Note that in the Clusters navigation box, the column **Cluster Identifier** shows both clusters and instances. Instances are listed underneath clusters, similar to the screenshot below.

Cluster identifier	Role	Engine version	Region & AZ
docdb-cloud9-getstarted	Cluster	3.6.0	us-east-1
docdb-cloud9-getstarted	Primary	3.6.0	us-east-1f
robo3t	Cluster	3.6.0	us-east-1
robo3t	Primary	3.6.0	us-east-1d

4. Choose the cluster's name, and select the **Configuration** tab. In the **Cluster details** section, locate **Deletion protection**. If deletion protection is enabled, modify the cluster to disable deletion protection. For information about modifying a cluster, see [Modifying an Amazon DocumentDB Cluster \(p. 287\)](#).

After **Deletion protection** is disabled, you are ready to delete the cluster.

To delete a cluster:

1. In the navigation pane, choose **Clusters**.
2. Determine whether the cluster has any instances by checking the **Instances** column. Before you can delete a cluster, you must delete all of its instances. For more information, see [Deleting an Amazon DocumentDB Instance \(p. 331\)](#).
3. Depending on whether your cluster has any instances, do one of the following steps.
 - If the cluster has no instances, select the button to the left of the cluster name and choose **Actions**. From the dropdown menu, choose **Delete**. Complete the **Delete <cluster-name>** dialog box, and then choose **Delete**.

- If the cluster has one or more instances, do the following:
 - a. In the navigation pane, choose **Instances**.
 - b. Delete each of the cluster's instances. When you delete the last instance, the cluster is also deleted. For information about deleting instances, see [Deleting an Amazon DocumentDB Instance \(p. 331\)](#).

It takes several minutes for the cluster to be deleted. To monitor the status of the cluster, see [Monitoring an Amazon DocumentDB Cluster's Status \(p. 402\)](#).

Deleting an Amazon DocumentDB Cluster Using the AWS CLI

You cannot delete a cluster that has any instances associated with it. To determine which instances are associated with your cluster, run the `describe-db-clusters` command and delete all of the cluster's instances. Then, if needed, disable deletion protection on your cluster, and finally, delete the cluster.

1. First, delete all of the cluster's instances.

To determine which instances you need to delete, run the following command.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].[DBClusterIdentifier,DBClusterMembers[*].DBInstanceIdentifier]'
```

Output from this operation looks something like the following (JSON format).

```
[  
  [  
    "sample-cluster",  
    [  
      "sample-instance-1",  
      "sample-instance-2"  
    ]  
  ]  
]
```

If the cluster you want to delete has any instances, delete them as shown below.

```
aws docdb delete-db-instance \
    --db-instance-identifier sample-instance
```

2. Second, disable deletion protection.

Using the AWS CLI to delete all of a cluster's instances does not delete the cluster. You must also delete the cluster, but you can do this only if deletion protection is disabled.

To determine whether the cluster has deletion protection enabled, run the following command.

Tip

To see the deletion protection status of all your Amazon DocumentDB clusters, omit the `--db-cluster-identifier` parameter.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].[DBClusterIdentifier,DeletionProtection]'
```

Output from this operation looks something like the following.

```
[  
  [  
    "sample-cluster",  
    "true"  
  ]  
]
```

If the cluster has deletion protection enabled, modify the cluster and disable deletion protection. To disable deletion protection on the cluster, run the following command.

```
aws docdb modify-db-cluster \  
  --db-cluster-identifier sample-cluster \  
  --no-deletion-protection \  
  --apply-immediately
```

3. Finally, delete the cluster.

After deletion protection is disabled, you are ready to delete the cluster. To delete a cluster, use the `delete-db-cluster` operation with the following parameters.

- **--db-cluster-identifier**—Required. The identifier of the cluster that you want to delete.
- **--final-db-snapshot-identifier**—Optional. If you want a final snapshot, you must include this parameter with a name for the final snapshot. You must include either `--final-db-snapshot-identifier` or `--skip-final-snapshot`.

Naming constraints:

- Length is [1–63] letters, numbers, or hyphens.
- First character must be a letter.
- Cannot end with a hyphen or contain two consecutive hyphens.
- Must be unique for all clusters across Amazon RDS, Amazon Neptune, and Amazon DocumentDB per AWS account, per Region.
- **--skip-final-snapshot**—Optional. Use this parameter only if you don't want to take a final snapshot before deleting your cluster. The default setting is to take a final snapshot. You must include either `--final-db-snapshot-identifier` or `--skip-final-snapshot`.

The following AWS CLI code deletes the cluster `sample-cluster` with a final snapshot. The operation fails if there are any instances associated with the cluster or if deletion protection is enabled.

Example

For Linux, macOS, or Unix:

```
aws docdb delete-db-cluster \  
  --db-cluster-identifier sample-cluster \  
  --final-db-snapshot-identifier sample-cluster-final-snapshot
```

For Windows:

```
aws docdb delete-db-cluster ^  
  --db-cluster-identifier sample-cluster ^
```

```
--final-db-snapshot-identifier sample-cluster-final-snapshot
```

Example

The following AWS CLI code deletes the cluster `sample-cluster` without taking a final snapshot.

For Linux, macOS, or Unix:

```
aws docdb delete-db-cluster \
--db-cluster-identifier sample-cluster \
--skip-final-snapshot
```

For Windows:

```
aws docdb delete-db-cluster ^
--db-cluster-identifier sample-cluster ^
--skip-final-snapshot
```

The output of the `delete-db-cluster` operation is the cluster you are deleting.

It takes several minutes for the cluster to be deleted. To monitor the status of the cluster, see [Monitoring a Cluster's Status \(p. 402\)](#).

Scaling Amazon DocumentDB Clusters

Amazon DocumentDB enables you to scale the storage and compute in your clusters based on your needs. This section describes how you can use storage scaling, instance scaling, and read scaling to manage performance and scaling for your Amazon DocumentDB clusters and instances.

Topics

- [Storage Scaling \(p. 300\)](#)
- [Instance Scaling \(p. 300\)](#)
- [Read Scaling \(p. 300\)](#)
- [Write Scaling \(p. 301\)](#)

Storage Scaling

Amazon DocumentDB storage automatically scales with the data in your cluster volume. As your data grows, your cluster volume storage grows in 10 GiB increments, up to 64 TiB.

Instance Scaling

You can scale your Amazon DocumentDB cluster as needed by modifying the instance class for each instance in the cluster. Amazon DocumentDB supports several instance classes that are optimized for Amazon DocumentDB.

For more information, see [Modifying an Amazon DocumentDB Instance \(p. 326\)](#).

Read Scaling

You can achieve read scaling for your Amazon DocumentDB cluster by creating up to 15 Amazon DocumentDB replicas in the cluster. Each Amazon DocumentDB replica returns the same data from the cluster volume with minimal replica lag—usually less than 100 milliseconds after the primary instance

has written an update. As your read traffic increases, you can create additional Amazon DocumentDB replicas and connect to them directly to distribute the read load for your cluster. Amazon DocumentDB replicas don't have to be of the same instance class as the primary instance.

For more information, see [Adding an Amazon DocumentDB Instance to a Cluster \(p. 319\)](#).

To read scale with Amazon DocumentDB, we recommend that you connect to your cluster as a replica set and distribute reads to replica instances using the built-in read preference capabilities of your driver. For more information, please see [Connecting to Amazon DocumentDB as a Replica Set \(p. 496\)](#)

Write Scaling

You can scale write capacity on your Amazon DocumentDB cluster by increasing the size of your cluster's primary instance. This section provides two methods for scaling your cluster's primary instance based on your needs. The first option seeks to minimize application impact but requires more steps to complete. The second option optimizes for simplicity as it has fewer steps, but it comes with the tradeoff of having more potential impact to your application.

Depending on your application, you can choose what approach below is best for you. For more information about available instance sizes and costs, see the [Amazon DocumentDB Pricing](#) page.

1. **Optimize for high availability and performance** — If you are connecting to your cluster in [replica set mode \(p. 496\)](#) (recommended), you can use the following process to minimize the impact to your application when scaling your primary instance. This method minimizes impact because it keeps your cluster at or above your high availability, and read scaling targets are added to the cluster as instances, instead of being updated in place.
 - a. Add one or more replicas of the larger instance type to your cluster (see [?? \(p. 319\)](#)). We recommend all replicas be of the same or larger instance type as the primary. This avoids an unintentional reduction in write performance from failing over to a smaller instance type. For most customers, this means temporarily doubling the number of instances in their cluster, then removing the smaller replicas after scaling is complete.
 - b. Set the failover tier on all new replicas to priority zero, ensuring a replica of the smaller instance type has the highest failover priority. For more information, see [?? \(p. 345\)](#).
 - c. Initiate a manual failover, which will promote one of the new replicas to be the primary instance. For more information, see [?? \(p. 346\)](#).

Note

This will incur ~30 seconds of downtime for your cluster. Please plan accordingly.

- d. Remove all replicas of an instance type smaller than your new primary from the cluster.
- e. Set the failover tier of all instances back to the same priority (usually, this means setting them back to 1).

As an example, suppose that you have a cluster that currently contains three `r5.large` instances (one primary and two replicas), and you want to scale to an `r5.xlarge` instance type. To do so, you would first add three `r5.xlarge` replica instances to your cluster and then set the failover tier of the new `r5.xlarge` replicas to zero. Next, you would initiate a manual failover (understanding that your application will experience ~30 seconds of downtime). Once the failover is complete, you would remove all three `r5.large` instances from your cluster, leaving the cluster scaled to `r5.xlarge` instances.

To help optimize costs, Amazon DocumentDB instances are billed in one second increments, with a ten minute minimum charge following a billable status change such as creating, modifying, or deleting an instance. For more information, see [Cost Optimization \(p. 87\)](#) in the best practices documentation.

2. **Optimize for simplicity** — This approach optimizes for simplicity. It doesn't expand and contract the cluster, but it might temporarily reduce your read capacity.

It is possible that changing the instance class of a replica will result in that instance not serving requests for a brief period of time, from a few seconds to less than 30 seconds. If you are connecting to your cluster in [replica set mode \(p. 496\)](#) (recommended), then this would reduce your read capacity by one replica (e.g., to 66% capacity in a 3-node cluster, or 75% capacity in a 4-node cluster, etc.) during the scaling operation.

- a. Scale one of the replica instances in your cluster. For more information, see [Managing Instance Classes \(p. 313\)](#).
- b. Wait until the instance is available (see [Monitoring an Amazon DocumentDB Instance's Status \(p. 404\)](#)).

Note

This will incur ~30 seconds of downtime for your cluster. Please plan accordingly.

- c. Continue executing steps 1 and 2 until all replicas instances have been scaled, one by one.
- d. Initiate a manual failover. This will promote one of the replicas to be the primary instance. For more information, see [Amazon DocumentDB Failover \(p. 345\)](#).

Note

This will incur up to 30 seconds of downtime for your cluster, but often takes less time than that. Please plan accordingly.

- e. Scale the former primary (now a replica) instance.

Cloning a volume for an Amazon DocumentDB cluster

By using Amazon DocumentDB cloning, you can create a new cluster that uses the same Amazon DocumentDB cluster volume and has the same data as the original. The process is designed to be fast and cost-effective. The new cluster with its associated data volume is known as a *clone*. Creating a clone is faster and more space-efficient than physically copying the data using other techniques, such as restoring a snapshot.

Amazon DocumentDB supports creating an Amazon DocumentDB provisioned clone from a provisioned Amazon DocumentDB cluster. When you create a clone using a different deployment configuration than the source, the clone is created using the latest version of the source's Amazon DocumentDB engine.

When you create clones from your Amazon DocumentDB clusters, the clones are created in your AWS account—the same account that owns the source Amazon DocumentDB cluster.

Topics

- [Overview of Amazon DocumentDB cloning \(p. 302\)](#)
- [Limitations of Amazon DocumentDB cloning \(p. 303\)](#)
- [How Amazon DocumentDB cloning works \(p. 303\)](#)
- [Creating an Amazon DocumentDB clone \(p. 306\)](#)

Overview of Amazon DocumentDB cloning

Amazon DocumentDB uses a *copy-on-write protocol* to create a clone. This mechanism uses minimal additional space to create an initial clone. When the clone is first created, Amazon DocumentDB keeps a single copy of the data that is used by the source DB cluster and the new (cloned) Amazon DocumentDB cluster. Additional storage is allocated only when changes are made to data (on the Amazon DocumentDB storage volume) by the source Amazon DocumentDB cluster or the Amazon DocumentDB cluster clone. To learn more about the copy-on-write protocol, see [How Amazon DocumentDB cloning works \(p. 303\)](#).

Amazon DocumentDB cloning is especially useful for quickly setting up test environments using your production data, without risking data corruption. You can use clones for many types of applications, such as the following:

- Experiment with potential changes (schema changes and parameter group changes, for example) to assess all impacts.
- Run workload-intensive operations, such as exporting data or running analytical queries on the clone.
- Create a copy of your production DB cluster for development, testing, or other purposes.

You can create more than one clone from the same Amazon DocumentDB cluster. You can also create multiple clones from another clone.

After creating an Amazon DocumentDB clone, you can configure the Amazon DocumentDB instances differently from the source Amazon DocumentDB cluster. For example, you might not need a clone for development purposes to meet the same high availability requirements as the source production Amazon DocumentDB cluster. In this case, you can configure the clone with a single Amazon DocumentDB instance rather than the multiple DB instances used by the Amazon DocumentDB cluster.

When you finish using the clone for your testing, development, or other purposes, you can delete it.

Limitations of Amazon DocumentDB cloning

Amazon DocumentDB cloning currently has the following limitations:

- You can create as many clones as you want, up to the maximum number of DB clusters allowed in the AWS Region. However, after you create 15 clones, the next clone is a full copy. The cloning operation acts like a point-in-time recovery.
- You can't create a clone in a different AWS Region from the source Amazon DocumentDB cluster.
- You can't create a clone from an Amazon DocumentDB cluster that has no DB instances. You can only clone Amazon DocumentDB clusters that have at least one DB instance.
- You can create a clone in a different virtual private cloud (VPC) than that of the Amazon DocumentDB cluster. If you do, the subnets of the VPCs must map to the same Availability Zones.

How Amazon DocumentDB cloning works

Amazon DocumentDB cloning works at the storage layer of an Amazon DocumentDB cluster. It uses a *copy-on-write* protocol that's both fast and space-efficient in terms of the underlying durable media supporting the Amazon DocumentDB storage volume. You can learn more about Amazon DocumentDB cluster volumes in [Managing Amazon DocumentDB Clusters \(p. 269\)](#).

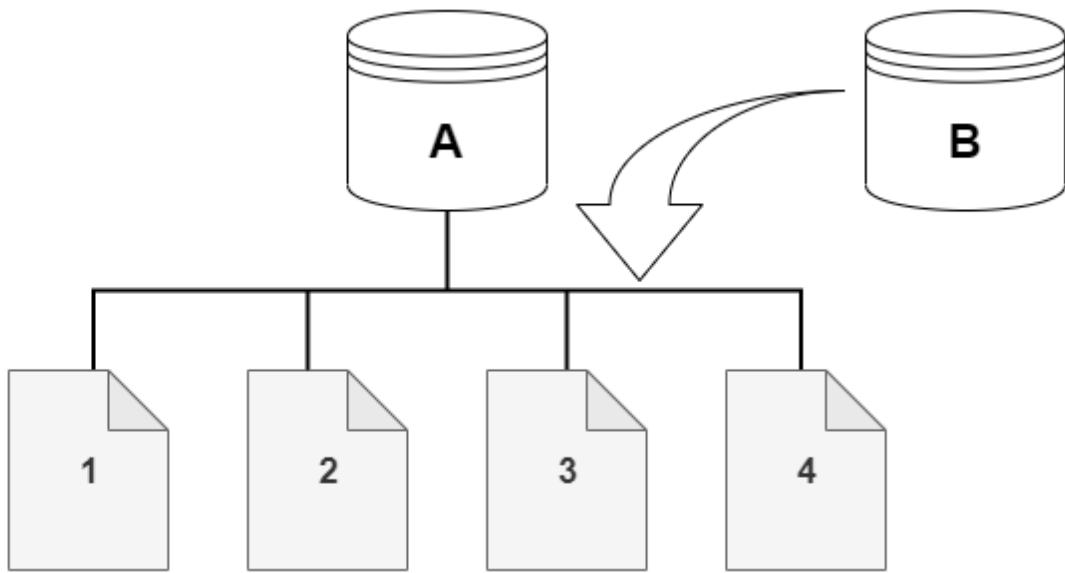
Topics

- [Understanding the copy-on-write protocol \(p. 303\)](#)
- [Deleting a source cluster volume \(p. 306\)](#)

Understanding the copy-on-write protocol

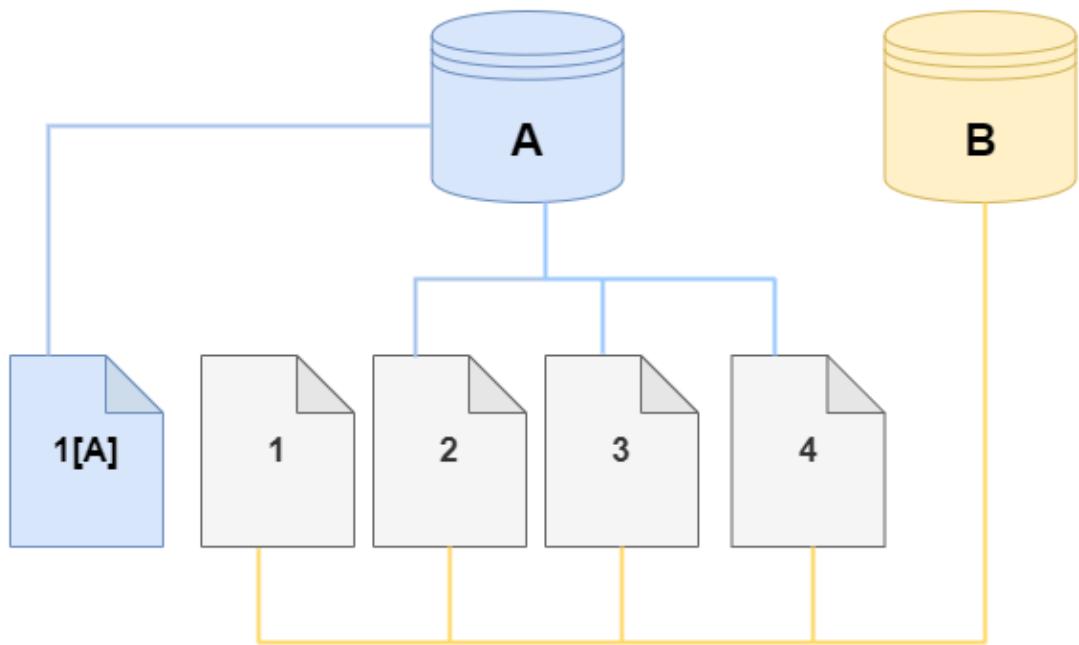
An Amazon DocumentDB cluster stores data in pages in the underlying Amazon DocumentDB storage volume.

For example, in the following diagram you can find an Amazon DocumentDB cluster (A) that has four data pages, 1, 2, 3, and 4. Imagine that a clone, B, is created from the Amazon DocumentDB cluster. When the clone is created, no data is copied. Rather, the clone points to the same set of pages as the source Amazon DocumentDB cluster.

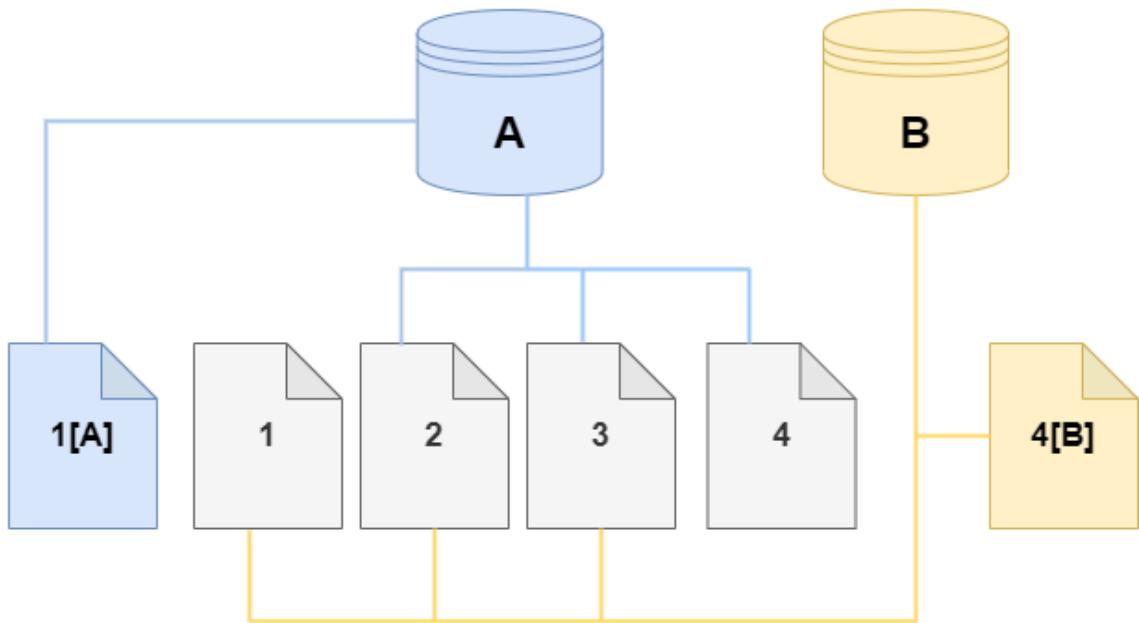


When the clone is created, no additional storage is usually needed. The copy-on-write protocol uses the same segment on the physical storage media as the source segment. Additional storage is required only if the capacity of the source segment isn't sufficient for the entire clone segment. If that's the case, the source segment is copied to another physical device.

In the following diagrams, you can find an example of the copy-on-write protocol in action using the same cluster A and its clone, B, as shown preceding. Let's say that you make a change to your Amazon DocumentDB cluster (A) that results in a change to data held on page 1. Instead of writing to the original page 1, Amazon DocumentDB creates a new page 1[A]. The Amazon DocumentDB cluster volume for cluster (A) now points to page 1[A], 2, 3, and 4, while the clone (B) still references the original pages.



On the clone, a change is made to page 4 on the storage volume. Instead of writing to the original page 4, Amazon DocumentDB creates a new page, 4[B]. The clone now points to pages 1, 2, 3, and to page 4[B], while the cluster (A) continues pointing to 1[A], 2, 3, and 4.



As more changes occur over time in both the source Amazon DocumentDB cluster volume and the clone, more storage is needed to capture and store the changes.

Deleting a source cluster volume

When you delete a source cluster volume that has one or more clones associated with it, the clones aren't affected. The clones continue to point to the pages that were previously owned by the source cluster volume.

Creating an Amazon DocumentDB clone

You can create a clone in the same AWS account as the source Amazon DocumentDB cluster. To do so, you can use the AWS Management Console or the AWS CLI and the procedures following.

By using Amazon DocumentDB cloning, you can create a provisioned Amazon DocumentDB cluster clone from a provisioned Amazon DocumentDB cluster.

Console

The following procedure describes how to clone an Amazon DocumentDB cluster using the AWS Management Console.

Creating a clone using the AWS Management Console results in an Amazon DocumentDB cluster with one Amazon DocumentDB instance.

These instructions apply for DB clusters owned by the same AWS account that is creating the clone. The DB cluster must be owned by the same AWS account as cross-account cloning is not supported in Amazon DocumentDB.

To create a clone of a DB cluster owned by your AWS account using the AWS Management Console

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Clusters**.
3. Choose your Amazon DocumentDB cluster from the list, and for **Actions**, choose **Create clone**.

The screenshot shows the AWS DocumentDB Clusters page. There are five clusters listed:

- docdb-2022-07-08-09-55-09**: Regional cluster, Primary instance, Engine version 4.0.0, us-east-1, Status available, CPU 7.44%, Current active connections 0.
- docdb-2022-07-08-09-55-09**: Replica instance, Engine version 4.0.0, us-east-1d, Status available, CPU 12.03%, Current active connections 0.
- docdb-2022-07-08-09-55-09**: Replica instance, Engine version 4.0.0, us-east-1a, Status available, CPU 13.88%, Current active connections 0.
- test-with-encryption**: Regional cluster, Primary instance, Engine version 4.0.0, us-east-1, Status available, CPU 6.81%, Current active connections 0.
- test-with-encryption-new-key**: Regional cluster, Primary instance, Engine version 4.0.0, us-east-1d, Status available, CPU 14.83%, Current active connections 0.
- volume-clone-test**: Regional cluster, Primary instance, Engine version 4.0.0, us-east-1, Status available, CPU 2.95%, Current active connections 0.
- volume-clone-test-Instance1**: Primary instance, Engine version 4.0.0, us-east-1b, Status available, CPU 0, Current active connections 0.

A context menu is open over the first cluster, with the 'Create clone' option highlighted. Other options in the menu include Stop, Modify, Delete, Reboot, Add instances, Failover, Take snapshot, Restore to point in time, Add Region, Remove from global, Upgrade now, Upgrade at next window, Enable deletion protection, and Create clone.

The Create clone page opens, where you can configure a **Cluster identifier** and an **Instance class**, and other options for the Amazon DocumentDB cluster clone.

4. In the **Settings** section, do the following:
 - a. For **Cluster identifier**, enter the name that you want to give to your cloned Amazon DocumentDB cluster.
 - b. For **Instance configuration**, select an appropriate **Instance class** for your cloned Amazon DocumentDB cluster.

Create Clone

You are cloning a DocumentDB cluster. This will create a new DB cluster that includes all of the data from the existing database as well as a writer DB instance.

Settings

Source cluster identifier
docdb-2022-07-08-09-55-09

Cluster identifier
Specify a unique cluster identifier.

Instance configuration

Instance class

2 vCPUs 16GiB RAM

- c. For **Network settings**, choose a **Subnet group** for your use case and the associated VPC security groups.
- d. For **Encryption-at-rest**, if the source cluster (the cluster that is being cloned) has encryption enabled, the cloned cluster must also have encryption enabled. If this scenario is true, then the **Enable encryption** options are grayed out (disabled) but with the **Enable encryption** choice selected. Conversely, if the source cluster does not have encryption enabled, the **Enable encryption** options are available and you can choose to enable or disable encryption.

The screenshot shows two sections of the cloning configuration interface:

- Network settings**:
 - Subnet group**: A dropdown menu set to "default".
 - VPC security groups**: A dropdown menu set to "Select VPC security groups", with "default" listed below it.
- Encryption-at-rest**:
 - Enable encryption**: A radio button group where "Enable encryption" is selected.
 - KMS key ID**: A dropdown menu set to "(default) aws/rds".
 - Account**: Displays the account ID "12345678910".
 - KMS key ID**: Displays the KMS key ID "example-key-abcdef123".

- e. Complete the new cluster clone configuration by selecting the type of logs to export (optional), entering a specific port used to connect to the cluster, and enabling protection from accidentally deleting the cluster (enabled by default).

The screenshot shows the 'Create New Stack' wizard in the AWS CloudFormation console. The current step is 'Set Cluster Options'. The configuration includes:

- Log exports:** Options for publishing logs to Amazon CloudWatch Logs, including Audit logs and Profiler logs.
- Cluster options:** Port is set to 27017.
- Deletion protection:** The 'Enable deletion protection' checkbox is checked, with a note explaining it protects the cluster from accidental deletion.
- Tags:** No tags are associated with the cluster.

At the bottom right are 'Cancel' and 'Create' buttons.

- f. Finish entering all settings for your Amazon DocumentDB cluster clone. To learn more about Amazon DocumentDB cluster and instance settings, see [Managing Amazon DocumentDB Clusters \(p. 269\)](#).
5. Choose **Create clone** to launch the Amazon DocumentDB clone of your chosen Amazon DocumentDB cluster.

When the clone is created, it's listed with your other Amazon DocumentDB clusters in the console **Databases** section and displays its current state. Your clone is ready to use when its state is **Available**.

AWS CLI

Using the AWS CLI for cloning your Amazon DocumentDB cluster involves a couple of steps.

The `restore-db-cluster-to-point-in-time` AWS CLI command that you use results in an empty Amazon DocumentDB cluster with 0 Amazon DocumentDB instances. That is, the command restores only the Amazon DocumentDB cluster, not the DB instances for that cluster. You do that separately after the clone is available. The two steps in the process are as follows:

1. Create the clone by using the [restore-db-cluster-to-point-in-time](#) CLI command. The parameters that you use with this command control the capacity type and other details of the empty Amazon DocumentDB cluster (clone) being created.
2. Create the Amazon DocumentDB instance for the clone by using the [create-db-instance](#) CLI command to recreate the Amazon DocumentDB instance in the restored Amazon DocumentDB cluster.

The commands following assume that the AWS CLI is set up with your AWS Region as the default. This approach saves you from passing the `--region` name in each of the commands. For more information, see [Configuring the AWS CLI](#). You can also specify the `--region` in each of the CLI commands that follow.

Topics

- [Creating the clone \(p. 310\)](#)
- [Checking the status and getting clone details \(p. 311\)](#)
- [Creating the Amazon DocumentDB instance for your clone \(p. 311\)](#)
- [Parameters to use for cloning \(p. 312\)](#)

Creating the clone

The specific parameters that you pass to the [restore-db-cluster-to-point-in-time](#) CLI command vary. What you pass depends on the type of clone that you want to create.

Use the following procedure to create a provisioned Amazon DocumentDB clone from a provisioned Amazon DocumentDB cluster.

To create a clone of the same engine mode as the source Amazon DocumentDB cluster

- Use the [restore-db-cluster-to-point-in-time](#) CLI command and specify values for the following parameters:
 - `--db-cluster-identifier` – Choose a meaningful name for your clone. You name the clone when you use the [restore-db-cluster-to-point-in-time](#) CLI command.
 - `--restore-type` – Use `copy-on-write` to create a clone of the source DB cluster. Without this parameter, the [restore-db-cluster-to-point-in-time](#) restores the Amazon DocumentDB cluster rather than creating a clone. Default for `restore-type` is `full-copy`.
 - `--source-db-cluster-identifier` – Use the name of the source Amazon DocumentDB cluster that you want to clone.
 - `--use-latest-restorable-time` – This value points to the latest restorable volume data for the clone. This parameter is required for `restore-type copy-on-write`, however, you can not use the `restore-to-time` parameter with it.

The following example creates a clone named `my-clone` from a cluster named `my-source-cluster`.

For Linux, macOS, or 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
```

For 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
```

The command returns the JSON object containing details of the clone. Check to make sure that your cloned DB cluster is available before trying to create the DB instance for your clone. For more information, see [Checking the status and getting clone details \(p. 311\)](#).

Checking the status and getting clone details

You can use the following command to check the status of your newly created empty DB cluster.

```
$ aws docdb describe-db-clusters --db-cluster-identifier my-clone --query '*[].[Status]' --
output text
```

Or you can obtain the status and the other values that you need to [create the DB instance for your clone \(p. 311\)](#) by using the following AWS CLI query.

For Linux, macOS, or Unix:

```
aws docdb describe-db-clusters --db-cluster-identifier my-clone \
--query '*[].{Status:Status,Engine:Engine,EngineVersion:EngineVersion}'
```

For Windows:

```
aws docdb describe-db-clusters --db-cluster-identifier my-clone ^
--query "*[].{Status:Status,Engine:Engine,EngineVersion:EngineVersion}"
```

This query returns output similar to the following.

```
[  
 {  
     "Status": "available",  
     "Engine": "docdb",  
     "EngineVersion": "4.0.0",  
 }  
]
```

Creating the Amazon DocumentDB instance for your clone

Use the [create-db-instance](#) CLI command to create the DB instance for your clone.

The `--db-instance-class` parameter is used for provisioned Amazon DocumentDB clusters only.

For Linux, macOS, or 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
```

For 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
```

Parameters to use for cloning

The following table summarizes the various parameters used with `restore-db-cluster-to-point-in-time` to clone Amazon DocumentDB clusters.

Parameter	Description
<code>--source-db-cluster-identifier</code>	Use the name of the source Amazon DocumentDB cluster that you want to clone.
<code>--db-cluster-identifier</code>	Choose a meaningful name for your clone. You name your clone with the <code>restore-db-cluster-to-point-in-time</code> command. Then you pass this name to the <code>create-db-instance</code> command.
<code>--restore-type</code>	Specify <code>copy-on-write</code> as the <code>--restore-type</code> to create a clone of the source DB cluster rather than restoring the source Amazon DocumentDB cluster.
<code>--use-latest-restorable-time</code>	This value points to the latest restorable volume data for the clone.

Understanding Amazon DocumentDB Cluster Fault Tolerance

Amazon DocumentDB clusters are fault tolerant by design. Each cluster's volume spans multiple Availability Zones in a single AWS Region, and each Availability Zone contains a copy of the cluster's volume data. This functionality means that your cluster can tolerate an Availability Zone failure without any loss of data and only a brief interruption of service.

If the primary instance in a cluster fails, Amazon DocumentDB automatically performs a failover to a new primary instance in one of two ways:

- By promoting an existing Amazon DocumentDB replica to the new primary instance chosen based on the Promotion Tier setting of each replica, and then creating a replacement for the former primary. A failover to the replica instance typically takes less than 30 seconds. Read and write operations may experience brief interruption during this period. To increase the availability of your cluster, we recommend that you create at least one or more Amazon DocumentDB replicas in two or more different Availability Zones.
- By creating a new primary instance. This only happens if you do not have a replica instance in your cluster and can take a few minutes to complete.

If the cluster has one or more Amazon DocumentDB replicas, an Amazon DocumentDB replica is promoted to the primary instance during a failure event. A failure event results in a brief interruption, during which read and write operations fail with an exception. However, service is typically restored in less than 120 seconds, and often less than 60 seconds. To increase the availability of your cluster, we recommend that you create at least one or more Amazon DocumentDB replicas in two or more different Availability Zones.

You can customize the order in which your Amazon DocumentDB replicas are promoted to the primary instance after a failure by assigning each replica a priority. Priorities range from 0 for the highest priority to 15 for the lowest priority. If the primary instance fails, the Amazon DocumentDB replica with the highest priority is promoted to the new primary instance. You can modify the priority of an Amazon

DocumentDB replica at any time. Modifying the priority doesn't trigger a failover. You can use the `modify-db-instance` operation with the `--promotion-tier` parameter. For more information about customizing the failover priority of an instance, see [Amazon DocumentDB Failover \(p. 345\)](#).

More than one Amazon DocumentDB replica can share the same priority, resulting in promotion tiers. If two or more Amazon DocumentDB replicas share the same priority, then the replica that is largest in size is promoted to primary. If two or more Amazon DocumentDB replicas share the same priority and size, an arbitrary replica in the same promotion tier is promoted.

If the cluster doesn't contain any Amazon DocumentDB replicas, the primary instance is re-created during a failure event. A failure event results in an interruption, during which read and write operations fail with an exception. Service is restored when the new primary instance is created, which typically takes less than 10 minutes. Promoting an Amazon DocumentDB replica to the primary instance is much faster than creating a new primary instance.

Managing Amazon DocumentDB Instances

The following topics provide information to help you manage your Amazon DocumentDB instances. They include details about instance classes and statuses, and how to create, delete, and modify an instance.

Topics

- [Managing Instance Classes \(p. 313\)](#)
- [Determining an Instance's Status \(p. 319\)](#)
- [Amazon DocumentDB Instance Lifecycle \(p. 319\)](#)

Managing Instance Classes

The instance class determines the computation and memory capacity of an Amazon DocumentDB (with MongoDB compatibility) instance. The instance class you need depends on your processing power and memory requirements.

Amazon DocumentDB supports the R4, R5, R6G, T3, and T4G families of instance classes. These classes are current-generation instance classes that are optimized for memory-intensive applications. For the specifications on these classes, see [Instance Class Specifications \(p. 317\)](#).

Topics

- [Determining an Instance Class \(p. 313\)](#)
- [Changing an Instance's Class \(p. 315\)](#)
- [Supported Instance Classes by Region \(p. 316\)](#)
- [Instance Class Specifications \(p. 317\)](#)

Determining an Instance Class

To determine the class of an instance, you can use the AWS Management Console or the `describe-db-instances` AWS CLI operation.

Using the AWS Management Console

To determine the instance class for your cluster's instances, complete the following steps in the console.

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Clusters** to find the instance that you're interested in.

Tip

If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.

3. In the Clusters navigation box, you'll see the column **Cluster Identifier**. Your instances are listed under clusters, similar to the screenshot below.

Cluster identifier	Role
docdb-cloud9-getstarted	Cluster
docdb-cloud9-getstarted	Primary
robo3t	Cluster
robo3t	Primary

4. In the list of instances, expand the cluster to find the instances you are interested in. Find the instance that you want. Then, look at the **Size** column of the instance's row to see its instance class.

In the following image, the instance class for instance `robo3t` is `db.r5.4xlarge`.

Cluster identifier	Role	Engine version	Region & AZ	Status	Size	Maintenance
docdb-cloud9-getstarted	Cluster	3.6.0	us-east-1	available	1 Instance	None
docdb-cloud9-getstarted	Primary	3.6.0	us-east-1f	available	db.r5.large	None
robo3t	Cluster	3.6.0	us-east-1	available	1 Instance	None
robo3t	Primary	3.6.0	us-east-1d	available	db.r5.large	None

Using the AWS CLI

To determine the class of an instance using the AWS CLI, use the `describe-db-instances` operation with the following parameters.

- **--db-instance-identifier** — Optional. Specifies the instance that you want to find the instance class for. If this parameter is omitted, `describe-db-instances` returns a description for up to 100 of your instances.
- **--query** — Optional. Specifies the members of the instance to include in the results. If this parameter is omitted, all instance members are returned.

Example

The following example finds the instance name and class for the instance `sample-instance-1`.

For Linux, macOS, or Unix:

```
aws docdb describe-db-instances \
--query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]' \
```

```
--db-instance-identifier sample-instance-1
```

For Windows:

```
aws docdb describe-db-instances ^
--query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]' ^
--db-instance-identifier sample-instance-1
```

Output from this operation looks something like the following.

```
[  
 [  
   "sample-instance-1",  
   "db.r5.large"  
 ]
```

Example

The following example finds the instance name and class for up to 100 Amazon DocumentDB instances.

For Linux, macOS, or Unix:

```
aws docdb describe-db-instances \
--query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]' \
--filter Name=engine,Values=docdb
```

For Windows:

```
aws docdb describe-db-instances ^
--query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]' ^
--filter Name=engine,Values=docdb
```

Output from this operation looks something like the following.

```
[  
 [  
   ["sample-instance-1",  
    "db.r5.large"  
   ],  
 [  
   ["sample-instance-2",  
    "db.r5.large"  
   ],  
 [  
   ["sample-instance-3",  
    "db.r5.4xlarge"  
   ],  
 [  
   ["sample-instance-4",  
    "db.r5.4xlarge"  
   ]  
 ]
```

For more information, see [Describing Amazon DocumentDB Instances \(p. 323\)](#).

Changing an Instance's Class

You can change the instance class of your instance using the AWS Management Console or the AWS CLI. For more information, see [Modifying an Amazon DocumentDB Instance \(p. 326\)](#).

Supported Instance Classes by Region

Amazon DocumentDB supports the following instance classes:

- R6G—Latest generation of memory-optimized instances powered by Arm-based AWS Graviton2 processors that provide up to 30% better performance over R5 instances at 5% cheaper cost. Graviton instance classes are supported on the 4.0.0 engine only.
- R5—Memory-optimized instances that provide up to 100% better performance over R4 instances for the same instance cost.
- R4—Previous generation of memory-optimized instances.
- T4G—Latest-generation low cost burstable general-purpose instance type powered by Arm-based AWS Graviton2 processors that provides a baseline level of CPU performance, delivering up to 35% better price performance over T3 instances and ideal for running applications with moderate CPU usage that experience temporary spikes in usage. Graviton instance classes are supported on the 4.0.0 engine only.
- T3—Low cost burstable general-purpose instance type that provides a baseline level of CPU performance with the ability to burst CPU usage at any time for as long as required.

For detailed specifications on the instance classes, see [Instance Class Specifications \(p. 317\)](#).

A particular instance class may or may not be supported in a given Region. The following table specifies which instance classes are supported by Amazon DocumentDB in each Region.

Supported instance classes by Region

Region	R6G	R5	R4	T4G	T3
US East (Ohio)	Supported	Supported	Supported	Supported	Supported
US East (N. Virginia)	Supported	Supported	Supported	Supported	Supported
US West (Oregon)	Supported	Supported	Supported	Supported	Supported
South America (São Paulo)	Supported	Supported		Supported	Supported
Asia Pacific (Mumbai)	Supported	Supported		Supported	Supported
Asia Pacific (Seoul)	Supported	Supported		Supported	Supported
Asia Pacific (Sydney)	Supported	Supported		Supported	Supported
Asia Pacific (Singapore)	Supported	Supported		Supported	Supported
Asia Pacific (Tokyo)	Supported	Supported		Supported	Supported
Canada (Central)	Supported	Supported		Supported	Supported
Europe (Frankfurt)	Supported	Supported		Supported	Supported

Region	R6G	R5	R4	T4G	T3
Europe (Ireland)	Supported	Supported	Supported	Supported	Supported
Europe (London)	Supported	Supported		Supported	Supported
Europe (Milan)	Supported	Supported		Supported	Supported
Europe (Paris)	Supported	Supported		Supported	Supported
China (Beijing) Region	Supported	Supported		Supported	Supported
China (Ningxia)	Supported	Supported		Supported	Supported
AWS GovCloud (US)	Supported	Supported			Supported

Instance Class Specifications

The following table provides details of the Amazon DocumentDB instance classes. You can find explanations for each table column below the table.

Supported Amazon DocumentDB instance classes

Instance class	vCPU ¹	Memory (GiB) ²	Max. temp. storage (GiB) ³	Max. bandwidth performance (Mbps) ⁴	Network performance ⁵	Supporting Engines ⁶
R6G – Current Generation Memory-Optimized Instance Class based on Graviton2						
db.r6g.large	2	16	32	Up to 4,750	Up to 10 Gbps	4.0.0 only
db.r6g.xlarge	4	32	63	Up to 4,750	Up to 10 Gbps	4.0.0 only
db.r6g.2xlarge	8	64	126	Up to 4,750	Up to 10 Gbps	4.0.0 only
db.r6g.4xlarge	16	128	252	4,750	Up to 10 Gbps	4.0.0 only
db.r6g.8xlarge	32	256	504	9,000	12 Gbps	4.0.0 only
db.r6g.12xlarge	48	384	756	13,500	20 Gbps	4.0.0 only
db.r6g.16xlarge	64	512	1008	19,000	25 Gbps	4.0.0 only
R5 – Previous Generation Memory-Optimized Instance Class						
db.r5.large	2	16	31	Up to 3,500	Up to 10 Gbps	3.6.0 and 4.0.0
db.r5.xlarge	4	32	62	Up to 3,500	Up to 10 Gbps	3.6.0 and 4.0.0
db.r5.2xlarge	8	64	124	Up to 3,500	Up to 10 Gbps	3.6.0 and 4.0.0

Instance class	vCPU ¹	Memory (GiB) ²	Max. temp. storage (GiB) ³	Max. bandwidth (Mbps) ⁴	Network performance ⁵	Supporting Engines ⁶
db.r5.4xlarge	16	128	249	3,500	Up to 10 Gbps	3.6.0 and 4.0.0
db.r5.8xlarge	32	256	504	6,800	10 Gbps	3.6.0 and 4.0.0
db.r5.12xlarge	48	384	748	7,000	10 Gbps	3.6.0 and 4.0.0
db.r5.16xlarge	64	512	1008	13,600	20 Gbps	3.6.0 and 4.0.0
db.r5.24xlarge	96	768	1500	14,000	25 Gbps	3.6.0 and 4.0.0
R4 – Previous Generation Memory-Optimized Instance Class						
db.r4.large	2	15.25	30	437	Up to 10 Gbps	3.6.0 only
db.r4.xlarge	4	30.5	60	875	Up to 10 Gbps	3.6.0 only
db.r4.2xlarge	8	61	120	875	Up to 10 Gbps	3.6.0 only
db.r4.4xlarge	16	122	240	875	Up to 10 Gbps	3.6.0 only
db.r4.8xlarge	32	244	480	875	10 Gbps	3.6.0 only
db.r4.16xlarge	64	488	960	14,000	25 Gbps	3.6.0 only
T4G – Latest Generation Burstable Performance Instance Classes based on Graviton2						
db.t4g.medium	2	4	8.13	Up to 2,085	Up to 5 Gbps	4.0.0 only
T3 – Previous Generation Burstable Performance Instance Classes						
db.t3.medium	2	4	7.5	Up to 1,536	Up to 5 Gbps	3.6.0 and 4.0.0
<ol style="list-style-type: none"> vCPU — The number of virtual central processing units (CPUs). A virtual CPU is a unit of capacity that you can use to compare instance classes. Instead of purchasing or leasing a particular processor to use for several months or years, you are renting capacity by the hour. Our goal is to provide a consistent amount of CPU capacity no matter what the actual underlying hardware. Memory (GiB) — The RAM, in gigabytes, that is allocated to the instance. There is often a consistent ratio between memory and vCPU. Max. temp. storage (GiB) — The RAM, in gigabytes, that is allocated to the instance for non-persistent temporary file storage. Max. bandwidth (Mbps) — The maximum bandwidth in megabits per second. Divide by 8 to get the expected throughput in megabytes per second. Network performance — The network speed relative to other instance classes. Supporting Engines — The Amazon DocumentDB engines that support the instance class. 						

Determining an Instance's Status

To see the valid instance statuses, their meaning, and how to determine the status of your instances, see [Monitoring an Amazon DocumentDB Instance's Status \(p. 404\)](#).

Amazon DocumentDB Instance Lifecycle

The lifecycle of an Amazon DocumentDB instance includes creating, modifying, maintaining and upgrading, performing backups and restores, rebooting, and deleting the instance. This section provides information about how to complete these processes.

Topics

- [Adding an Amazon DocumentDB Instance to a Cluster \(p. 319\)](#)
- [Describing Amazon DocumentDB Instances \(p. 323\)](#)
- [Modifying an Amazon DocumentDB Instance \(p. 326\)](#)
- [Rebooting an Amazon DocumentDB Instance \(p. 329\)](#)
- [Deleting an Amazon DocumentDB Instance \(p. 331\)](#)

Adding an Amazon DocumentDB Instance to a Cluster

You can create a new Amazon DocumentDB instance using the AWS Management Console or the AWS CLI. To add an instance to a cluster, the cluster must be in an *available* state. You cannot add an instance to a cluster that is stopped. If the cluster is stopped, first start the cluster, wait for the cluster to become *available*, and then add an instance. For more information, see [Stopping and Starting an Amazon DocumentDB Cluster \(p. 293\)](#).

Note

If you create an Amazon DocumentDB cluster using the console, an instance is automatically created for you at the same time. If you want to create additional instances, use one of the following procedures.

Adding an Instance Using the AWS Management Console

Use the following procedure to create an instance for your cluster using the Amazon DocumentDB console.

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Clusters**.

Tip

If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.

3. To choose the cluster that you want to add an instance to, select the button to the left of the cluster's name.
4. Choose **Actions**, and then choose **Add instances**.
5. In the **Add instance to: <cluster-name>** page, repeat the following steps for each instance that you want to add to the cluster. You can have up to 15.
 - a. **Instance identifier**— You can either enter a unique identifier for this instance or allow Amazon DocumentDB to provide the instance identifier based on the cluster identifier.

Instance naming constraints:

- Length is [1—63] letters, numbers, or hyphens.
 - First character must be a letter.
 - Cannot end with a hyphen or contain two consecutive hyphens.
 - Must be unique for all instances across Amazon RDS, Neptune, and Amazon DocumentDB per AWS account, per Region.
- b. **Instance class** — From the drop-down list, choose the instance type you want for this instance.
 - c. **Promotion tier** — From the drop-down list, choose the promotion tier for your instance or choose *No preference* to allow Amazon DocumentDB to set the promotion tier for your instance. Lower numbers mean higher priority. For more information, see [Controlling the Failover Target \(p. 345\)](#).
 - d. To add more instances, choose **Add additional instances** and repeat steps a, b, and c.
6. Finish the operation.
 - To add the instances to your cluster, choose **Create**.
 - To cancel the operation, choose **Cancel**.

It takes several minutes to create an instance. You can use the console or AWS CLI to view the instance's status. For more information, see [Monitoring an Instance's Status \(p. 404\)](#).

Adding an Instance Using the AWS CLI

Use the `create-db-instance` AWS CLI operation with the following parameters to create the primary instance for your cluster.

- **--db-instance-class** — Required. The compute and memory capacity of the instance, for example, `db.m4.large`. Not all instance classes are available in all AWS Regions.
- **--db-instance-identifier** — Required. A string that identifies the instance.

Instance Naming Constraints:

- Length is [1—63] letters, numbers, or hyphens.
 - First character must be a letter.
 - Cannot end with a hyphen or contain two consecutive hyphens.
 - Must be unique for all instances across Amazon RDS, Neptune, and Amazon DocumentDB per AWS account, per Region.
- **--engine** — Required. Must be `docdb`.
 - **--availability-zone** — Optional. The Availability Zone that you want this instance to be created in. Use this parameter to locate your instances in different Availability Zones to increase fault tolerance. For more information, see [Amazon DocumentDB High Availability and Replication \(p. 343\)](#).
 - **--promotion-tier** — Optional. The failover priority tier for this instance. Must be between 0 and 15 with lower numbers being higher priority. For more information, see [Controlling the Failover Target \(p. 345\)](#).

1. First, determine what Availability Zones you can create your instance in.

If you want to specify the Availability Zone before you create your instance, run the following command to determine which Availability Zones are available for your Amazon DocumentDB cluster.

For Linux, macOS, or Unix:

```
aws docdb describe-db-clusters \
    --query 'DBClusters[*].[DBClusterIdentifier,AvailabilityZones[*]]'
```

For Windows:

```
aws docdb describe-db-clusters ^
--query 'DBClusters[*].[DBClusterIdentifier,AvailabilityZones[*]]'
```

Output from this operation looks something like the following.

```
[  
  [  
    "sample-cluster",  
    [  
      "us-east-1c",  
      "us-east-1b",  
      "us-east-1a"  
    ]  
  ]  
]
```

2. Second, determine what instance classes you can create in your Region.

To determine which instance classes are available to you in your Region, run the following command. From the output, choose an instance class for the instance you want to add to your Amazon DocumentDB cluster.

For Linux, macOS, or Unix:

```
aws docdb describe-orderable-db-instance-options \
--engine docdb \
--query 'OrderableDBInstanceOptions[*].DBInstanceClass'
```

For Windows:

```
aws docdb describe-orderable-db-instance-options ^
--engine docdb ^
--query 'OrderableDBInstanceOptions[*].DBInstanceClass'
```

Output from this operation looks something like the following.

```
[  
  "db.r5.16xlarge",  
  "db.r5.2xlarge",  
  "db.r5.4xlarge",  
  "db.r5.8xlarge",  
  "db.r5.large",  
  "db.r5.xlarge"  
]
```

3. Finally, add an instance to your Amazon DocumentDB cluster.

To add an instance to your Amazon DocumentDB cluster, run the following command..

For Linux, macOS, or 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
```

For 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 from this operation looks something like the following.

```
{
  "DBInstance": {
    "DBInstanceIdentifier": "sample-instance-2",
    "DBInstanceClass": "db.r5.xlarge",
    "Engine": "docdb",
    "DBInstanceState": "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-ABCDEFGHIJKLMNPQRSTUVWXYZ",
    "CACertificateIdentifier": "rds-ca-2019",
    "ProcessorFeatures": [
      {
        "ProcessorFeatureType": "ProcessorFeatureType_1"
      }
    ],
    "ProcessorFeatureStatus": [
      {
        "ProcessorFeatureType": "ProcessorFeatureType_1"
      }
    ],
    "ProcessorFeatureArn": [
      "arn:aws:docdb:us-east-1:123456789012:processor:sample-processor"
    ],
    "ProcessorFeatureStatusList": [
      {
        "ProcessorFeatureType": "ProcessorFeatureType_1"
      }
    ],
    "ProcessorArn": "arn:aws:docdb:us-east-1:123456789012:processor:sample-processor"
  }
}
```

```

        "PromotionTier": 2,
        "DBInstanceArn": "arn:aws:rds:us-east-1:<accountID>:db:sample-instance-2"
    }
}

```

It takes several minutes to create the instance. You can use the console or AWS CLI to view the instance's status. For more information, see [Monitoring an Amazon DocumentDB Instance's Status \(p. 404\)](#).

Describing Amazon DocumentDB Instances

You can use either the Amazon DocumentDB Management Console or the AWS CLI to see details such as connection endpoints, security groups VPCs, certificate authority, and parameter groups pertaining to your Amazon DocumentDB instances.

Using the AWS Management Console

To view the details of your instances using the AWS Management Console, follow the steps below.

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Clusters**.

Tip

If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.

3. In the Clusters navigation box, you'll see the column **Cluster Identifier**. Your instances are listed under clusters, similar to the screenshot below.

Clusters (2)		
Filter Resources		
	Cluster identifier	Role
<input type="checkbox"/>	docdb-cloud9-getstarted	Cluster
<input type="checkbox"/>	robo3t	Cluster

4. In the list of instances, choose the name of the instance that you want to see its details. The information about the instance is organized into the following groupings:
 - **Summary**—General information about the instance, including the engine version, class, status, and any pending maintenance.
 - **Connectivity & Security**—The **Connect** section lists the connection endpoints to connect to this instance with the mongo shell or with an application. The **Security Groups** section lists the security groups associated with this instance and their VPC ID and descriptions.
 - **Configuration**—The **Details** section lists the configurations and status of the instance, including the instance's Amazon Resource Name (ARN), endpoint, role, class, and certificate authority. It also lists the instance's security and network settings, and backup information. The **Cluster details** section lists the details of the cluster that this instance belongs to. The **Cluster instances** section lists all the instances that belong to your cluster with each instance's role and cluster parameter group status.

Note

You can modify the cluster associated with your instance by selecting **Modify** next to the **Cluster details** header. For more information, see [Modifying an Amazon DocumentDB Cluster \(p. 287\)](#).

- **Monitoring**—The CloudWatch Logs metrics for this instance. For more information, see [Monitoring Amazon DocumentDB with CloudWatch \(p. 414\)](#).
- **Events & tags**—The **Recent events** section lists the recent events for this instance. Amazon DocumentDB keeps a record of events that relate to your clusters, instances, snapshots, security groups, and cluster parameter groups. This information includes the date, time, and message associated with each event. The **<guilablel>Tags</guilablel>** section lists the tags attached to this cluster. For more information, see [Tagging Amazon DocumentDB Resources \(p. 387\)](#).

Using the AWS CLI

To view the details of your Amazon DocumentDB instances using the AWS CLI, use the `describe-db-clusters` command as shown in the examples below. For more information, see [DescribeDBInstances](#) in the *Amazon DocumentDB Resource Management API Reference*.

Note

For certain management features such as cluster and instance lifecycle management, Amazon DocumentDB leverages operational technology that is shared with Amazon RDS. The `filterName=engine,Values=docdb` filter parameter returns only Amazon DocumentDB clusters.

1. List all Amazon DocumentDB instances.

The following AWS CLI code lists the details for all Amazon DocumentDB instances in a region.

For Linux, macOS, or Unix:

```
aws docdb describe-db-instances \
--filter Name=engine,Values=docdb
```

For Windows:

```
aws docdb describe-db-instances \
--filter Name=engine,Values=docdb
```

2. List all details for a specified Amazon DocumentDB instance

The following code lists the details for `sample-cluster-instance`. Including the `--db-instance-identifier` parameter with the name of an instance restricts the output to information on that particular instance.

For Linux, macOS, or Unix:

```
aws docdb describe-db-instances \
--db-instance-identifier sample-cluster-instance
```

For Windows:

```
aws docdb describe-db-instances \
--db-instance-identifier sample-cluster-instance
```

Output from this operation looks like the following.

```
{
    "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",
            "DBInstanceState": "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"
}
]
}

```

Modifying an Amazon DocumentDB Instance

You can modify your Amazon DocumentDB instance using either the AWS Management Console or the AWS CLI. To modify an instance, the instance must be in the *available* state. You cannot modify an instance that is stopped. If the cluster is stopped, first start the cluster, wait for the instance to become *available*, and then make the desired modifications. For more information, see [Stopping and Starting an Amazon DocumentDB Cluster \(p. 293\)](#).

Using the AWS Management Console

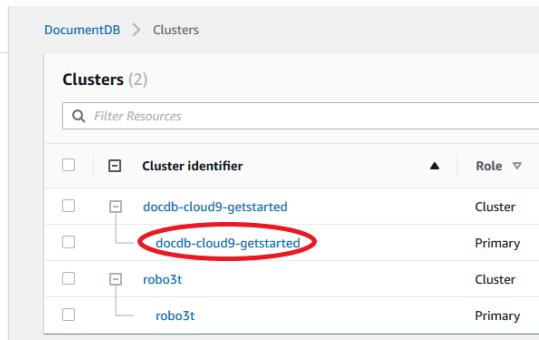
To modify a specific Amazon DocumentDB instance using the console, complete the following steps.

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Clusters**.

Tip

If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.

3. In the Clusters navigation box, you'll see the column **Cluster Identifier**. Your instances are listed under clusters, similar to the screenshot below.



Cluster Identifier	Role
docdb-cloud9-getstarted	Primary
robo3t	Cluster

4. Check the box to the left of the instance you wish to modify.
5. Choose **Actions**, and then choose **Modify**.

6. In the **Modify instance: <instance-name>** pane, make the changes that you want. You can make the following changes:
 - **Instance specifications** — The instance identifier and class. Instance identifier naming constraints:
 - **Instance identifier** — Enter a name that is unique for all instances owned by your AWS account in the current region. The instance identifier must contain [1–63] alphanumeric characters or hyphens, have a letter as the first character, and cannot end with a hyphen or contain two consecutive hyphens.
 - **Instance class** — From the drop-down menu, select an instance class for your Amazon DocumentDB instance. For more information, see [Managing Instance Classes \(p. 313\)](#).
 - **Certificate authority** — Server certificate for this instance. For more information, see [Updating Your Amazon DocumentDB TLS Certificates \(p. 187\)](#).
 - **Failover** — During failover, the instance with the highest promotion tier will be promoted to primary. For more information, see [Amazon DocumentDB Failover \(p. 345\)](#).
 - **Maintenance** — The maintenance window in which pending modifications or patches are applied to instances in the cluster.
7. When you have finished, choose **Continue** to see a summary of your changes.
8. After verifying your changes, you can apply them immediately or during the next maintenance window under **Scheduling of modifications**. Choose **Modify instance** to save your changes. Alternatively, you can choose **Cancel** to discard your changes.

It takes a few minutes for your changes to be applied. You can use the instance only when its status is *available*. You can monitor the instance's status using the console or AWS CLI. For more information, see [Monitoring an Amazon DocumentDB Instance's Status \(p. 404\)](#).

Using the AWS CLI

To modify a specific Amazon DocumentDB instance using the AWS CLI, use the `modify-db-instance` with the following parameters. For more information, see [ModifyDBInstance](#). The following code modifies the instance class to `db.r5.large` for the instance `sample-instance`.

Parameters

- **--db-instance-identifier** — Required. The identifier for the instance to be modified.
- **--db-instance-class** — Optional. The new compute and memory capacity of the instance; for example, `db.r5.large`. Not all instance classes are available in all AWS Regions. If you modify the instance class, an outage occurs during the change. The change is applied during the next maintenance window, unless `ApplyImmediately` is specified as true for this request.
- **--apply-immediately** or **--no-apply-immediately** — Optional. Specifies whether this modification should be applied immediately or wait until the next maintenance window. If this parameter is omitted, the modification is performed during the next maintenance window.

Example

For Linux, macOS, or Unix:

```
aws docdb modify-db-instance \
    --db-instance-identifier sample-instance \
    --db-instance-class db.r5.large \
    --apply-immediately
```

For Windows:

```
aws docdb modify-db-instance ^
```

```
--db-instance-identifier sample-instance ^
--db-instance-class db.r5.large ^
--apply-immediately
```

Output from this operation looks something like the following.

```
{
    "DBInstances": [
        {
            "DBInstanceIdentifier": "sample-instance-1",
            "DBInstanceClass": "db.r5.large",
            "Engine": "docdb",
            "DBInstanceState": "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"
        ]
    }
}
```

It takes a few minutes for your modifications to be applied. You can use the instance only when its status is *available*. You can monitor the instance's status using the AWS Management Console or AWS CLI. For more information, see [Monitoring an Amazon DocumentDB Instance's Status \(p. 404\)](#).

Rebooting an Amazon DocumentDB Instance

Occasionally, you might need to reboot your Amazon DocumentDB instance, usually for maintenance reasons. If you make certain changes, such as changing the cluster parameter group that is associated with a cluster, you must reboot the instances in the cluster for the changes to take effect. You can reboot a specified instance using the AWS Management Console or the AWS CLI.

Rebooting an instance restarts the database engine service. Rebooting results in a momentary outage, during which the instance status is set to `rebooting`. An Amazon DocumentDB event is created when the reboot is completed.

Rebooting an instance doesn't result in a failover. To failover an Amazon DocumentDB cluster, use the AWS Management Console or the AWS CLI operation `failover-db-cluster`. For more information, see [Amazon DocumentDB Failover \(p. 345\)](#).

You can't reboot your instance if it isn't in the *available* state. Your database can be unavailable for several reasons, such as a previously requested modification, or a maintenance-window action. For more information on instance states, see [Monitoring an Amazon DocumentDB Instance's Status \(p. 404\)](#).

Rebooting an Instance Using the AWS Management Console

The following procedure reboots an instance that you specify using the console.

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Clusters**.

Tip

If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.

3. In the Clusters navigation box, you'll see the column **Cluster Identifier**. Your instances are listed under clusters, similar to the screenshot below.

Clusters (2)		
Cluster identifier	Role	Status
docdb-cloud9-getstarted	Cluster	Primary
robo3t	Cluster	Primary

4. Check the box to the left of the instance you wish to reboot.
5. Choose **Actions**, choose **Reboot**, and then choose **Reboot** to confirm your reboot.

It takes a few minutes for your instance to reboot. You can use the instance only when its status is *available*. You can monitor the instance's status using the console or the AWS CLI. For more information, see [Monitoring an Amazon DocumentDB Instance's Status \(p. 404\)](#).

Rebooting an Instance Using the AWS CLI

To reboot an Amazon DocumentDB instance, use the `reboot-db-instance` operation with the `--db-instance-identifier` parameter. This parameter specifies the identifier for the instance to be rebooted.

The following code reboots the instance `sample-instance`.

Example

For Linux, macOS, or Unix:

```
aws docdb reboot-db-instance \
    --db-instance-identifier sample-instance
```

For Windows:

```
aws docdb reboot-db-instance ^
    --db-instance-identifier sample-instance
```

Output from this operation looks something like the following.

```
{
  "DBInstance": {
    "DBInstanceIdentifier": "sample-instance",
    "DBInstanceClass": "db.r5.large",
    "Engine": "docdb",
    "DBInstanceState": "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,
"DBInstanceStateArn": "arn:aws:rds:us-east-1:<accountID>:db:sample-instance",
"EnabledCloudwatchLogsExports": [
    "profiler"
]
}
}
```

It takes a few minutes for your instance to reboot. You can use the instance only when its status is *available*. You can monitor the instance's status using the console or AWS CLI. For more information, see [Monitoring an Amazon DocumentDB Instance's Status \(p. 404\)](#).

Deleting an Amazon DocumentDB Instance

You can delete your Amazon DocumentDB instance using either the AWS Management Console or the AWS CLI. To delete an instance, the instance must be in the *available* state. You cannot delete an instance that is stopped. If the Amazon DocumentDB cluster that contains your instance is stopped, first start the cluster, wait for the instance to become *available*, and then delete the instance. For more information, see [Stopping and Starting an Amazon DocumentDB Cluster \(p. 293\)](#).

Note

Amazon DocumentDB stores all of your data in the cluster volume. The data persists in that cluster volume, even if you remove all the instances from your cluster. If you need to access the data again, you can add an instance to the cluster at any time and pick up where you left off.

Deletion Protection

Deleting the last instance of an Amazon DocumentDB cluster will also delete the cluster, as well as the automatic snapshots and continuous backups associated with that cluster. Amazon DocumentDB enforces deletion protection for a cluster whether you perform the delete operation using the AWS Management Console or the AWS CLI. If deletion protection is enabled, you can't delete a cluster.

To delete a cluster that has deletion protection enabled, you must first modify the cluster and disable deletion protection. For more information, see [Deleting an Amazon DocumentDB Cluster \(p. 296\)](#).

Deleting an Instance Using the AWS Management Console

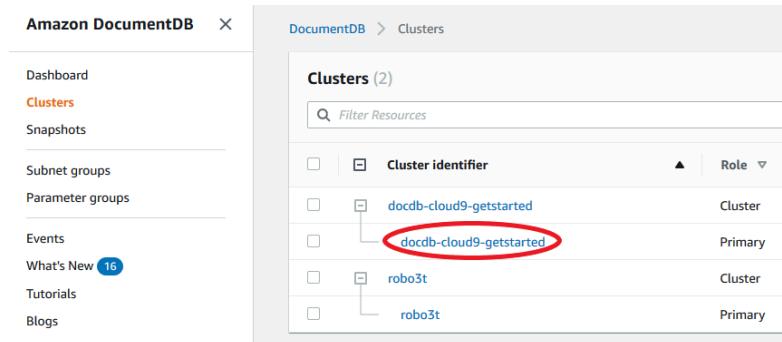
The following procedure deletes a specified Amazon DocumentDB instance using the console.

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Clusters**.

Tip

If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.

3. In the Clusters navigation box, you'll see the column **Cluster Identifier**. Your instances are listed under clusters, similar to the screenshot below.



The screenshot shows the Amazon DocumentDB console interface. On the left, there's a navigation sidebar with links like Dashboard, Clusters (which is selected and highlighted in blue), Snapshots, Subnet groups, Parameter groups, Events, What's New (with 16 items), Tutorials, and Blogs. The main content area is titled "Clusters (2)". It shows a table with two rows. The first row is for the cluster "docdb-cloud9-getstarted", which has one primary instance named "docdb-cloud9-getstarted". The second row is for the cluster "robo3t", which also has one primary instance named "robo3t". The instance "docdb-cloud9-getstarted" is circled in red.

Cluster identifier	Role
docdb-cloud9-getstarted	Cluster
docdb-cloud9-getstarted	Primary
robo3t	Cluster
robo3t	Primary

4. Check the box to the left of the instance you wish to delete.
5. Select **Actions**, and then choose **Delete**.

1. If you are deleting the last instance in your cluster:
 - **Create final cluster snapshot?** — Choose **Yes** if you want to create a final snapshot before the cluster is deleted. Otherwise, choose **No**.
 - **Final snapshot name** — If you choose to create a final snapshot, enter the cluster snapshot identifier of the new cluster snapshot created.
 - **Delete <instance-name> instance?** — Enter the phrase **delete entire cluster** into the field to confirm the deletion.
2. If you are not deleting the last instance in your cluster:
 - **Delete <instance-name> instance?** — Enter the phrase **delete me** into the field to confirm the deletion.
6. Select **Delete** to delete the instance.

It takes several minutes for an instance to be deleted. To monitor the status of an instance, see [Monitoring an Amazon DocumentDB Instance's Status \(p. 404\)](#).

Deleting an Instance Using the AWS CLI

The following procedure deletes an Amazon DocumentDB instance using the AWS CLI.

1. **First, determine how many instances are in your Amazon DocumentDB cluster:**

To determine how many instances are in your cluster, run the `describe-db-clusters` command, as follows.

```
aws docdb describe-db-clusters \
  --db-cluster-identifier sample-cluster \
  --query 'DBClusters[*].[DBClusterIdentifier,DBClusterMembers[*].DBInstanceIdentifier]'
```

Output from this operation looks something like the following.

```
[
```

```
        "sample-cluster",
        [
            "sample-instance-1",
            "sample-instance-2"
        ]
    ]
```

2. If there are more than one instances in your Amazon DocumentDB cluster:

To delete a specified Amazon DocumentDB instance, use the `delete-db-instance` command with the `--db-instance-identifier` parameter, as shown below. It takes several minutes for an instance to be deleted. To monitor the status of an instance, see [Monitoring an Amazon DocumentDB Instance's Status \(p. 404\)](#).

```
aws docdb delete-db-instance \
--db-instance-identifier sample-instance-2
```

Output from this operation looks something like the following.

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "sample-instance-2",
        "DBInstanceClass": "db.r5.large",
        "Engine": "docdb",
        "DBInstanceState": "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-ABCDEFIGHIJKLMNOPQRSTUVWXYZ",
"CACertificateIdentifier": "rds-ca-2019",
"PromotionTier": 1,
"DBInstanceArn": "arn:aws:rds:us-east-1:<accountID>:db:sample-instance-2",
"EnabledCloudwatchLogsExports": [
    "profiler"
]
}
}
```

3. If the instance you want to delete is the last instance in your Amazon DocumentDB cluster:

If you delete the last instance in an Amazon DocumentDB cluster, you also delete that cluster and the automatic snapshots and continuous backups associated with that cluster.

To delete the last instance in your cluster, you can delete the cluster and optionally create a final snapshot. For more information, see [Deleting an Amazon DocumentDB Cluster \(p. 296\)](#).

Managing Amazon DocumentDB Subnet Groups

A virtual private cloud (VPC) is a virtual network dedicated to your AWS account. It is logically isolated from other virtual networks in the AWS Cloud. You can launch your AWS resources, such as Amazon DocumentDB clusters, into your Amazon VPC. You can specify an IP address range for the VPC, add subnets, associate security groups, and configure route tables.

A subnet is a range of IP addresses in your Amazon VPC. You can launch AWS resources into a specified subnet. Use a *public* subnet for resources that must be connected to the internet. Use a *private* subnet for resources that won't be connected to the internet. For more information about public and private subnets, see [VPC and Subnet Basics](#) in the *Amazon Virtual Private Cloud User Guide*.

A DB subnet group is a collection of subnets that you create in a VPC that you then designate for your clusters. A subnet group allows you to specify a particular VPC when creating clusters. If you use the default subnet group, it spans all subnets in the VPC.

Each DB subnet group should have subnets in at least two Availability Zones in a given Region. When creating a DB cluster in a VPC, you must select a DB subnet group. Amazon DocumentDB uses that DB subnet group and your preferred Availability Zone to select a subnet and an IP address within that subnet to associate with your cluster. If the primary instance fails, Amazon DocumentDB can promote a corresponding replica instance to be the new primary. It can then create a new replica instance using an IP address of the subnet in which the previous primary was located.

When Amazon DocumentDB creates an instance in a VPC, it assigns a network interface to your cluster by using an IP address selected from your DB subnet group. We strongly recommend that you use the DNS name because the underlying IP address can change during failover. For more information, see [Amazon DocumentDB Endpoints \(p. 8\)](#).

For information about creating your own VPC and subnets, see [Working with VPCs and Subnets](#) in the *Amazon Virtual Private Cloud User Guide*.

Topics

- [Creating an Amazon DocumentDB Subnet Group \(p. 335\)](#)
- [Describing an Amazon DocumentDB Subnet Group \(p. 338\)](#)
- [Modifying an Amazon DocumentDB Subnet Group \(p. 340\)](#)
- [Deleting an Amazon DocumentDB Subnet Group \(p. 342\)](#)

Creating an Amazon DocumentDB Subnet Group

When creating an Amazon DocumentDB cluster, you must choose a Amazon VPC and corresponding subnet group within that Amazon VPC to launch your cluster. Subnets determine the availability zone and IP range within the availability zone that you want to use to launch an instance.

A subnet group is a named set of subnets (or AZs) that allows you to specify the availability zones that you want to use to for launching Amazon DocumentDB instances. For example, in a cluster with three instances, it is recommended that each of those instances are provisioned in separate AZs—doing so optimizes for high availability. Thus, if a single AZ fails, it will only affect a single instance.

Currently, Amazon DocumentDB instances can be provisioned in up to three AZs. Even if a subnet group has more than three subnets, you will only be able to use three of those subnets to create an Amazon DocumentDB cluster. Therefore, we recommend that when you create a subnet group that you only choose the three subnets of which you want to deploy your instances.

For example: A cluster is created and Amazon DocumentDB choose AZs {1A, 1B, and 1C}. If you attempt to create an instance in AZ {1D} the API call will fail. However, if you choose to create an instance, without specifying the particular AZ, then Amazon DocumentDB will choose an AZ on your behalf. Amazon DocumentDB uses an algorithm to load balance the instances across AZs to help you achieve high availability. If three instances are provisioned, by default, they will be provisioned across three AZs and will not be provisioned all in a single AZ.

Best Practices

- Unless you have a specific reason, always create a subnet group with three subnets. This ensures that clusters with three or more instances will be able to achieve higher availability as instances will be provisioned across three AZs.
- Always spread instances across multiple AZs to achieve high availability. Never place all instances for a cluster in a single AZ.
- Because failover events can happen at any time, you should not assume that a primary instance or replica instances will always be in a particular AZ.

How to create a subnet group

You can use the AWS Management Console or AWS CLI to create an Amazon DocumentDB subnet group.

Topics

- [Using the AWS Management Console \(p. 335\)](#)
- [Using the AWS CLI \(p. 336\)](#)

Using the AWS Management Console

Use the following steps to create an Amazon DocumentDB subnet group.

To create an Amazon DocumentDB subnet group

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.

2. In the navigation pane, choose **Subnet groups**, then choose **Create**.

Tip

If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.

3. On the **Create subnet group** page:

- a. In the **Subnet group details** section:
 - i. **Name**—Enter a meaningful name for the subnet group.
 - ii. **Description**—Enter a description for the subnet group.
- b. In the **Add subnets** section:
 - i. **VPC**—In the list, choose a VPC for this subnet group.
 - ii. Do one of the following:
 - To include all subnets in the chosen VPC, choose **Add all the subnets related to this VPC**.
 - To specify subnets for this subnet group, do the following for each Availability Zone for which you want to include subnets. You must include at least two Availability Zones.
 - A. **Availability zone**—In the list, choose an Availability Zone.
 - B. **Subnet**—In the list, choose a subnet from the chosen Availability Zone for this subnet group.
 - C. Choose **Add subnet**.

4. Choose **Create**. When the subnet group is created, it is listed with your other subnet groups.

Subnet groups (2)			
<input type="text"/> Filter subnet groups			
Name	Description	Status	VPC
default	default	Complete	vpc-91280df6
sample-subnet-group	A sample subnet group	Complete	vpc-91280df6

Using the AWS CLI

Before you can create a subnet group using the AWS CLI, you must first determine which subnets are available. Run the following AWS CLI operation to list the Availability Zones and their subnets.

Parameters:

- **--db-subnet-group**—Optional. Specifying a particular subnet group lists the Availability Zones and subnets for that group. Omitting this parameter lists Availability Zones and subnets for all your subnet groups. Specifying the `default` subnet group lists all the VPC's subnets.

Example

For Linux, macOS, or Unix:

```
aws docdb describe-db-subnet-groups \
--db-subnet-group-name default \
--query 'DBSubnetGroups[*].[DBSubnetGroupName,Subnets[*].\
[SubnetAvailabilityZone.Name,SubnetIdentifier]]'
```

For Windows:

```
aws docdb describe-db-subnet-groups ^
--db-subnet-group-name default ^
--query 'DBSubnetGroups[*].[DBSubnetGroupName,Subnets[*].[
[SubnetAvailabilityZone.Name,SubnetIdentifier]]'
```

Output from this operation looks something like the following (JSON format).

```
[  
  [  
    "default",  
    [  
      [  
        ["us-east-1a",  
         "subnet-4e26d263"]  
      ],  
      [  
        ["us-east-1c",  
         "subnet-afc329f4"]  
      ],  
      [  
        ["us-east-1e",  
         "subnet-b3806e8f"]  
      ],  
      [  
        ["us-east-1d",  
         "subnet-53ab3636"]  
      ],  
      [  
        ["us-east-1b",  
         "subnet-991cb8d0"]  
      ],  
      [  
        ["us-east-1f",  
         "subnet-29ab1025"]  
      ]  
    ]  
  ]
```

Using the output from the previous operation, you can create a new subnet group. The new subnet group must include subnets from at least two Availability Zones.

Parameters:

- **--db-subnet-group-name**—Required. The name for this subnet group.
- **--db-subnet-group-description**—Required. The description of this subnet group.
- **--subnet-ids**—Required. A list of subnets to include in this subnet group. Example: subnet-53ab3636.
- **--Tags**—Optional. A list of tags (key-value pairs) to attach to this subnet group.

The following code creates the subnet group *sample-subnet-group* with three subnets, subnet-4e26d263, subnet-afc329f4, and subnet-b3806e8f.

For Linux, macOS, or 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
```

For 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
```

Output from this operation looks something like the following (JSON format).

```
{
    "DBSubnetGroup": {
        "DBSubnetGroupDescription": "A sample subnet group",
        "DBSubnetGroupName": "sample-subnet-group",
        "Subnets": [
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1a"
                },
                "SubnetIdentifier": "subnet-4e26d263",
                "SubnetStatus": "Active"
            },
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1c"
                },
                "SubnetIdentifier": "subnet-afc329f4",
                "SubnetStatus": "Active"
            },
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1e"
                },
                "SubnetIdentifier": "subnet-b3806e8f",
                "SubnetStatus": "Active"
            }
        ],
        "VpcId": "vpc-91280df6",
        "DBSubnetGroupArn": "arn:aws:rds:us-east-1:123SAMPLE012:subgrp:sample-subnet-
group",
        "SubnetGroupStatus": "Complete"
    }
}
```

Describing an Amazon DocumentDB Subnet Group

You can use the AWS Management Console or the AWS CLI to get the details of an Amazon DocumentDB subnet group.

Using the AWS Management Console

The following procedure shows you how to get the details of an Amazon DocumentDB subnet group.

To find the details of a subnet group

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Subnet groups**.

Tip

If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.

3. To see the details of a subnet group, choose the name of that subnet group.

Subnet group details		
VPC ID	vpc-91280df6	
ARN	arn:aws:rds:us-east-1:██████████:subgrp:sample-subnet-group	
Description	A sample subnet group	
Subnet group status	Complete	
Subnets (3)		
Availability zone	Subnet ID	Subnet group status
us-east-1a	subnet-4e26d263	Active
us-east-1c	subnet-afc329f4	Active
us-east-1e	subnet-b3806e8f	Active
Tags (2)		
Key	Value	
tag1	One	
tag2	2	

Using the AWS CLI

To find the details of an Amazon DocumentDB subnet group, use the `describe-db-subnet-groups` operation with the following parameter.

Parameter

- `--db-subnet=group-name`—Optional. If included, details for the named subnet group are listed. If omitted, details for up to 100 subnet groups are listed.

Example

The following code lists details for the `sample-subnet-group` subnet group that we created in the [Creating an Amazon DocumentDB Subnet Group \(p. 335\)](#) section.

For Linux, macOS, or Unix:

```
aws docdb describe-db-subnet-groups \
--db-subnet-group-name sample-subnet-group
```

For Windows:

```
aws docdb describe-db-subnet-groups ^
--db-subnet-group-name sample-subnet-group
```

Output from this operation looks something like the following (JSON format).

```
{  
    "DBSubnetGroup": {
```

```
        "DBSubnetGroupArn": "arn:aws:rds:us-east-1:123SAMPLE012:subgrp:sample-subnet-group",
        "VpcId": "vpc-91280df6",
        "SubnetGroupStatus": "Complete",
        "DBSubnetGroupName": "sample-subnet-group",
        "Subnets": [
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1a"
                },
                "SubnetStatus": "Active",
                "SubnetIdentifier": "subnet-4e26d263"
            },
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1c"
                },
                "SubnetStatus": "Active",
                "SubnetIdentifier": "subnet-afc329f4"
            },
            {
                "SubnetAvailabilityZone": {
                    "Name": "us-east-1e"
                },
                "SubnetStatus": "Active",
                "SubnetIdentifier": "subnet-b3806e8f"
            }
        ],
        "DBSubnetGroupDescription": "A sample subnet group"
    }
}
```

Modifying an Amazon DocumentDB Subnet Group

You can use the AWS Management Console or AWS CLI to modify a subnet group's description or to add or remove subnets from an Amazon DocumentDB subnet group. However, you cannot modify the default subnet group.

Topics

- [Using the AWS Management Console \(p. 340\)](#)
- [Using the AWS CLI \(p. 341\)](#)

Using the AWS Management Console

You can use the AWS Management Console to change a subnet group's description or to add and remove subnets. Remember that when you're finished, you must have at least two Availability Zones associated with your subnet group.

To modify your subnet group

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Subnet groups**. Then choose the button to the left of the subnet group's name. Remember that you can't modify the default subnet group.

Tip

If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.

3. Choose **Actions**, and then choose **Modify**.
4. **Description**—To change the description of your subnet group, enter a new description.
5. To change the subnets associated with your subnet group, in the **Add subnets** section, do any one or more of the following:
 - To remove all subnets from this subnet group, choose **Remove all**.
 - To remove specific subnets from this subnet group, choose **Remove** for each subnet you want to remove.
 - To add all the subnets associated with this VPC, choose **Add all the subnets related to this VPC**.
 - To add specific subnets to this subnet group, do the following for each Availability Zone for which you want to add a subnet.
 - a. **Availability zone**—In the list, choose a new Availability Zone.
 - b. **Subnet**—In the list, choose a subnet from the chosen Availability Zone for this subnet group.
 - c. Choose **Add subnet**.
6. In the confirmation dialog box:
 - To make these changes to the subnet group, choose **Modify**.
 - To keep the subnet group unchanged, choose **Cancel**.

Using the AWS CLI

You can use the AWS CLI to change a subnet group's description or to add and remove subnets. Remember that when you're finished, you must have at least two Availability Zones associated with your subnet group. You can't modify the default subnet group.

Parameters:

- **--db-subnet-group-name**—Required. The name of the Amazon DocumentDB subnet group you are modifying.
- **--subnet-ids**—Required. A list of all the subnets that you want in the subnet group after this change is done.

Important

Any subnets currently in the subnet group that are not included in this list are removed from the subnet group. If you want to keep any of the subnets currently in the subnet group, you must include them in this list.

- **--db-subnet-group-description**—Optional. The description of the subnet group.

Example

The following code modifies the description and replaces the existing subnets with the subnets `subnet-991cb8d0`, `subnet-53ab3636`, and `subnet-29ab1025`.

For Linux, macOS, or 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"
```

For 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"
```

Output from this operation looks something like the following (JSON format). Notice that this is the same subnet group that was created in the [Creating an Amazon DocumentDB Subnet Group \(p. 335\)](#) section. However, the subnets in the subnet group are replaced with those listed in the modify-db-subnet-group operation.

```
{
    "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"
    }
}
```

Deleting an Amazon DocumentDB Subnet Group

You can use the AWS Management Console or AWS CLI to delete an Amazon DocumentDB subnet group. However, you cannot delete the default subnet group.

Topics

- [Using the AWS Management Console \(p. 342\)](#)
- [Using the AWS CLI \(p. 343\)](#)

Using the AWS Management Console

You can use the AWS Management Console to delete a subnet group. But you can't delete the default subnet group.

To delete a subnet group

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Subnet groups**. Then choose the button to the left of the subnet group's name. Remember that you can't delete the default subnet group.

Tip

If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.

3. Choose **Actions**, and then choose **Delete**.
4. In the confirmation dialog box:
 - To delete the subnet group, choose **Delete**.
 - To keep the subnet group, choose **Cancel**.

Using the AWS CLI

To delete an Amazon DocumentDB subnet group using the AWS CLI, use the `delete-db-subnet-group` operation with the following parameter.

Parameter

- `--db-subnet-group-name`—Required. The name of the Amazon DocumentDB subnet group to delete. Remember that you can't delete the default subnet group.

Example

The following code deletes `sample-subnet-group`.

For Linux, macOS, or Unix:

```
aws docdb delete-db-subnet-group \
--db-subnet-group-name sample-subnet-group
```

For Windows:

```
aws docdb delete-db-subnet-group ^
--db-subnet-group-name sample-subnet-group
```

This operation produces no output.

Amazon DocumentDB High Availability and Replication

You can achieve high availability and read scaling in Amazon DocumentDB (with MongoDB compatibility) by using replica instances. A single Amazon DocumentDB cluster supports a single primary instance and up to 15 replica instances. These instances can be distributed across Availability Zones within the cluster's Region. The primary instance accepts read and write traffic, and replica instances accept only read requests.

The cluster volume is made up of multiple copies of the data for the cluster. However, the data in the cluster volume is represented as a single, logical volume to the primary instance and to Amazon DocumentDB replicas in the cluster. Replica instances are eventually consistent. They return query results with minimal replica lag—usually much less than 100 milliseconds after the primary instance has written an update. Replica lag varies depending on the rate of database change. That is, during periods in which a large number of write operations occur for the database, you might see an increase in the replica lag.

Read Scaling

Amazon DocumentDB replicas work well for read scaling because they are fully dedicated to read operations on your cluster volume. Write operations are managed by the primary instance. The cluster volume is shared among all instances in your cluster. Therefore, you don't have to replicate and maintain a copy of the data for each Amazon DocumentDB replica.

High Availability

When you create an Amazon DocumentDB cluster, depending upon the number of Availability Zones in the subnet group (there must be at least two), Amazon DocumentDB provisions instances across the Availability Zones. When you create instances in the cluster, Amazon DocumentDB automatically distributes the instances across the Availability Zones in a subnet group to balance the cluster. This action also prevents all instances from being located in the same Availability Zone.

Example

To illustrate the point, consider an example where you create a cluster that has a subnet group with three Availability Zones: AZ1, AZ2, and AZ3.

When the first instance in the cluster is created, it is the primary instance and is located in one of the Availability Zones. In this example, it's in AZ1. The second instance created is a replica instance and is located in one of the other two Availability Zones, say AZ2. The third instance created is a replica instance and is located in the remaining Availability Zone, AZ3. If you create more instances, they are distributed across the Availability Zones so that you achieve balance in the cluster.

If a failure occurs in the primary instance (AZ1), a failover is triggered, and one of the existing replicas is promoted to primary. When the old primary recovers, it becomes a replica in the same Availability Zone in which it was provisioned (AZ1). When you provision a three-instance cluster, Amazon DocumentDB continues to preserve that three-instance cluster. Amazon DocumentDB automatically handles detection, failover, and recovery of instance failures without any manual intervention.

When Amazon DocumentDB performs a failover and recovers an instance, the recovered instance remains in the Availability Zone in which it was originally provisioned. However, the role of the instance might change from primary to replica. Doing this prevents the scenario in which a series of failovers could result in all instances being in the same Availability Zone.

You can specify Amazon DocumentDB replicas as failover targets. That is, if the primary instance fails, the specified Amazon DocumentDB replica or replica from a tier is promoted to the primary instance. There is a brief interruption during which read and write requests made to the primary instance fail with an exception. If your Amazon DocumentDB cluster doesn't include any Amazon DocumentDB replicas, when the primary instance fails, it is re-created. Promoting an Amazon DocumentDB replica is much faster than re-creating the primary instance.

For high availability scenarios, we recommend that you create one or more Amazon DocumentDB replicas. These replicas should be of the same instance class as the primary instance and in different Availability Zones for your Amazon DocumentDB cluster.

For more information, see the following:

- [Understanding Amazon DocumentDB Cluster Fault Tolerance \(p. 312\)](#)
- [Amazon DocumentDB Failover \(p. 345\)](#)
 - [Controlling the Failover Target \(p. 345\)](#)

High Availability with Global Clusters

For high availability across multiple AWS Regions, you can set up [Amazon DocumentDB global clusters](#). Each global cluster spans multiple regions, enabling low latency global reads and disaster recovery from outages across an AWS Region. Amazon DocumentDB automatically handles replicating all data and updates from the primary region to each of the secondary regions.

Adding Replicas

The first instance added to the cluster is the primary instance. Every instance that is added after the first instance is a replica instance. A cluster can have up to 15 replica instances in addition to the primary.

When you create a cluster using the AWS Management Console, a primary instance is automatically created at the same time. To create a replica at the same time as you create the cluster and the primary instance, choose **Create replica in different zone**. For more information, see step 4.d in [Creating an Amazon DocumentDB Cluster \(p. 274\)](#). To add more replicas to an Amazon DocumentDB cluster, see [Adding an Amazon DocumentDB Instance to a Cluster \(p. 319\)](#).

When using the AWS CLI to create your cluster, you must explicitly create your primary and replica instances. For more information, see the "Using the AWS CLI" section in the following topics:

- [Creating an Amazon DocumentDB Cluster \(p. 274\)](#)
- [Adding an Amazon DocumentDB Instance to a Cluster \(p. 319\)](#)

Amazon DocumentDB Failover

In certain cases, such as certain types of planned maintenance, or in the unlikely event of a primary node or Availability Zone failure, Amazon DocumentDB (with MongoDB compatibility) detects the failure and replaces the primary node. During a failover, write down time is minimized. This is because the role of primary node fails over to one of the read replicas instead of having to create and provision a new primary node. This failure detection and replica promotion ensure that you can resume writing to the new primary as soon as promotion is complete.

For failover to function, your cluster must have at least two instances — a primary and at least one replica instance.

Controlling the Failover Target

Amazon DocumentDB provides you with failover tiers as a means to control which replica instance is promoted to primary when a failover occurs.

Failover Tiers

Each replica instance is associated with a failover tier (0–15). When a failover occurs due to maintenance or an unlikely hardware failure, the primary instance fails over to a replica with the highest priority (the lowest numbered tier). If multiple replicas have the same priority tier, the primary fails over to that tier's replica that is the closest in size to the previous primary.

By setting the failover tier for a group of select replicas to `0` (the highest priority), you can ensure that a failover will promote one of the replicas in that group. You can effectively prevent specific replicas from being promoted to primary in case of a failover by assigning a low-priority tier (high number) to these replicas. This is useful in cases where specific replicas are receiving heavy use by an application and failing over to one of them would negatively impact a critical application.

You can set the failover tier of an instance when you create it or later by modifying it. Setting an instance failover tier by modifying the instance does not trigger a failover. For more information see the following topics:

- [Adding an Amazon DocumentDB Instance to a Cluster \(p. 319\)](#)
- [Modifying an Amazon DocumentDB Instance \(p. 326\)](#)

When manually initiating a failover, you have two means to control which replica instance is promoted to primary: the failover tiers as previously described, and the `--target-db-instance-identifier` parameter.

--target-db-instance-identifier

For testing, you can force a failover event using the `failover-db-cluster` operation. You can use the `--target-db-instance-identifier` parameter to specify which replica to promote to primary. Using the `--target-db-instance-identifier` parameter supersedes the failover priority tier. If you do not specify the `--target-db-instance-identifier` parameter, the primary failover is in accordance with the failover priority tier.

What Happens During a Failover

Failover is automatically handled by Amazon DocumentDB so that your applications can resume database operations as quickly as possible without administrative intervention.

- If you have an Amazon DocumentDB replica instance in the same or different Availability Zone when failing over: Amazon DocumentDB flips the canonical name record (CNAME) for your instance to point at the healthy replica, which is, in turn, promoted to become the new primary. Failover typically completes within 30 seconds from start to finish.
- If you don't have an Amazon DocumentDB replica instance (for example, a single instance cluster): Amazon DocumentDB will attempt to create a new instance in the same Availability Zone as the original instance. This replacement of the original instance is done on a best-effort basis and may not succeed if, for example, there is an issue that is broadly affecting the Availability Zone.

Your application should retry database connections in the event of a connection loss.

Testing Failover

A failover for a cluster promotes one of the Amazon DocumentDB replicas (read-only instances) in the cluster to be the primary instance (the cluster writer).

When the primary instance fails, Amazon DocumentDB automatically fails over to an Amazon DocumentDB replica, if one exists. You can force a failover when you want to simulate a failure of a primary instance for testing. Each instance in a cluster has its own endpoint address. Therefore, you need to clean up and re-establish any existing connections that use those endpoint addresses when the failover is complete.

To force a failover, use the `failover-db-cluster` operation with these parameters.

- `--db-cluster-identifier`—Required. The name of the cluster to fail over.

- **--target-db-instance-identifier**—Optional. The name of the instance to be promoted to the primary instance.

Example

The following operation forces a failover of the sample-cluster cluster. It does not specify which instance to make the new primary instance, so Amazon DocumentDB chooses the instance according to failover tier priority.

For Linux, macOS, or Unix:

```
aws docdb failover-db-cluster \
--db-cluster-identifier sample-cluster
```

For Windows:

```
aws docdb failover-db-cluster ^
--db-cluster-identifier sample-cluster
```

The following operation forces a failover of the sample-cluster cluster, specifying that sample-cluster-instance is to be promoted to the primary role. (Notice "IsClusterWriter": true in the output.)

For Linux, macOS, or Unix:

```
aws docdb failover-db-cluster \
--db-cluster-identifier sample-cluster \
--target-db-instance-identifier sample-cluster-instance
```

For Windows:

```
aws docdb failover-db-cluster ^
--db-cluster-identifier sample-cluster ^
--target-db-instance-identifier sample-cluster-instance
```

Output from this operation looks something like the following (JSON format).

```
{
    "DBCluster": {
        "HostedZoneId": "Z2SUY0A1719RZT",
        "Port": 27017,
        "EngineVersion": "3.6.0",
        "PreferredMaintenanceWindow": "thu:04:05-thu:04:35",
        "BackupRetentionPeriod": 1,
        "ClusterCreateTime": "2018-06-28T18:53:29.455Z",
        "AssociatedRoles": [],
        "DBSubnetGroup": "default",
        "MasterUsername": "master-user",
        "Engine": "docdb",
        "ReadReplicaIdentifiers": [],
        "EarliestRestorableTime": "2018-08-21T00:04:10.546Z",
        "DBClusterIdentifier": "sample-cluster",
        "ReaderEndpoint": "sample-cluster.node.us-east-1.docdb.amazonaws.com",
        "DBClusterMembers": [
            {
                "DBInstanceIdentifier": "sample-cluster-instance",
```

```

        "DBClusterParameterGroupStatus": "in-sync",
        "PromotionTier": 1,
        "IsClusterWriter": true
    },
    {
        "DBInstanceIdentifier": "sample-cluster-instance-00",
        "DBClusterParameterGroupStatus": "in-sync",
        "PromotionTier": 1,
        "IsClusterWriter": false
    },
    {
        "DBInstanceIdentifier": "sample-cluster-instance-01",
        "DBClusterParameterGroupStatus": "in-sync",
        "PromotionTier": 1,
        "IsClusterWriter": false
    }
],
"AvailabilityZones": [
    "us-east-1b",
    "us-east-1c",
    "us-east-1a"
],
"DBClusterParameterGroup": "default.docdb3.6",
"Endpoint": "sample-cluster.node.us-east-1.docdb.amazonaws.com",
"IAMDatabaseAuthenticationEnabled": false,
"AllocatedStorage": 1,
"LatestRestorableTime": "2018-08-22T21:57:33.904Z",
"PreferredBackupWindow": "00:00-00:30",
"StorageEncrypted": false,
"MultiAZ": true,
"Status": "available",
"DBClusterArn": "arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster",
"VpcSecurityGroups": [
    {
        "Status": "active",
        "VpcSecurityGroupId": "sg-12345678"
    }
],
"DbClusterResourceId": "cluster-ABCDEFGHIJKLMNOPQRSTUVWXYZ"
}
}

```

Replication Lag

Replication lag is typically 50ms or less. The most common reasons for increased replica lag are:

- A high write rate on the primary that causes the read replicas to fall behind the primary.
- Contention on the read replicas between long running queries (e.g., large sequential scans, aggregation queries) and incoming write replication.
- Very large number of concurrent queries on the read replicas.

To minimize replication lag, try these troubleshooting techniques:

- If you have a high write rate or high CPU utilization, we recommend that you scale up the instances in your cluster.
- If there are long running queries on your read replicas, and very frequent updates to the documents being queried, consider altering your long running queries, or running them against the primary/write replica to avoid contention on the read replicas.
- If there is a very large number of concurrent queries or high CPU utilization only on the read replicas, another option is to scale out the number of read replicas to spread out the workload.

- Because replication lag is a result of high write throughput and long running queries, we recommend troubleshooting the replication lag by utilizing the `DBClusterReplicaLagMaximum CW` metric in combination with the slow query logger and `WriteThroughput/WriteIOPS` metrics.

In general, we recommend that all your replicas are of the same instance type, so that a cluster failover will not cause a degradation in performance.

If you are choosing between scaling up and scaling out (eg. six smaller instances vs three larger instances), we generally recommend trying to scale up first (larger instances) before scaling out, as you will get a larger buffer cache per DB instance.

Proactively, you should set a replication lag alarm and set its threshold to a value that you feel is the upper bound for how far behind (or "stale") your data on replica instances can be before it starts affecting the functionality of your application. In general, we would advise that the replication lag threshold be exceeded for several data points before alarming, due to transient workloads.

Note

In addition, we recommend that you set another alarm for replication lags that exceed 10 seconds. If you surpass this threshold for multiple data points, we recommend that you scale up your instances or reduce your write throughput on the primary instance.

Managing Amazon DocumentDB Events

Amazon DocumentDB (with MongoDB compatibility) keeps a record of events that relate to your clusters, instances, snapshots, security groups, and cluster parameter groups. This information includes the date and time of the event, the source name and source type of the event, and a message that is associated with the event.

Important

For certain management features, Amazon DocumentDB uses operational technology that is shared with Amazon RDS and Amazon Neptune. Region limits, limits that are governed at the Region level, are shared between Amazon DocumentDB, Amazon RDS, and Amazon Neptune. For more information, see [Regional Quotas \(p. 536\)](#).

Topics

- [Viewing Amazon DocumentDB Event Categories \(p. 349\)](#)
- [Viewing Amazon DocumentDB Events \(p. 351\)](#)

Viewing Amazon DocumentDB Event Categories

Each Amazon DocumentDB resource type has specific types of events that can be associated with it. You can use the AWS CLI `describe-event-categories` operation to view the mapping between event types and Amazon DocumentDB resource types.

Parameters

- **--source-type**—Optional. Use the `--source-type` parameter to see the event categories for a particular source type. The following are permitted values:
 - db-cluster
 - db-instance
 - db-parameter-group
 - db-security-group
 - db-cluster-snapshot

- **--filters**—Optional. To view the event categories for just Amazon DocumentDB, use the filter --filter Name=engine,Values=docdb.

Example

The following code lists the event categories associated with clusters.

For Linux, macOS, or Unix:

```
aws docdb describe-event-categories \
  --filter Name=engine,Values=docdb \
  --source-type db-cluster
```

For Windows:

```
aws docdb describe-event-categories ^
  --filter Name=engine,Values=docdb ^
  --source-type db-cluster
```

Output from this operation looks something like the following (JSON format).

```
{
  "EventCategoriesMapList": [
    {
      "EventCategories": [
        "notification",
        "failure",
        "maintenance",
        "failover"
      ],
      "SourceType": "db-cluster"
    }
  ]
}
```

The following code lists the event categories that are associated with each Amazon DocumentDB source type.

```
aws docdb describe-event-categories
```

Output from this operation looks something like the following (JSON format).

```
{
  "EventCategoriesMapList": [
    {
      "SourceType": "db-instance",
      "EventCategories": [
        "notification",
        "failure",
        "creation",
        "maintenance",
        "deletion",
        "recovery",
        "restoration",
        "configuration change",
        "read replica",
        "backtrack",
        "low storage",
        "modification"
      ]
    }
  ]
}
```

```
        "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"
    ]
}
]
```

Viewing Amazon DocumentDB Events

You can retrieve events for your Amazon DocumentDB resources through the Amazon DocumentDB console, which shows events from the past 24 hours. You can also retrieve events for your Amazon DocumentDB resources by using the [describe-events](#) AWS CLI command, or the [DescribeEvents](#) Amazon DocumentDB API operation. If you use the AWS CLI or the Amazon DocumentDB API to view events, you can retrieve events for up to the past 14 days.

Topics

- [Using the AWS Management Console \(p. 351\)](#)
- [Using the AWS CLI \(p. 352\)](#)

Using the AWS Management Console

To view all Amazon DocumentDB instance events for the past 24 hours

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Events**. The available events appear in a list.
3. Use the **Filter** list to filter the events by type. Enter a term in the text box to further filter your results. For example, the following screenshot shows filtering all Amazon DocumentDB events for *snapshot* events.

Events (10)			Message
Source	Type	Time	Message
docdb-2018-10-23-21-08-23-final-snapshot	db-cluster-snapshot	Tue Oct 23 2018	Manual cluster snapshot created
docdb-2018-10-23-21-08-23-final-snapshot	db-cluster-snapshot	Tue Oct 23 2018	Creating manual cluster snapshot

Using the AWS CLI

To view all Amazon DocumentDB instance events for the past 7 days

You can view all Amazon DocumentDB instance events for the past 7 days by running the [describe-events](#) AWS CLI operation with the --duration parameter set to 10080 (10,080 minutes).

```
aws docdb describe-events --duration 10080
```

Filtering for Amazon DocumentDB Events

To see specific Amazon DocumentDB events, use the `describe-events` operation with the following parameters.

Parameters

- **--filter**—Required to limit returned values to Amazon DocumentDB events. Use `Name=engine,Values=docdb` to filter all events for Amazon DocumentDB only.
- **--source-identifier**—Optional. The identifier of the event source for which events are returned. If omitted, events from all sources are included in the results.
- **--source-type**—Optional, unless `--source-identifier` is provided, then required. If `--source-identifier` is provided, `--source-type` must agree with the type of the `--source-identifier`. The following are permitted values:
 - db-cluster
 - db-instance
 - db-parameter-group
 - db-security-group
 - db-cluster-snapshot

The following example lists all your Amazon DocumentDB events.

```
aws docdb describe-events --filters Name=engine,Values=docdb
```

Output from this operation looks something like the following (JSON format).

```
{  
  "Events": [  
    {  
      "SourceArn": "arn:aws:rds:us-east-1:123SAMPLE012:db:sample-cluster-instance3",  
      "Message": "instance created",  
      "SourceType": "db-instance",  
      "Date": "2018-12-11T21:17:40.023Z",  
      "SourceIdentifier": "sample-cluster-instance3",  
      "EventCategories": [  
        "creation"  
      ]  
    }  
  ]  
}
```

```

        ],
        {
            "SourceArn": "arn:aws:rds:us-east-1:123SAMPLE012:db:docdb-2018-12-11-21-08-23",
            "Message": "instance shutdown",
            "SourceType": "db-instance",
            "Date": "2018-12-11T21:25:01.245Z",
            "SourceIdentifier": "docdb-2018-12-11-21-08-23",
            "EventCategories": [
                "availability"
            ]
        },
        {
            "SourceArn": "arn:aws:rds:us-east-1:123SAMPLE012:db:docdb-2018-12-11-21-08-23",
            "Message": "instance restarted",
            "SourceType": "db-instance",
            "Date": "2018-12-11T21:25:11.441Z",
            "SourceIdentifier": "docdb-2018-12-11-21-08-23",
            "EventCategories": [
                "availability"
            ]
        }
    ]
}

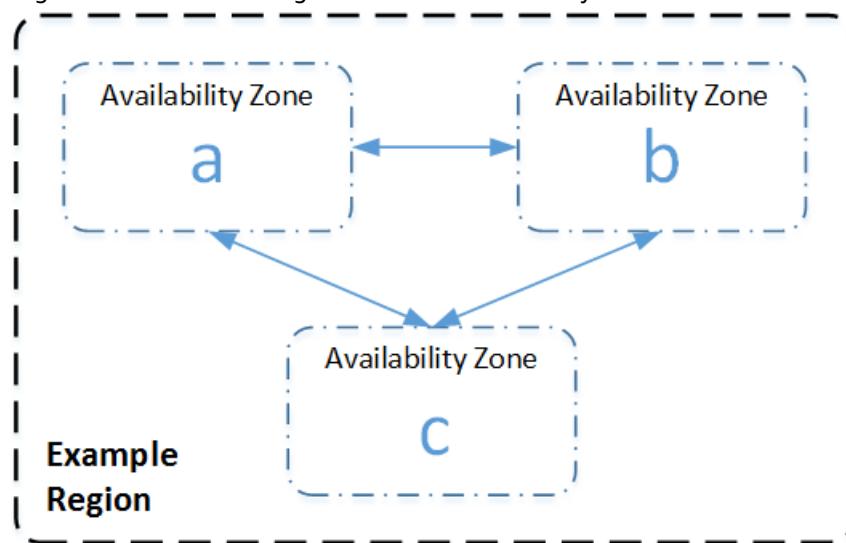
```

For more information, see [Auditing Amazon DocumentDB Events \(p. 205\)](#).

Choosing Regions and Availability Zones

Amazon cloud computing resources are hosted in multiple locations worldwide. These locations consist of AWS Regions and Availability Zones. Each *AWS Region* is a separate geographic area. Each Region has multiple, isolated locations known as *Availability Zones*. Amazon DocumentDB provides you the ability to place resources, such as instances, and data in multiple locations. Resources aren't replicated across AWS Regions unless you do so specifically.

Amazon operates advanced, highly available data centers. Although rare, failures can occur that affect the availability of instances that are in the same location. If you host all your instances in a single location that is affected by such a failure, none of your instances would be available. The following diagram shows an AWS Region with three Availability Zones.



It is important to remember that each Region is independent. Any Amazon DocumentDB activity that you initiate (for example, creating instances or listing available instances) runs only in your current default AWS Region. You can change the default Region on the console by setting the EC2_REGION environment variable. Or you can override it by using the --region parameter in the AWS CLI. For more information, see [Configuring the AWS Command Line Interface](#), specifically, the sections on environment variables and command line options.

When you create a cluster using the Amazon DocumentDB console, and you choose to create a replica in a different Availability Zone, Amazon DocumentDB creates two instances. It creates the primary instance in one Availability Zone and the replica instance in a different Availability Zone. The cluster volume is always replicated across three Availability Zones.

To create or work with an Amazon DocumentDB instance in a specific AWS Region, use the corresponding regional service endpoint.

Region Availability

Amazon DocumentDB is available in the following AWS Regions.

Regions Supported by Amazon DocumentDB

Region Name	Region	Availability Zones (compute)
US East (Ohio)	us-east-2	3
US East (N. Virginia)	us-east-1	6
US West (Oregon)	us-west-2	4
South America (São Paulo)	sa-east-1	3
Asia Pacific (Mumbai)	ap-south-1	3
Asia Pacific (Seoul)	ap-northeast-2	4
Asia Pacific (Singapore)	ap-southeast-1	3
Asia Pacific (Sydney)	ap-southeast-2	3
Asia Pacific (Tokyo)	ap-northeast-1	3
Canada (Central)	ca-central-1	3
China (Beijing) Region	cn-north-1	3
China (Ningxia)	cn-northwest-1	3
Europe (Frankfurt)	eu-central-1	3
Europe (Ireland)	eu-west-1	3
Europe (London)	eu-west-2	3
Europe (Milan)	eu-south-1	3
Europe (Paris)	eu-west-3	3
AWS GovCloud (US)	us-gov-west-1	3

By default, the time zone for an Amazon DocumentDB cluster is Universal Time Coordinated (UTC).

For information on finding the connection endpoints for clusters and instances in a particular region, see [Understanding Amazon DocumentDB Endpoints \(p. 378\)](#).

Managing Amazon DocumentDB Cluster Parameter Groups

You can manage Amazon DocumentDB engine configuration by using parameters in a cluster parameter group. A *cluster parameter group* is a collection of Amazon DocumentDB configuration values that make it easier to manage the parameters of your Amazon DocumentDB clusters. Cluster parameter groups act as a container for engine configuration values that are applied to all instances in the cluster.

This section describes how to create, view, and modify cluster parameter groups. It also shows how you can determine which cluster parameter group is associated with a given cluster.

Topics

- [Describing Amazon DocumentDB Cluster Parameter Groups \(p. 355\)](#)
- [Creating Amazon DocumentDB Cluster Parameter Groups \(p. 360\)](#)
- [Modifying Amazon DocumentDB Cluster Parameter Groups \(p. 362\)](#)
- [Modifying Amazon DocumentDB Clusters to Use Customized Cluster Parameter Groups \(p. 365\)](#)
- [Copying Amazon DocumentDB Cluster Parameter Groups \(p. 366\)](#)
- [Resetting Amazon DocumentDB Cluster Parameter Groups \(p. 367\)](#)
- [Deleting Amazon DocumentDB Cluster Parameter Groups \(p. 369\)](#)
- [Amazon DocumentDB Cluster Parameters Reference \(p. 371\)](#)

Describing Amazon DocumentDB Cluster Parameter Groups

When you create an Amazon DocumentDB cluster, a default `.docdb4.0` cluster parameter group is automatically created for that cluster. You can view the details of this default cluster parameter group or any additional cluster parameter groups that you have. You can also determine which parameter group is associated with a particular cluster.

Topics

- [Describing the Details of an Amazon DocumentDB Cluster Parameter Group \(p. 355\)](#)
- [Determining an Amazon DocumentDB Cluster's Parameter Group \(p. 358\)](#)

Describing the Details of an Amazon DocumentDB Cluster Parameter Group

To describe the details of a given cluster parameter group, complete the following steps using the AWS Management Console or the AWS Command Line Interface (AWS CLI).

Using the AWS Management Console

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Parameter groups**.

Tip

If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.

3. In the **Cluster parameter groups** pane, select the name of the parameter group that you want to see the details of.
4. The resulting page shows the parameter group's parameters, recent activity, and tags.
 - Under **Cluster parameters**, you can see the parameter's name, current value, allowed values, whether the parameter is modifiable, its apply type, data type, and description. You can modify individual parameters by selecting the parameter and then choosing **Edit** in the **Cluster parameters** section. For more information, see [Modifying Amazon DocumentDB Cluster Parameters \(p. 374\)](#).
 - Under **Recent events**, you can see the most recent events for this parameter group. You can filter through these events using the search bar in this section. For more information, see [Managing Amazon DocumentDB Events \(p. 349\)](#).
 - Under **Tags**, you can see the tags that are on this cluster parameter group. You can add or remove tags by choosing **Edit** in the **Tags** section. For more information, see [Tagging Amazon DocumentDB Resources \(p. 387\)](#).

Using the AWS CLI

You can use the `describe-db-cluster-parameter-groups` AWS CLI command to view the Amazon Resource Name (ARN), family, description, and name of a single cluster parameter group or all cluster parameter groups that you have for Amazon DocumentDB. You can also use the `describe-db-cluster-parameters` AWS CLI command to view the parameters and their details inside a single cluster parameter group.

- **--describe-db-cluster-parameter-groups** — To see a listing of all your cluster parameter groups and their details.
 - **--db-cluster-parameter-group-name** — Optional. The name of the cluster parameter group that you want described. If this parameter is omitted, all cluster parameter groups are described.
 - **--describe-db-cluster-parameters** — To list all the parameters inside a parameter group and their values.
 - **--db-cluster-parameter-group name** — Required. The name of the cluster parameter group that you want described.

Example

The following code lists up to 100 cluster parameter groups and their ARN, family, description, and name.

```
aws docdb describe-db-cluster-parameter-groups
```

Output from this operation looks something like the following (JSON format).

```
{  
    "DBClusterParameterGroups": [  
        {
```

```
        "DBClusterParameterGroupArn": "arn:aws:rds:us-  
east-1:012345678912:cluster-pg:default.docdb4.0",  
        "DBParameterGroupFamily": "docdb4.0",  
        "Description": "Default cluster parameter group for docdb4.0",  
        "DBClusterParameterGroupName": "default.docdb4.0"  
    },  
    {  
        "DBClusterParameterGroupArn": "arn:aws:rds:us-  
east-1:012345678912:cluster-pg:sample-parameter-group",  
        "DBParameterGroupFamily": "docdb4.0",  
        "Description": "Custom docdb4.0 parameter group",  
        "DBClusterParameterGroupName": "sample-parameter-group"  
    }  
}  
]
```

Example

The following code lists the ARN, family, description, and name for sample-parameter-group.

For Linux, macOS, or Unix:

```
aws docdb describe-db-cluster-parameter-groups \  
--db-cluster-parameter-group-name sample-parameter-group
```

For Windows:

```
aws docdb describe-db-cluster-parameter-groups ^  
--db-cluster-parameter-group-name sample-parameter-group
```

Output from this operation looks something like the following (JSON format).

```
{  
    "DBClusterParameterGroups": [  
        {  
            "DBClusterParameterGroupArn": "arn:aws:rds:us-  
east-1:123456789012:cluster-pg:sample-parameter-group",  
            "Description": "Custom docdb4.0 parameter group",  
            "DBParameterGroupFamily": "docdb4.0",  
            "DBClusterParameterGroupName": "sample-parameter-group"  
        }  
    ]  
}
```

Example

The following code lists the values of the parameters in sample-parameter-group.

For Linux, macOS, or Unix:

```
aws docdb describe-db-cluster-parameters \  
--db-cluster-parameter-group-name sample-parameter-group
```

For Windows:

```
aws docdb describe-db-cluster-parameters ^
--db-cluster-parameter-group-name sample-parameter-group
```

Output from this operation looks something like the following (JSON format).

```
{
  "Parameters": [
    {
      "ParameterName": "audit_logs",
      "ParameterValue": "disabled",
      "Description": "Enables auditing on cluster.",
      "Source": "system",
      "ApplyType": "dynamic",
      "DataType": "string",
      "AllowedValues": "enabled,disabled",
      "IsModifiable": true,
      "ApplyMethod": "pending-reboot"
    },
    {
      "ParameterName": "change_stream_log_retention_duration",
      "ParameterValue": "17777",
      "Description": "Duration of time in seconds that the change stream log is retained and can be consumed.",
      "Source": "user",
      "ApplyType": "dynamic",
      "DataType": "integer",
      "AllowedValues": "3600-86400",
      "IsModifiable": true,
      "ApplyMethod": "pending-reboot"
    }
  ]
}
```

Determining an Amazon DocumentDB Cluster's Parameter Group

To determine which parameter group is associated with a particular cluster, complete the following steps using the AWS Management Console or the AWS CLI.

Using the AWS Management Console

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the left navigation pane, choose **Clusters**.
3. In the list of clusters, select the name of the cluster you are interested in.
4. The resulting page shows the details of the cluster that you selected. Scroll down to **Cluster details**. At the bottom of that section, locate the parameter group's name below **Cluster parameter group**.

Cluster details

Configurations and status

ARN

arn:aws:rds: [REDACTED]:cluster:sample-cluster

Cluster identifier

sample-cluster (available)

Cluster creation time

1/10/2020, 2:13:38 PM UTC-8

Cluster endpoint

sample-cluster.
[REDACTED].docdb.amazonaws.com

Reader endpoint

sample-cluster.
[REDACTED].docdb.amazonaws.com

Master username

Port

27017

Status

available

Cluster parameter group

359

sample-parameter-group

Using the AWS CLI

The following AWS CLI code determines which parameter group is governing the cluster sample-cluster.

```
aws docdb describe-db-clusters \
    --db-cluster-identifier sample-cluster \
    --query 'DBClusters[*].[DBClusterIdentifier,DBClusterParameterGroup]'
```

Output from this operation looks something like the following (JSON format).

```
[  
  [  
    "sample-cluster",  
    "sample-parameter-group"  
  ]  
]
```

Creating Amazon DocumentDB Cluster Parameter Groups

Amazon DocumentDB provides 3.6 and 4.0 versioned cluster parameter groups. You can create a default cluster parameter group based on the docdb4.0 family named default.docdb4.0. In either version, you cannot modify the default.docdb4.0 or default.docdb3.6 cluster parameter group directly. To customize the parameters within a cluster parameter group and associate it to your Amazon DocumentDB cluster, you must first create a new, non-default cluster parameter group. Then you can modify the parameters within that new, custom parameter group and associate it to your Amazon DocumentDB cluster.

The following procedure guides you through creating a custom cluster parameter group based on the docdb4.0 family. You can then [modify the parameters within that parameter group](#).

Note

After you create a cluster parameter group, you should wait at least 5 minutes before using that particular parameter group. This allows Amazon DocumentDB to fully complete the create action before the cluster parameter group is used for a new cluster. You can use the AWS Management Console or the describe-db-cluster-parameter-groups AWS CLI operation to verify that your cluster parameter group has been created. For more information, see [Describing Amazon DocumentDB Cluster Parameter Groups \(p. 355\)](#).

Using the AWS Management Console

To create a cluster parameter group

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Parameter groups**.

Tip

If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.

3. In the **Cluster parameter groups** pane, choose **Create**.
4. In the **Create cluster parameter group** pane, enter the following:

- a. **Group name** — Enter a name for the cluster parameter group. For example, `sample-parameter-group`. Cluster parameter groups have the following naming constraints:
 - Length is [1–255] alphanumeric characters.
 - First character must be a letter.
 - Cannot end with a hyphen or contain two consecutive hyphens.
 - b. **Description** — Provide a description for this cluster parameter group.
5. To create the cluster parameter group, choose **Create**. To cancel the operation, choose **Cancel**.
6. After you choose **Create**, the following text appears at the top of the page to verify that your cluster parameter group has been successfully created:

Successfully created cluster parameter group '`sample-parameter-group`'.

Using the AWS CLI

To create a new cluster parameter group for Amazon DocumentDB 4.0 clusters, use the AWS CLI `create-db-cluster-parameter-group` operation with the following parameters:

- **--db-cluster-parameter-group-name** — The name of the custom cluster parameter group. For example, `sample-parameter-group`.
- **--db-cluster-parameter-group-family** — The cluster parameter group family that is used as the template for the custom cluster parameter group. Currently, this must be `docdb4.0`.
- **--description** — The user-provided description for this cluster parameter group. The following example uses "Custom docdb4.0 parameter group".

For Linux, macOS, or Unix:

Example

```
aws docdb create-db-cluster-parameter-group \
--db-cluster-parameter-group-name sample-parameter-group \
--db-parameter-group-family docdb4.0 \
--description "Custom docdb4.0 parameter group"
```

For Windows:

```
aws docdb create-db-cluster-parameter-group ^
--db-cluster-parameter-group-name sample-parameter-group ^
--db-parameter-group-family docdb4.0 ^
--description "Custom docdb4.0 parameter group"
```

Output from this operation looks something like the following (JSON format).

```
{  
    "DBClusterParameterGroup": {  
        "DBClusterParameterGroupName": "sample-parameter-group",  
        "DBParameterGroupFamily": "docdb4.0",  
        "Description": "Custom docdb4.0 parameter group",  
        "DBClusterParameterGroupArn": "sample-parameter-group-arn"  
    }  
}
```

Modifying Amazon DocumentDB Cluster Parameter Groups

This section explains how to modify a *custom* Amazon DocumentDB parameter group. In Amazon DocumentDB, you cannot make modifications directly to the default.docdb4.0 cluster parameter group. If your Amazon DocumentDB cluster is using the default cluster parameter group and you want to modify a value in it, you must first [create a new parameter group](#) or [copy an existing parameter group](#), modify it, and then apply the modified parameter group to your cluster.

Complete the following steps to modify a non-default cluster parameter group. After you modify a cluster parameter group, you should wait at least 5 minutes before using it. This allows Amazon DocumentDB to fully complete the modify action before the cluster parameter group is used. You can use the AWS Management Console or the AWS CLI `describe-db-cluster-parameters` operation to verify that your cluster parameter group has been modified. For more information, see [Describing Cluster Parameter Groups \(p. 355\)](#).

Using the AWS Management Console

Follow these steps to modify a custom Amazon DocumentDB parameter group. You can't modify a default parameter group. If you want to modify a value in the default parameter group, you can [copy the default cluster parameter group](#), modify it, and then apply the modified parameter group to your cluster. For more information about applying parameter groups to your cluster, see [Modifying an Amazon DocumentDB Cluster \(p. 287\)](#).

To modify a custom cluster parameter group

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane on the left side of the console, choose **Parameter groups**. In the list of parameter groups, choose the name of the parameter group that you want to modify.

Tip

If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.

3. For each parameter in the parameter group that you want to modify, do the following:
 - a. Locate the parameter that you want to modify, and verify that it is modifiable by checking if it is listed as `true` under the **Modifiable** column.
 - b. If it is modifiable, select the parameter and choose **Edit** from the top right of the console page.
 - c. In the **Modify <parameter-name>** dialog box, make the changes that you want. Then choose **Modify cluster parameter**, or choose **Cancel** to discard the changes.

Using the AWS CLI

You can modify the `ParameterValue`, `Description`, or `ApplyMethod` of any modifiable parameter in a custom Amazon DocumentDB cluster parameter group using the AWS CLI. You can't make modifications directly to a default cluster parameter group.

To modify a custom cluster parameter group's parameters, use the `modify-db-cluster-parameter-group` operation with the following parameters.

- **--db-cluster-parameter-group-name** — Required. The name of the cluster parameter group that you are modifying.

- **--parameters** — Required. The parameters that you are modifying. For a list of the parameters that apply to all instances in an Amazon DocumentDB cluster, see the [Amazon DocumentDB Cluster Parameters Reference \(p. 371\)](#). Each parameter entry must include the following:
 - **ParameterName** — The name of the parameter that you are modifying.
 - **ParameterValue** — The new value for this parameter.
 - **ApplyMethod** — How you want changes to this parameter applied. Permitted values are immediate and pending-reboot.

Note

Parameters with the ApplyType of static must have an ApplyMethod of pending-reboot.

Example - Modifying a parameter's value

In this example, you list the parameter values of sample-parameter-group and modify the tls parameter. Then, after waiting 5 minutes, you again list the parameter values of sample-parameter-group to see the changed parameter values.

1. List the parameters and their values of sample-parameter-group.

For Linux, macOS, or Unix:

```
aws docdb describe-db-cluster-parameters \
    --db-cluster-parameter-group-name sample-parameter-group
```

For Windows:

```
aws docdb describe-db-cluster-parameters ^
    --db-cluster-parameter-group-name sample-parameter-group
```

Output from this operation looks something like the following (JSON format).

```
{
    "Parameters": [
        {
            "Source": "system",
            "ApplyType": "static",
            "AllowedValues": "disabled,enabled",
            "ParameterValue": "enabled",
            "ApplyMethod": "pending-reboot",
            "DataType": "string",
            "ParameterName": "tls",
            "IsModifiable": true,
            "Description": "Config to enable/disable TLS"
        },
        {
            "Source": "user",
            "ApplyType": "dynamic",
            "AllowedValues": "disabled,enabled",
            "ParameterValue": "enabled",
            "ApplyMethod": "pending-reboot",
            "DataType": "string",
            "ParameterName": "ttl_monitor",
            "IsModifiable": true,
            "Description": "Enables TTL Monitoring"
        }
    ]
}
```

2. Modify the `tls` parameter so that its value is disabled.

You can't modify the `ApplyMethod` because the `ApplyType` is `static`.

For Linux, macOS, or Unix:

```
aws docdb modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name sample-parameter-group \
    --parameters
    "ParameterName"=tls,"ParameterValue"=disabled,"ApplyMethod"=pending-reboot
```

For 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 from this operation looks something like the following (JSON format).

```
{
    "DBClusterParameterGroupName": "sample-parameter-group"
}
```

3. Wait at least 5 minutes.
4. List the parameter values of `sample-parameter-group` to verify that the `tls` parameter was modified.

For Linux, macOS, or Unix:

```
aws docdb describe-db-cluster-parameters \
    --db-cluster-parameter-group-name sample-parameter-group
```

For Windows:

```
aws docdb describe-db-cluster-parameters ^
    --db-cluster-parameter-group-name sample-parameter-group
```

Output from this operation looks something like the following (JSON format).

```
{
    "Parameters": [
        {
            "ParameterValue": "false",
            "ParameterName": "enable_audit_logs",
            "ApplyType": "dynamic",
            "DataType": "string",
            "Description": "Enables auditing on cluster.",
            "AllowedValues": "true,false",
            "Source": "system",
            "IsModifiable": true,
            "ApplyMethod": "pending-reboot"
        },
        {
            "ParameterValue": "disabled",
            "ParameterName": "tls",
            "ApplyType": "static",
            "DataType": "string",
            "Description": "TLS encryption for connections to the cluster."
        }
    ]
}
```

```
        "Description": "Config to enable/disable TLS",
        "AllowedValues": "disabled,enabled",
        "Source": "system",
        "IsModifiable": true,
        "ApplyMethod": "pending-reboot"
    }
}
```

Modifying Amazon DocumentDB Clusters to Use Customized Cluster Parameter Groups

When you create an Amazon DocumentDB cluster, a default .docdb4.0 parameter group is automatically created for that cluster. You can't modify the default cluster parameter group. Instead, you can modify your Amazon DocumentDB cluster to associate a new customized parameter group with it.

This section explains how to modify an existing Amazon DocumentDB cluster to use a custom cluster parameter group using the AWS Management Console and the AWS Command Line Interface (AWS CLI).

Using the AWS Management Console

To modify an Amazon DocumentDB cluster to use a new, non-default cluster parameter group

1. Before you begin, make sure you have created an Amazon DocumentDB cluster and a cluster parameter group. See [Creating an Amazon DocumentDB Cluster \(p. 274\)](#) and [Creating Amazon DocumentDB Cluster Parameter Groups \(p. 360\)](#) for further instructions.
2. After creating your cluster parameter group, open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>. In the navigation pane, choose **Clusters** to add your new parameter group to a cluster.
3. Choose the cluster that you want to associate your parameter group with. Choose **Actions**, and then choose **Modify** to modify your cluster.
4. Under **Cluster options**, choose the new parameter group that you want to associate your cluster with.
5. Choose **Continue** to view a summary of your modifications.
6. After verifying your changes, you can apply them immediately or during the next maintenance window under **Scheduling of modifications**.
7. Choose **Modify cluster** to update your cluster with your new parameter group.

Using the AWS CLI

Before you begin, make sure that you have created an Amazon DocumentDB cluster and a cluster parameter group. You can [create an Amazon DocumentDB cluster](#) using the AWS CLI `create-db-cluster` operation. You can [create a cluster parameter group](#) using the AWS CLI `create-db-cluster-parameter-group` operation.

To add your new cluster parameter group to your cluster, use the AWS CLI `modify-db-cluster` operation with the following parameters.

- **--db-cluster-identifier** — The name of your cluster (for example, `sample-cluster`).
- **--db-cluster-parameter-group-name** — The name of the parameter group that you want to associate your cluster with (for example, `sample-parameter-group`).

Example

```
aws docdb modify-db-cluster \
--db-cluster-identifier sample-cluster
--db-cluster-parameter-group-name sample-parameter-group
```

Output from this operation looks something like the following (JSON format).

```
"DBCluster": {
    "AvailabilityZones": [
        "us-west-2c",
        "us-west-2b",
        "us-west-2a"
    ],
    "BackupRetentionPeriod": 1,
    "DBClusterIdentifier": "sample-cluster",
    "DBClusterParameterGroup": "sample-parameter-group",
    "DBSubnetGroup": "default",
    ...
}
```

Copying Amazon DocumentDB Cluster Parameter Groups

You can make a copy of a cluster parameter group in Amazon DocumentDB using the AWS Management Console or the AWS Command Line Interface (AWS CLI).

Using the AWS Management Console

The following procedure guides you through making a new cluster parameter group by making a copy of an existing cluster parameter group.

To copy a cluster parameter group

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Parameter groups**.
3. In the **Cluster parameter groups** pane, choose the name of the cluster parameter group that you want to copy.
4. Choose **Actions**, and then choose **Copy** to copy that parameter group.
5. Under **Copy options**, enter a name and description for the new cluster parameter group. Then choose **Copy** to save your changes.

Using the AWS CLI

To make a copy of a cluster parameter group, use the `copy-db-cluster-parameter-group` operation with the following parameters.

- **--source-db-cluster-parameter-group-identifier** — Required. The name or Amazon Resource Name (ARN) of the cluster parameter group that you want to make a copy of.

If the source and target cluster parameter groups are in the same AWS Region, the identifier can be either a name or an ARN.

If the source and target cluster parameter groups are in different AWS Regions, the identifier must be an ARN.

- **--target-db-cluster-parameter-group-identifier** — Required. The name or ARN of the cluster parameter group copy.

Constraints:

- Cannot be null, empty, or blank.
- Must contain 1–255 letters, numbers, or hyphens.
- First character must be a letter.
- Cannot end with a hyphen or contain two consecutive hyphens.
- **--target-db-cluster-parameter-group-description** — Required. A user-supplied description for the cluster parameter group copy.

Example

The following code makes a copy of sample-parameter-group, naming the copy sample-parameter-group-copy.

For Linux, macOS, or 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"
```

For 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 from this operation looks something like the following (JSON format).

```
{
  "DBClusterParameterGroup": {
    "DBClusterParameterGroupArn": "arn:aws:rds:us-east-1:123456789012:cluster-
pg:sample-parameter-group-copy",
    "DBClusterParameterGroupName": "sample-parameter-group-copy",
    "DBParameterGroupFamily": "docdb4.0",
    "Description": "Copy of sample-parameter-group"
  }
}
```

Resetting Amazon DocumentDB Cluster Parameter Groups

You can reset some or all of an Amazon DocumentDB cluster parameter group's parameter values to their default values by using the AWS Management Console or the AWS Command Line Interface (AWS CLI) to reset the cluster parameter group.

Using the AWS Management Console

Follow these steps to reset some or all of a cluster parameter group's parameter values to their default values.

To reset a cluster parameter group's parameter values

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane on the left side of the console, choose **Parameter groups**.
3. In the **Cluster parameter groups** pane, choose the name of the cluster parameter group that you want to reset.
4. Choose **Actions**, and then choose **Reset** to reset that parameter group.
5. On the resulting **Cluster parameter group reset confirmation** page, confirm that you want to reset all cluster parameters for that parameter group to their defaults. Then choose **Reset** to reset your parameter group. You can also choose **Cancel** to discard your changes.

Using the AWS CLI

To reset some or all of a cluster parameter group's parameter values to their default values, use the `reset-db-cluster-parameter-group` operation with the following parameters.

- **--db-cluster-parameter-group-name** — Required. The name of the cluster parameter group to reset.
- **--parameters** — Optional. A list of `ParameterName` and `ApplyMethod` in the cluster parameter group to reset to their default values. Static parameters must be set to `pending-reboot` to take effect on the next instance restart or `reboot-db-instance` request. You must call `reboot-db-instance` for every instance in your cluster that you want the updated static parameter to apply to.

This parameter and `--reset-all-parameters` are mutually exclusive: you can use either one but not both.

- **--reset-all-parameters** or **--no-reset-all-parameters** — Optional. Specifies whether to reset all parameters (`--reset-all-parameters` or only some of the parameters (`--no-reset-all-parameters`) to their default values. The `--reset-all-parameters` parameter and `--parameters` are mutually exclusive: you can use either one but not both.

When you reset the entire group, dynamic parameters are updated immediately. Static parameters are set to `pending-reboot` to take effect on the next instance restart or `reboot-db-instance` request. You must call `reboot-db-instance` for every instance in your cluster that you want the updated static parameter applied to.

Example

Example 1: Resetting all parameters to their default values

The following code resets all parameters in the cluster parameter group `sample-parameter-group` to their default values.

For Linux, macOS, or Unix:

```
aws docdb reset-db-cluster-parameter-group \
    --db-cluster-parameter-group-name sample-parameter-group \
    --reset-all-parameters
```

For Windows:

```
aws docdb reset-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name sample-parameter-group ^
    --reset-all-parameters
```

Example 2: Resetting specified parameters to their default values

The following code resets the `tls` parameter in the cluster parameter group `sample-parameter-group` to its default value.

For Linux, macOS, or 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
```

For 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 from this operation looks something like the following (JSON format).

```
{  
    "DBClusterParameterGroupName": "sample-parameter-group"  
}
```

Rebooting a cluster instance

Before a static parameter's value is changed, the cluster instance must be rebooted. Reboot each instance in your cluster that you want the updated static parameter to apply to.

For Linux, macOS, or Unix:

```
aws docdb reboot-db-instance \
    --db-instance-identifier sample-cluster-instance
```

For Windows:

```
aws docdb reboot-db-instance ^
    --db-instance-identifier sample-cluster-instance
```

Deleting Amazon DocumentDB Cluster Parameter Groups

You can delete a custom Amazon DocumentDB cluster parameter group using the AWS Management Console or the AWS Command Line Interface (AWS CLI). You can't delete the default `.docdb4.0` cluster parameter group.

Using the AWS Management Console

To delete a cluster parameter group

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Parameter groups**.

Tip

If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.

3. In the **Parameter groups** pane, choose the radio button to the left of the cluster parameter group that you want to delete.
4. Choose **Actions**, and then choose **Delete**.
5. In the **Delete** confirmation pane, choose **Delete** to delete the cluster parameter group. To keep the cluster parameter group, choose **Cancel**.

Using the AWS CLI

To delete a cluster parameter group, use the `delete-db-cluster-parameter-group` operation with the following parameter.

- **--db-cluster-parameter-group-name** — Required. The name of the cluster parameter group to delete. This must be an existing cluster parameter group. *You cannot delete the default.docdb4.0 cluster parameter group.*

Example - Deleting a cluster parameter group

The following example walks you through the three steps for deleting a cluster parameter group:

1. Finding the name of the cluster parameter group that you want to delete.
2. Deleting the specified cluster parameter group.
3. Verifying that the cluster parameter group was deleted.

1. Find the name of the cluster parameter group that you want to delete.

The following code lists the names of all cluster parameter groups.

For Linux, macOS, or Unix:

```
aws docdb describe-db-cluster-parameter-groups \
--query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

For Windows:

```
aws docdb describe-db-cluster-parameter-groups ^
--query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

The output of the preceding operation is a list the names of cluster parameter groups similar to the following (JSON format).

```
[  
  [  
    "default.docdb4.0"  
  ],  
  [  
    "sample-parameter-group"  
  ],  
  [  
    "sample-parameter-group-copy"  
  ]
```

]

2. Delete a specific cluster parameter group.

The following code deletes the cluster parameter group `sample-parameter-group-copy`.

For Linux, macOS, or Unix:

```
aws docdb delete-db-cluster-parameter-group \
--db-cluster-parameter-group-name sample-parameter-group-copy
```

For Windows:

```
aws docdb delete-db-cluster-parameter-group ^
--db-cluster-parameter-group-name sample-parameter-group-copy
```

There is no output from this operation.

3. Verify that the specified cluster parameter group was deleted.

The following code lists the names of all remaining cluster parameter groups.

For Linux, macOS, or Unix:

```
aws docdb describe-db-cluster-parameter-groups \
--query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

For Windows:

```
aws docdb describe-db-cluster-parameter-groups ^
--query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
```

The output of the preceding operation is a list of cluster parameter groups similar to the following (JSON format). The cluster parameter group that you just deleted should not be in the list.

Output from this operation looks something like the following (JSON format).

```
[  
  [  
    "default.docdb4.0"  
  ],  
  [  
    "sample-parameter-group"  
  ]  
]
```

Amazon DocumentDB Cluster Parameters Reference

When you change a dynamic parameter and save the cluster parameter group, the change is applied immediately regardless of the *Apply immediately* setting. When you change a static parameter and save the cluster parameter group, the parameter change takes effect after you manually reboot the instance. You can reboot an instance using the Amazon DocumentDB console or by explicitly calling `reboot-db-instance`.

The following table shows the parameters that apply to all instances in an Amazon DocumentDB cluster.

Amazon DocumentDB cluster-level parameters

Parameter	Default Value	Valid Values	Modifiable	Apply Type	Data Type	Description	
audit_logs	disabled	enabled, disabled	Yes	Dynamic	String	<p>Defines whether AWS CloudTrail audit logs are enabled.</p> <ul style="list-style-type: none"> • enabled—CloudTrail audit logs are enabled. • disabled—CloudTrail audit logs are disabled. 	
change_stream_log_reten	10800	100-86400	tion	Yes	Dynamic	Integer	Defines the duration of time (in seconds) that the change stream log is retained and can be consumed.
profiler	disabled	enabled, disabled	Yes	Dynamic	String	<p>Enables profiling for slow operations.</p> <ul style="list-style-type: none"> • enabled—operations that take longer than a customer-defined threshold value (e.g., 100ms) are logged to Amazon CloudWatch Logs. • disabled—slow operations are not logged to 	

Parameter	Default Value	Valid Values	Modifiable	Apply Type	Data Type	Description
						CloudWatch Logs.
profiler_sampling_rate	0.0-1.0		Yes	Dynamic	Float	Defines the sampling rate for logged operations.
profiler_threshold_ms	50-2147483646	50-2147483646	Yes	Dynamic	Integer	Defines the threshold for profiler. <ul style="list-style-type: none"> All operations greater than profiler_threshold_ms are logged to CloudWatch Logs.
tls	enabled	enabled, disabled	Yes	Static	String	Defines whether Transport Layer Security (TLS) connections are required. <ul style="list-style-type: none"> enabled— TLS connections are required to connect. disabled— TLS connections cannot be used to connect.

Parameter	Default Value	Valid Values	Modifiable	Apply Type	Data Type	Description
ttl_monitor	enabled	enabled, disabled	Yes	Dynamic	String	<p>Defines whether Time to Live (TTL) monitoring is enabled for the cluster.</p> <ul style="list-style-type: none"> • enabled—TTL monitoring is enabled. • disabled—TTL monitoring is disabled.

Modifying Amazon DocumentDB Cluster Parameters

In Amazon DocumentDB, *cluster parameter groups* consist of *parameters* that apply to all of the instances that you create in the cluster. For custom cluster parameter groups, you can modify a parameter value at any time or reset all the parameter values to their defaults for parameter groups that you create. This section describes how to view the parameters that make up an Amazon DocumentDB cluster parameter group and their values, and how you can change or update these values.

Parameters can be *dynamic* or *static*. When you change a dynamic parameter and save the cluster parameter group, the change is applied immediately regardless of the *Apply Immediately* setting. When you change a static parameter and save the cluster parameter group, the parameter change takes effect only after you manually reboot the instances.

Viewing an Amazon DocumentDB Cluster Parameter Group's Parameters

You can see an Amazon DocumentDB cluster's parameters and their values using the AWS Management Console or AWS CLI.

Using the AWS Management Console

To view the details of a cluster parameter group

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Parameter groups**.

Tip

If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.

3. In the **Parameter groups** pane, choose the name of the cluster parameter group that you want to see the details of.
4. The resulting page shows the following values for each parameter: the parameter's name, current value, allowed values, whether the parameter is modifiable, apply type, data type, and description.

Cluster parameter name	Values	Allowed values
audit_logs	disabled	enabled,disabled
tls	enabled	disabled,enabled
ttl_monitor	enabled	disabled,enabled

Using the AWS CLI

To see a cluster's parameter group's parameters and their values, use the `describe-db-cluster-parameters` operation with the following parameters.

- **--db-cluster-parameter-group-name** — Required. The name of the cluster parameter group for which you want a detailed parameter list.
- **--source** — Optional. If supplied, returns only parameters for a specific source. Parameter sources can be `engine-default`, `system`, or `user`.

Example

The following code lists the parameters and their values for the `custom3-6-param-grp` parameter group. For more information about the parameter group, omit the `--query` line. For information about all parameter groups, omit the `--db-cluster-parameter-group-name` line.

For Linux, macOS, or Unix:

```
aws docdb describe-db-cluster-parameters \
--db-cluster-parameter-group-name custom3-6-param-grp \
--query 'Parameters[*].[ParameterName,ParameterValue]'
```

For Windows:

```
aws docdb describe-db-cluster-parameters ^
--db-cluster-parameter-group-name custom3-6-param-grp ^
--query 'Parameters[*].[ParameterName,ParameterValue]'
```

Output from this operation looks something like the following (JSON format).

```
[  
  [  
    "audit_logs",  
    "disabled"  
  ],  
  [  
    "tls",  
    "enabled"  
  ],  
  [  
    "ttl_monitor",  
    "enabled"  
  ]]
```

Modifying an Amazon DocumentDB Cluster Parameter Group's Parameters

You can modify a parameter group's parameters using the AWS Management Console or AWS CLI.

Using the AWS Management Console

To update the parameters of a cluster parameter group

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Parameter groups**.

Tip
If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.
3. In the **Parameter groups** pane, choose the cluster parameter group that you want to update the parameters of.
4. The resulting page shows the parameters and their corresponding details for this cluster parameter group. Select a parameter to update.
5. On the top right of the page, choose **Edit** to change the value of the parameter. For more information about the types of cluster parameters, see [Amazon DocumentDB Cluster Parameters Reference \(p. 371\)](#).
6. Make your change, and then choose **Modify cluster parameter** to save the changes. To discard your changes, choose **Cancel**.

Using the AWS CLI

To modify a cluster parameter group's parameters, use the `modify-db-cluster-parameter-group` operation with the following parameters:

- **--db-cluster-parameter-group-name** — Required. The name of the cluster parameter group that you are modifying.
- **--parameters** — Required. The parameter or parameters that you are modifying. Each parameter entry must include the following:
 - **ParameterName** — The name of the parameter that you are modifying.
 - **ParameterValue** — The new value for this parameter.
 - **ApplyMethod** — How you want changes to this parameter applied. Permitted values are `immediate` and `pending-reboot`.

Note

Parameters with the `ApplyType` of `static` must have an `ApplyMethod` of `pending-reboot`.

To change the values of a cluster parameter group's parameters (AWS CLI)

The following example changes the `tls` parameter.

1. **List the parameters and their values of sample-parameter-group**

For Linux, macOS, or Unix:

```
aws docdb describe-db-cluster-parameters \
--db-cluster-parameter-group-name sample-parameter-group
```

For Windows:

```
aws docdb describe-db-cluster-parameters ^
```

```
--db-cluster-parameter-group-name sample-parameter-group
```

Output from this operation looks something like the following (JSON format).

```
{
  "Parameters": [
    {
      "Source": "system",
      "ApplyType": "static",
      "AllowedValues": "disabled,enabled",
      "ParameterValue": "enabled",
      "ApplyMethod": "pending-reboot",
      "DataType": "string",
      "ParameterName": "tls",
      "IsModifiable": true,
      "Description": "Config to enable/disable TLS"
    },
    {
      "Source": "user",
      "ApplyType": "dynamic",
      "AllowedValues": "disabled,enabled",
      "ParameterValue": "enabled",
      "ApplyMethod": "pending-reboot",
      "DataType": "string",
      "ParameterName": "ttl_monitor",
      "IsModifiable": true,
      "Description": "Enables TTL Monitoring"
    }
  ]
}
```

2. **Modify the `tls` parameter so that its value is disabled.** You can't modify the `ApplyMethod` because the `ApplyType` is static.

For Linux, macOS, or Unix:

```
aws docdb modify-db-cluster-parameter-group \
--db-cluster-parameter-group-name sample-parameter-group \
--parameters "ParameterName=tls,ParameterValue=disabled,ApplyMethod=pending-reboot"
```

For 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 from this operation looks something like the following (JSON format).

```
{
  "DBClusterParameterGroupName": "sample-parameter-group"
}
```

3. **Wait at least 5 minutes.**
4. **List the parameter values of `sample-parameter-group`.**

For Linux, macOS, or Unix:

```
aws docdb describe-db-cluster-parameters \
```

```
--db-cluster-parameter-group-name sample-parameter-group
```

For Windows:

```
aws docdb describe-db-cluster-parameters ^
--db-cluster-parameter-group-name sample-parameter-group
```

Output from this operation looks something like the following (JSON format).

```
{
  "Parameters": [
    {
      "ParameterName": "audit_logs",
      "ParameterValue": "disabled",
      "Description": "Enables auditing on cluster.",
      "Source": "system",
      "ApplyType": "dynamic",
      "DataType": "string",
      "AllowedValues": "enabled,disabled",
      "IsModifiable": true,
      "ApplyMethod": "pending-reboot"
    },
    {
      "ParameterName": "tls",
      "ParameterValue": "disabled",
      "Description": "Config to enable/disable TLS",
      "Source": "user",
      "ApplyType": "static",
      "DataType": "string",
      "AllowedValues": "disabled,enabled",
      "IsModifiable": true,
      "ApplyMethod": "pending-reboot"
    }
  ]
}
```

Understanding Amazon DocumentDB Endpoints

You can use Amazon DocumentDB (with MongoDB compatibility) endpoints to connect to a cluster or instance. Amazon DocumentDB has three different types of endpoints, each with its own purpose.

Topics

- [Finding a Cluster's Endpoints \(p. 379\)](#)
- [Finding an Instance's Endpoint \(p. 380\)](#)
- [Connecting to Endpoints \(p. 383\)](#)

Cluster endpoint

A cluster endpoint is an endpoint for an Amazon DocumentDB cluster that connects to the current primary instance for the cluster. Each Amazon DocumentDB cluster has a single cluster endpoint and one primary instance. In case of a failover, the cluster endpoint is remapped to the new primary instance.

Reader endpoint

A reader endpoint is an endpoint for an Amazon DocumentDB cluster that connects to one of the available replicas for that cluster. Each Amazon DocumentDB cluster has a reader endpoint. If there

is more than one replica, the reader endpoint directs each connection request to one of the Amazon DocumentDB replicas.

Instance endpoint

An instance endpoint is an endpoint that connects to a specific instance. Each instance in a cluster, regardless of whether it is a primary or replica instance, has its own unique instance endpoint. It is best to not use instance endpoints in your application. This is because they can change roles in case of a failover, thus requiring code changes in your application.

Finding a Cluster's Endpoints

You can find a cluster's cluster endpoint and reader endpoint using the Amazon DocumentDB console or AWS CLI.

Using the Console

To find a cluster's endpoints using the console

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **clusters**.
3. From the list of clusters, choose the name of the cluster you are interested in.
4. Scroll down to the **Details** section and locate the cluster endpoint and the reader endpoint.

Cluster details

Configurations and status

ARN
arn:aws:rds:us-east-1:XXXXXXXXXX:cluster:docdb-2019-01-09-23-55-38

Cluster identifier
docdb-2019-01-09-23-55-38 (available)

Cluster creation time
1/9/2019, 3:56:03 PM UTC-8

Cluster endpoint
docdb-2019-01-09-23-55-38.cluster-XXXXXXXXXX.us-east-1.docdb.amazonaws.com

Reader endpoint
docdb-2019-01-09-23-55-38.cluster-ro-XXXXXXXXXX.us-east-1.docdb.amazonaws.com

5. To connect to this cluster, scroll up to the **Connect** section. Locate the connection string for the mongo shell and a connection string that can be used in the application code to connect to your cluster.

Connect

Connect to this cluster with the mongo shell
mongo --ssl --host sample-cluster.cluster-crcjzrlsf.us-east-1.rds.amazonaws.com:27017 --sslCAFile rds-combined-ca-bundle.pem --username XXXXXXXXXX --password <insertYourPassword>

Connect to this Chimera cluster with a connection string
mongodb://<insertYourPassword>@sample-cluster.cluster-crcjzrlsf.us-east-1.rds.amazonaws.com:27017/?replicaSet=rs0&ssl_ca_certs=rds-combined-ca-bundle.pem

Using the AWS CLI

To find the cluster and reader endpoints for your cluster using the AWS CLI, run the `describe-db-clusters` command with these parameters.

Parameters

- **--db-cluster-identifier**—Optional. Specifies the cluster to return endpoints for. If omitted, returns endpoints for up to 100 of your clusters.
- **--query**—Optional. Specifies the fields to display. Helpful by reducing the amount of data that you need to view to find the endpoints. If omitted, all information about a cluster is returned.
- **--region**—Optional. Use the `--region` parameter to specify the Region that you want to apply the command to. If omitted, your default Region is used.

Example

The following example returns the `DBClusterIdentifier`, endpoint (cluster endpoint), and `ReaderEndpoint` for `sample-cluster`.

For Linux, macOS, or Unix:

```
aws docdb describe-db-clusters \
  --region us-east-1 \
  --db-cluster-identifier sample-cluster \
  --query 'DBClusters[*].[DBClusterIdentifier,Port,Endpoint,ReaderEndpoint]'
```

For Windows:

```
aws docdb describe-db-clusters ^
  --region us-east-1 ^
  --db-cluster-identifier sample-cluster ^
  --query 'DBClusters[*].[DBClusterIdentifier,Port,Endpoint,ReaderEndpoint]'
```

Output from this operation looks something like the following (JSON format).

```
[  
  [  
    "sample-cluster",  
    27017,  
    "sample-cluster.cluster-c0rlsfccj0zr.us-east-1.docdb.amazonaws.com",  
    "sample-cluster.cluster-ro-c0rlsfccj0zr.us-east-1.docdb.amazonaws.com"  
  ]  
]
```

Now that you have the cluster endpoint, you can connect to the cluster using either `mongo` or `mongodb`. For more information, see [Connecting to Endpoints \(p. 383\)](#).

Finding an Instance's Endpoint

You can find the endpoint for an instance using the Amazon DocumentDB console or the AWS CLI.

Using the Console

To find an instance's endpoint using the console

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.

2. In the navigation pane, choose **Clusters**.

Tip

If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.

3. In the Clusters navigation box, you'll see the column **Cluster Identifier**. Your instances are listed under clusters, similar to the screenshot below.

Cluster identifier	Role
docdb-cloud9-getstarted	Cluster Primary
robo3t	Cluster Primary

4. Check the box to the left of the instance you are interested in.
5. Scroll down to the **Details** section then locate the instance endpoint.

Details

Configurations and status

ARN
arn:aws:rds:us-east-1:...:db:docdb-2019-01-09-23-55-38

Instance identifier
docdb-2019-01-09-23-55-38 (available)

Instance creation time
1/9/2019, 4:02:10 PM UTC-8

Instance endpoint
docdb-2019-01-09-23-55-38.us-east-1.docdb.amazonaws.com

6. To connect to this instance, scroll up to the **Connect** section. Locate the connection string for the mongo shell and a connection string that can be used in your application code to connect to your instance.

Connect

Connect to this instance with the mongo shell
mongo --ssl --host docdb-2019-01-09-23-55-38.us-east-1.docdb.amazonaws.com:27017 --sslCAFile rds-combined-ca-bundle.pem --username <insertYourUsername> --password <insertYourPassword>

Connect to this cluster with an application
mongodb://<insertYourUsername>@docdb-2019-01-09-23-55-38.us-east-1.docdb.amazonaws.com:27017/?ssl_ca_certs=rds-combined-ca-bundle.pem

Using the AWS CLI

To find the instance endpoint using the AWS CLI, run the following command with these arguments.

Arguments

- **--db-instance-identifier**—Optional. Specifies the instance to return the endpoint for. If omitted, returns the endpoint for up to 100 of your instances.
- **--query**—Optional. Specifies the fields to display. Helpful by reducing the amount of data that you need to view to find the endpoints. If omitted, all information on an instance is returned. The Endpoint field has three members, so listing it in the query as in the following example returns all three members. If you're only interested in some of the Endpoint members, replace Endpoint in the query with the members you're interested in, as in the second example.
- **--region**—Optional. Use the --region parameter to specify the Region that you want to apply the command to. If omitted, your default Region is used.

Example

For Linux, macOS, or Unix:

```
aws docdb describe-db-instances \
  --region us-east-1 \
  --db-instance-identifier sample-cluster-instance \
  --query 'DBInstances[*].[DBInstanceIdentifier,Endpoint]'
```

For Windows:

```
aws docdb describe-db-instances ^
  --region us-east-1 ^
  --db-instance-identifier sample-cluster-instance ^
  --query 'DBInstances[*].[DBInstanceIdentifier,Endpoint]'
```

Output from this operation looks something like the following (JSON format).

```
[[
  [
    "sample-cluster-instance",
    {
      "Port": 27017,
      "Address": "sample-cluster-instance.corcjozrlsf.us-
east-1.docdb.amazonaws.com",
      "HostedZoneId": "Z2R2ITUGPM61AM"
    }
  ]
]]
```

Reducing the output to eliminate the endpoint's HostedZoneId, you can modify your query by specifying Endpoint.Port and Endpoint.Address.

For Linux, macOS, or Unix:

```
aws docdb describe-db-instances \
  --region us-east-1 \
  --db-instance-identifier sample-cluster-instance \
  --query 'DBInstances[*].[DBInstanceIdentifier,Endpoint.Port,Endpoint.Address]'
```

For Windows:

```
aws docdb describe-db-instances ^
  --region us-east-1 ^
  --db-instance-identifier sample-cluster-instance ^
```

```
--query 'DBInstances[*].[DBInstanceIdentifier,Endpoint.Port,Endpoint.Address]'
```

Output from this operation looks something like the following (JSON format).

```
[  
  [  
    "sample-cluster-instance",  
    27017,  
    "sample-cluster-instance.corcjozrlsfc.us-east-1.docdb.amazonaws.com"  
  ]  
]
```

Now that you have the instance endpoint, you can connect to the instance using either mongo or mongodb. For more information, see [Connecting to Endpoints \(p. 383\)](#).

Connecting to Endpoints

When you have your endpoint, either cluster or instance, you can connect to it using the mongo shell or a connection string.

Connecting Using the mongo Shell

Use the following structure to construct the string that you need to connect to your cluster or instance using the mongo shell:

```
mongo \  
  --ssl \  
  --host Endpoint:Port \  
  --sslCAFile rds-combined-ca-bundle.pem \  
  --username UserName \  
  --password Password
```

mongo shell examples

Connect to a cluster:

```
mongo \  
  --ssl \  
  --host sample-cluster.corcjozrlsfc.us-east-1.docdb.amazonaws.com:27017 \  
  --sslCAFile rds-combined-ca-bundle.pem \  
  --username UserName \  
  --password Password
```

Connect to an instance:

```
mongo \  
  --ssl \  
  --host sample-cluster-instance.corcjozrlsfc.us-east-1.docdb.amazonaws.com:27017 \  
  --sslCAFile rds-combined-ca-bundle.pem \  
  --username UserName \  
  --password Password
```

Connecting Using a Connection String

Use the following structure to construct the connection string that you need to connect to your cluster or instance.

```
mongodb://UserName:Password@endpoint:port?replicaSet=rs0&ssl_ca_certs=rds-combined-ca-bundle.pem
```

Connection string examples

Connect to a cluster:

```
mongodb://UserName:Password@sample-cluster.cluster-c0rlsfccj0zr.us-east-1.docdb.amazonaws.com:27017?replicaSet=rs0&ssl_ca_certs=rds-combined-ca-bundle.pem
```

Connect to an instance:

```
mongodb://UserName:Password@sample-cluster-instance.cluster-c0rlsfccj0zr.us-east-1.docdb.amazonaws.com:27017?replicaSet=rs0&ssl_ca_certs=rds-combined-ca-bundle.pem
```

Understanding Amazon DocumentDB Amazon Resource Names (ARNs)

Resources that you create in AWS are each uniquely identified with an Amazon Resource Name (ARN). For certain Amazon DocumentDB (with MongoDB compatibility) operations, you must uniquely identify an Amazon DocumentDB resource by specifying its ARN. For example, when you add a tag to a resource, you must provide the resource's ARN.

Topics

- [Constructing an ARN for an Amazon DocumentDB Resource \(p. 384\)](#)
- [Finding an Amazon DocumentDB Resource ARN \(p. 386\)](#)

Constructing an ARN for an Amazon DocumentDB Resource

You can construct an ARN for an Amazon DocumentDB resource using the following syntax. Amazon DocumentDB shares the format of Amazon Relational Database Service (Amazon RDS) ARNs. Amazon DocumentDB ARNs contain rds and not docdb.

`arn:aws:rds:region:account_number:resource_type:resource_id`

Region Name	Region	Availability Zones (compute)
US East (Ohio)	us-east-2	3
US East (N. Virginia)	us-east-1	6
US West (Oregon)	us-west-2	4
South America (São Paulo)	sa-east-1	3
Asia Pacific (Mumbai)	ap-south-1	3

Region Name	Region	Availability Zones (compute)
Asia Pacific (Seoul)	ap-northeast-2	4
Asia Pacific (Singapore)	ap-southeast-1	3
Asia Pacific (Sydney)	ap-southeast-2	3
Asia Pacific (Tokyo)	ap-northeast-1	3
Canada (Central)	ca-central-1	3
China (Beijing) Region	cn-north-1	3
China (Ningxia)	cn-northwest-1	3
Europe (Frankfurt)	eu-central-1	3
Europe (Ireland)	eu-west-1	3
Europe (London)	eu-west-2	3
Europe (Milan)	eu-south-1	3
Europe (Paris)	eu-west-3	3
AWS GovCloud (US)	us-gov-west-1	3

Note

The Amazon DocumentDB architecture separates storage and compute. For the storage layer, Amazon DocumentDB replicates six copies of your data across three AWS Availability Zones (AZs). The AZs listed in the table above are the number of AZs that you can use in a given region to provision compute instances. As an example, if you are launching an Amazon DocumentDB cluster in ap-northeast-1, your storage will be replicated six ways across three AZs but your compute instances will only be available in two AZs.

The following table shows the format that you should use when constructing an ARN for a particular Amazon DocumentDB resource. Amazon DocumentDB shares the format of Amazon RDS ARNs. Amazon DocumentDB ARNs contain `rds` and not `docdb`.

Resource Type	ARN Format / Example
Instance (db)	<code>arn:aws:rds:<i>region</i>:<i>account_number</i>:db:<i>resource_id</i></code> <code>arn:aws:rds:us-east-1:<i>1234567890</i>:db:<i>sample-db-instance</i></code>
Cluster (cluster)	<code>arn:aws:rds:<i>region</i>:<i>account_number</i>:cluster:<i>resource_id</i></code> <code>arn:aws:rds:us-east-1:<i>1234567890</i>:cluster:<i>sample-db-cluster</i></code>
Cluster parameter group (cluster-pg)	<code>arn:aws:rds:<i>region</i>:<i>account_number</i>:cluster-pg:<i>resource_id</i></code> <code>arn:aws:rds:us-east-1:<i>1234567890</i>:cluster-pg:<i>sample-db-cluster-parameter-group</i></code>
Security group (secgrp)	<code>arn:aws:rds:<i>region</i>:<i>account_number</i>:secgrp:<i>resource_id</i></code>

Resource Type	ARN Format / Example
	arn:aws:rds:us-east-1: <i>1234567890</i> :secgrp: <i>sample-public-secgrp</i>
Cluster snapshot (cluster-snapshot)	arn:aws:rds: <i>region:account_number</i> :cluster-snapshot: <i>resource_id</i> arn:aws:rds:us-east-1: <i>1234567890</i> :cluster-snapshot: <i>sample-db-cluster-snapshot</i>
Subnet group (subgrp)	arn:aws:rds: <i>region:account_number</i> :subgrp: <i>resource_id</i> arn:aws:rds:us-east-1: <i>1234567890</i> :subgrp: <i>sample-subnet-10</i>

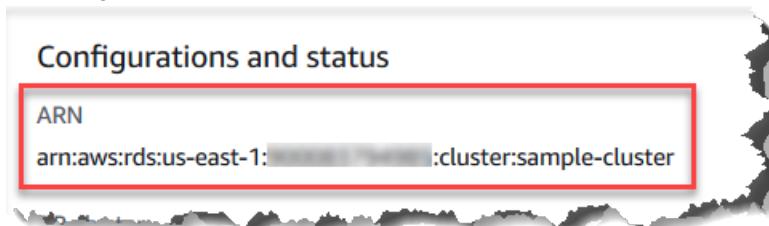
Finding an Amazon DocumentDB Resource ARN

You can find the ARN of an Amazon DocumentDB resource using the AWS Management Console or the AWS CLI.

Using the AWS Management Console

To find an ARN using the console, navigate to the resource that you want an ARN for, and view the details for that resource.

For example, you can get the ARN for a cluster in the **Details** pane for the cluster, as shown in the following screenshot.



Using the AWS CLI

To get an ARN using the AWS CLI for a particular Amazon DocumentDB resource, use the `describe` operation for that resource. The following table shows each AWS CLI operation and the ARN property that is used with the operation to get an ARN.

AWS CLI Command	ARN Property
<code>describe-db-instances</code>	<code>DBInstanceArn</code>
<code>describe-db-clusters</code>	<code>DBClusterArn</code>
<code>describe-db-parameter-groups</code>	<code>DBParameterGroupArn</code>
<code>describe-db-cluster-parameter-groups</code>	<code>DBClusterParameterGroupArn</code>
<code>describe-db-security-groups</code>	<code>DBSecurityGroupArn</code>

AWS CLI Command	ARN Property
describe-db-snapshots	DBSnapshotArn
describe-db-cluster-snapshots	DBClusterSnapshotArn
describe-db-subnet-groups	DBSubnetGroupArn

Example - Finding the ARN for a cluster

The following AWS CLI operation finds the ARN for the cluster sample-cluster.

For Linux, macOS, or Unix:

```
aws docdb describe-db-clusters \
--db-cluster-identifier sample-cluster \
--query 'DBClusters[*].DBClusterArn'
```

For Windows:

```
aws docdb describe-db-clusters ^
--db-cluster-identifier sample-cluster \
--query 'DBClusters[*].DBClusterArn'
```

Output from this operation looks something like the following (JSON format).

```
[ "arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster" ]
```

Example - Finding ARNs for multiple parameter groups

For Linux, macOS, or Unix:

```
aws docdb describe-db-cluster-parameter-groups \
--query 'DBClusterParameterGroups[*].DBClusterParameterGroupArn'
```

For Windows:

```
aws docdb describe-db-cluster-parameter-groups ^
--query 'DBClusterParameterGroups[*].DBClusterParameterGroupArn'
```

Output from this operation looks something like the following (JSON format).

```
[ "arn:aws:rds:us-east-1:123456789012:cluster-pg:custom3-6-param-grp",
"arn:aws:rds:us-east-1:123456789012:cluster-pg:default.aurora5.6",
"arn:aws:rds:us-east-1:123456789012:cluster-pg:default.docdb3.6" ]
```

Tagging Amazon DocumentDB Resources

You can use Amazon DocumentDB (with MongoDB compatibility) tags to add metadata to your Amazon DocumentDB resources. These tags can be used with AWS Identity and Access Management (IAM) policies

to manage access to Amazon DocumentDB resources and to control what actions can be applied to the resources. You can also use tags to track costs by grouping expenses for similarly tagged resources.

You can tag the following Amazon DocumentDB resources:

- Clusters
- Instances
- Snapshots
- Cluster snapshots
- Parameter groups
- Cluster parameter groups
- Security groups
- Subnet groups

Overview of Amazon DocumentDB Resource Tags

An Amazon DocumentDB tag is a name-value pair that you define and associate with an Amazon DocumentDB resource. The name is referred to as the *key*. Supplying a value for the key is optional. You can use tags to assign arbitrary information to an Amazon DocumentDB resource. You can use a tag key, for example, to define a category, and the tag value might be an item in that category. For example, you might define a tag key of `project` and a tag value of `Salix`, indicating that the Amazon DocumentDB resource is assigned to the `Salix` project. You can also use tags to designate Amazon DocumentDB resources as being used for test or production by using a key such as `environment=test` or `environment=production`. We recommend that you use a consistent set of tag keys to make it easier to track metadata that is associated with Amazon DocumentDB resources.

You can use tags to organize your AWS bill to reflect your own cost structure. To do this, sign up to get your AWS account bill with tag key values included. Then, to see the cost of combined resources, organize your billing information according to resources with the same tag key values. For example, you can tag several resources with a specific application name, and then organize your billing information to see the total cost of that application across several services. For more information, see [Using Cost Allocation Tags](#) in the *AWS Billing and Cost Management User Guide*.

Each Amazon DocumentDB resource has a tag set, which contains all the tags that are assigned to that resource. A tag set can contain as many as 10 tags, or it can be empty. If you add a tag to an Amazon DocumentDB resource that has the same key as an existing tag on resource, the new value overwrites the old value.

AWS does not apply any semantic meaning to your tags; tags are interpreted strictly as character strings. Amazon DocumentDB can set tags on an instance or other Amazon DocumentDB resources, depending on the settings that you use when you create the resource. For example, Amazon DocumentDB might add a tag indicating that an instance is for production or for testing.

You can add a tag to a snapshot, but your bill will not reflect this grouping.

You can use the AWS Management Console or the AWS CLI to add, list, and delete tags on Amazon DocumentDB resources. When using the AWS CLI, you must provide the Amazon Resource Name (ARN) for the resource that you want to work with. For more information about Amazon DocumentDB ARNs, see [Understanding Amazon DocumentDB Amazon Resource Names \(ARNs\)](#) (p. 384).

Tag Constraints

The following constraints apply to Amazon DocumentDB tags:

- Maximum number of tags per resource - 10

- Maximum **Key** length - 128 Unicode characters
- Maximum **Value** length - 256 Unicode characters
- Valid characters for **Key** and **Value** - uppercase and lowercase letters in the UTF-8 character set, digits, space, and the following characters: _ . : / = + - and @ (Java regex: "^(\\p{L}\\p{Z})*\\p{N}_\\p{L}:\\p{Z}*=\\p{L}-*)\$")
- Tag keys and values are case sensitive.
- The prefix aws: cannot be used for tag keys or values; it is reserved for AWS.

Adding and Updating Tags on an Amazon DocumentDB Resource

You can add up to 10 tags to a resource using the AWS Management Console or the AWS CLI.

Using the AWS Management Console

The process for adding a tag to a resource is similar regardless of what resource you're adding the tag to. In this example, you add a tag to a cluster.

To add or update tags to a cluster using the console

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. From the navigation pane, choose **clusters**.
3. Choose the name of the cluster that you want to add tags to.
4. Scroll down to the **Tags** section, and then choose **Edit**.
5. For each tag you want to add to this resource, do the following:
 - a. To add a new tag, enter in the name of the tag in the **Key** box. To change a tag's value, find the tag's name in the **Key** column.
 - b. To give the tag a new or updated value, enter a value for the tag in the **Value** box.
 - c. If you have more tags to add, choose **Add**. Otherwise, when finished, choose **Save**.

Using the AWS CLI

The process for adding a tag to a resource is similar regardless of what resource you're adding the tags to. In this example, you add three tags to a cluster. The second tag, key2, has no value.

Use the AWS CLI operation `add-tags-to-resource` with these parameters.

Parameters

- **--resource-name**—The ARN of the Amazon DocumentDB resource that you want to add tags to.
- **--tags**—A list the tags (key-value pair) that you want to add to this resource in the format `Key=key-name, Value=tag-value`.

Example

For Linux, macOS, or 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
```

For 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
```

The add-tags-to-resource operation produces no output. To see the results of the operation, use the list-tags-for-resource operation.

Listing Tags on an Amazon DocumentDB Resource

You can use the AWS Management Console or the AWS CLI to get a listing of the tags for an Amazon DocumentDB resource.

Using the AWS Management Console

The process for listing tags on a resource is similar regardless of what resource you're adding the tag to. In this example, you list the tags for a cluster.

To list the tags on a cluster using the console

1. Open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. From the navigation pane, choose **clusters**.
3. Choose the name of the cluster that you want to list tags for.
4. To see a listing of the tags on this resource, scroll down to the **Tags** section.

Using the AWS CLI

The process for listing the tags on a resource is similar regardless of what resource you're listing the tag for. In this example, you list the tags on a cluster.

Use the AWS CLI operation `list-tags-for-resource` with these parameters.

Parameters

- **--resource-name**—Required. The ARN of the Amazon DocumentDB resource that you want to list tags for.

Example

For Linux, macOS, or Unix:

```
aws docdb list-tags-for-resource \
--resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster
```

For Windows:

```
aws docdb list-tags-for-resource ^
--resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster
```

Output from this operation looks something like the following (JSON format).

```
{  
    "TagList": [  
        {  
            "Key": "key1",  
            "Value": "value1"  
        },  
        {  
            "Key": "key2",  
            "Value": ""  
        },  
        {  
            "Key": "key3",  
            "Value": "value3"  
        }  
    ]  
}
```

Removing Tags from an Amazon DocumentDB Resource

You can use the AWS Management Console or the AWS CLI to remove tags from Amazon DocumentDB resources.

Using the AWS Management Console

The process for removing tags from a resource is similar regardless of what resource you're adding the tag to. In this example, you remove tags from a cluster.

To remove tags from a cluster using the console

1. Open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. From the navigation pane, choose **clusters**.
3. Choose the name of the cluster that you want to remove tags from.
4. Scroll down to the **Tags** section, and then choose **Edit**.
5. If you want to remove all tags from this resource, choose **Remove all**. Otherwise, for each tag that you want to remove from this resource, do the following:
 - a. Locate the name of the tag in the **Key** column.
 - b. Choose **Remove** on the same row as the tag key.
 - c. When finished, choose **Save**.

Using the AWS CLI

The process for removing a tag from a resource is similar regardless of what resource you're removing the tag from. In this example, you remove a tag from a cluster.

Use the AWS CLI operation `remove-tags-from-resource` with these parameters.

- **--resource-name**—Required. The ARN of the Amazon DocumentDB resource that you want to remove tags from.
- **--tag-keys**—Required. A list the tag keys that you want removed from this resource.

Example

For Linux, macOS, or Unix:

```
aws docdb remove-tags-from-resource \
--resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster \
--tag-keys key1 key3
```

For Windows:

```
aws docdb remove-tags-from-resource ^
--resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster \
--tag-keys key1 key3
```

The `remove-tags-from-resource` operation produces no output. To see the results of the operation, use the `list-tags-for-resource` operation.

Maintaining Amazon DocumentDB

Periodically, Amazon DocumentDB performs maintenance on Amazon DocumentDB resources. Maintenance most often involves updates to the database engine (cluster maintenance) or the instance's underlying operating system (OS) (instance maintenance).

Some maintenance items require that Amazon DocumentDB take your instance offline for a short time. Maintenance items that require an instance to be offline include required operating system or engine patching. Required patching is automatically scheduled only for patches that are related to security and instance reliability. You should expect that when maintenance is performed on your cluster or instance, if the instance is a primary instance, it will fail over. For more information, see [Amazon DocumentDB Failover \(p. 345\)](#).

Both cluster and instances maintenance have their own respective maintenance windows. By default, when you create a cluster, Amazon DocumentDB assigns a maintenance window for both a cluster and each individual instance. You can choose the maintenance window when creating a cluster or an instance. You can also modify the maintenance windows at any time to fit your business schedules or practices. It is generally advised to choose maintenance windows that minimize the impact of the maintenance on your application (for example, on evenings or weekends). This guidance is highly contextual upon the type of application and usage patterns that you experience.

Topics

- [Determining Pending Amazon DocumentDB Maintenance Actions \(p. 392\)](#)
- [Applying Amazon DocumentDB Updates \(p. 393\)](#)
- [User-Initiated Updates \(p. 396\)](#)
- [Managing Your Amazon DocumentDB Maintenance Windows \(p. 397\)](#)

Determining Pending Amazon DocumentDB Maintenance Actions

You can view whether a maintenance update is available for your cluster by using the AWS Management Console or the AWS CLI.

If an update is available, you can do one of the following:

- Defer the maintenance actions.
- Apply the maintenance actions immediately.
- Schedule the maintenance actions to start during your next maintenance window.
- Take no action.

Important

Certain OS updates are marked as **Required**. If you defer a required update, you receive a notice from Amazon DocumentDB indicating when the update will be performed on your instance or cluster. Other updates are **Available**. You can defer these updates indefinitely.

The maintenance window determines when pending operations start, but it does not limit the total execution time of these operations. Maintenance operations are not guaranteed to finish before the maintenance window ends, and they can continue beyond the specified end time.

Using the AWS Management Console

If an update is available, it is indicated by the word **Available** or **Required** in the **Maintenance** column for the cluster on the Amazon DocumentDB console, as shown here:

DB clusters (2)	
Actions ▾	
DB cluster identifier	Status
sample-cluster	active
sample-cluster-7	active
Maintenance	
	None
	None

Using the AWS CLI

Use the following AWS CLI operation to determine what maintenance actions are pending. The output here shows no pending maintenance actions.

```
aws docdb describe-pending-maintenance-actions
```

Output from this operation looks something like the following (JSON format).

```
{  
    "PendingMaintenanceActions": []  
}
```

Applying Amazon DocumentDB Updates

With Amazon DocumentDB, you can choose when to apply maintenance operations. You can decide when Amazon DocumentDB applies updates using the AWS Management Console or AWS CLI.

Use the procedures in this topic to immediately upgrade or schedule an upgrade for your instance.

Using the AWS Management Console

You can use the console to manage updates for your Amazon DocumentDB instances and clusters.

To manage an update for an instance or cluster

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **clusters**.
3. In the list of clusters, choose the button next to the name of the cluster that you want to apply the maintenance operation to.
4. On the **Actions** menu, choose one of the following:
 - **Upgrade now** to immediately perform the pending maintenance tasks.
 - **Upgrade at next window** to perform the pending maintenance tasks during the cluster's next maintenance window.

Note

If there are no pending maintenance tasks, both of the preceding options are inactive.

Using the AWS CLI

To apply a pending update to an instance or cluster, use the `apply-pending-maintenance-action` AWS CLI operation.

Parameters

- **--resource-identifier**—The Amazon DocumentDB Amazon Resource Name (ARN) of the resource that the pending maintenance action applies to.
- **--apply-action**—The pending maintenance action to apply to this resource.
Valid values: `system-update` and `db-upgrade`.
- **--opt-in-type**—A value that specifies the type of opt-in request, or undoes an opt-in request. An opt-in request of type `immediate` can't be undone.
Valid values:
 - `immediate`—Apply the maintenance action immediately.
 - `next-maintenance`—Apply the maintenance action during the next maintenance window for the resource.
 - `undo-opt-in`—Cancel any existing `next-maintenance` opt-in requests.

Example

For Linux, macOS, or Unix:

```
aws docdb apply-pending-maintenance-action \
  --resource-identifier arn:aws:rds:us-east-1:123456789012:db:docdb \
  --apply-action system-update \
  --opt-in-type immediate
```

For Windows:

```
aws docdb apply-pending-maintenance-action ^
  --resource-identifier arn:aws:rds:us-east-1:123456789012:db:docdb ^
  --apply-action system-update ^
  --opt-in-type immediate
```

To return a list of resources that have at least one pending update, use the `describe-pending-maintenance-actions` AWS CLI operation.

Example

For Linux, macOS, or Unix:

```
aws docdb describe-pending-maintenance-actions \
--resource-identifier arn:aws:rds:us-east-1:001234567890:db:docdb
```

For Windows:

```
aws docdb describe-pending-maintenance-actions ^
--resource-identifier arn:aws:rds:us-east-1:001234567890:db:docdb
```

Output from this operation looks something like the following (JSON format).

```
{
  "PendingMaintenanceActions": [
    {
      "ResourceIdentifier": "arn:aws:rds:us-east-1:001234567890:cluster:sample-cluster",
      "PendingMaintenanceActionDetails": [
        {
          "Action": "system-update",
          "CurrentApplyDate": "2019-01-11T03:01:00Z",
          "Description": "db-version-upgrade",
          "ForcedApplyDate": "2019-01-18T03:01:00Z",
          "AutoAppliedAfterDate": "2019-01-11T03:01:00Z"
        }
      ]
    }
  ]
}
```

You can also return a list of resources for an instance or cluster by specifying the `--filters` parameter of the `describe-pending-maintenance-actions` AWS CLI operation. The format for the `--filters` operation is `Name=filter-name,Values=resource-id,...`.

The following are acceptable values for the Name parameter of filter:

- `db-cluster-id`—Accepts a list of cluster identifiers or ARNs. The returned list only includes pending maintenance actions for the clusters identified by these identifiers or ARNs.
- `db-instance-id`—Accepts a list of instance identifiers or ARNs. The returned list only includes pending maintenance actions for the instances identified by these identifiers or ARNs.

The following example returns the pending maintenance actions for the `sample-cluster1` and `sample-cluster2` clusters.

Example

For Linux, macOS, or Unix:

```
aws docdb describe-pending-maintenance-actions \
--filters Name=db-cluster-id,Values=sample-cluster1,sample-cluster2
```

For Windows:

```
aws docdb describe-pending-maintenance-actions ^
--filters Name=db-cluster-id,Values=sample-cluster1,sample-cluster2
```

Apply Dates

Each maintenance action has a respective apply date that you can find when describing the pending maintenance actions. When you read the output of pending maintenance actions from the AWS CLI, three dates are listed:

- **CurrentApplyDate**—The date the maintenance action will get applied either immediately or during the next maintenance window. If the maintenance is optional, this value can be null.
- **ForcedApplyDate**—The date when the maintenance will be automatically applied, independent of your maintenance window.
- **AutoAppliedAfterDate**—The date after which the maintenance will be applied during the cluster's maintenance window.

User-Initiated Updates

As an Amazon DocumentDB user, you can initiate updates to your clusters or instances. For example, you can modify an instance's class to one with more or less memory, or you can change a cluster's parameter group. Amazon DocumentDB views these changes differently from Amazon DocumentDB initiated updates. For more information about modifying a cluster or instance, see the following:

- [Modifying an Amazon DocumentDB Cluster \(p. 287\)](#)
- [Modifying an Amazon DocumentDB Instance \(p. 326\)](#)

To see a list of pending user initiated modifications, run the following command.

Example

To see pending user initiated changes for your instances

For Linux, macOS, or Unix:

```
aws docdb describe-db-instances \
  --query 'DBInstances[*].
[DBClusterIdentifier,DBInstanceIdentifier,PendingModifiedValues]'
```

For Windows:

```
aws docdb describe-db-instances ^
  --query 'DBInstances[*].
[DBClusterIdentifier,DBInstanceIdentifier,PendingModifiedValues]'
```

Output from this operation looks something like the following (JSON format).

In this case, `sample-cluster-instance` has a pending change to a `db.r5.xlarge` instance class, while `sample-cluster-instance-2` has no pending changes.

```
[  
  [  
    "sample-cluster",  
    "sample-cluster-instance",  
    {  
      "DBInstanceClass": "db.r5.xlarge"  
    }  
  ],  
  [  
  ]
```

```

        "sample-cluster",
        "sample-cluster-instance-2",
        []
    ]
]
```

Managing Your Amazon DocumentDB Maintenance Windows

Each instance and cluster has a weekly maintenance window during which any pending changes are applied. The maintenance window is an opportunity to control when modifications and software patching occur, in the event either are requested or required. If a maintenance event is scheduled for a given week, it is initiated during the 30-minute maintenance window that you identify. Most maintenance events also complete during the 30-minute maintenance window, although larger maintenance events might take more than 30 minutes to complete.

The 30-minute maintenance window is selected at random from an 8-hour block of time per Region. If you don't specify a preferred maintenance window when you create the instance or cluster, Amazon DocumentDB assigns a 30-minute maintenance window on a randomly selected day of the week.

Region	UTC Time Block
US East (Ohio)	03:00-11:00
US East (N. Virginia)	03:00-11:00
US West (Oregon)	06:00-14:00
Asia Pacific (Mumbai)	17:30-01:30
Asia Pacific (Seoul)	13:00-21:00
Asia Pacific (Singapore)	14:00-22:00
Asia Pacific (Sydney)	12:00-20:00
Asia Pacific (Tokyo)	13:00-21:00
Canada (Central)	22:00-06:00
Europe (Frankfurt)	23:00-07:00
Europe (Ireland)	22:00-06:00
Europe (Paris)	22:00-06:00
AWS GovCloud (US)	22:00-06:00

Changing a Maintenance Window

The maintenance window should fall at the time of lowest usage and thus might need changing from time to time. Your cluster or instance is unavailable during this time only if system changes (such as a scale storage operation or an instance class change) are being applied and require an outage. And then it is unavailable only for the minimum amount of time required to make the necessary changes.

For upgrades to the database engine, Amazon DocumentDB uses the cluster's preferred maintenance window and not the maintenance window for individual instances.

To change the maintenance window

- For a cluster, see [Modifying an Amazon DocumentDB Cluster \(p. 287\)](#).
- For an instance, see [Modifying an Amazon DocumentDB Instance \(p. 326\)](#).

Understanding Service-Linked Roles

Amazon DocumentDB (with MongoDB compatibility) uses AWS Identity and Access Management (IAM) service-linked roles. A [service-linked role](#) is a unique type of IAM role that is linked directly to Amazon DocumentDB. Service-linked roles are predefined by Amazon DocumentDB and include all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes using Amazon DocumentDB easier because you don't have to manually add the necessary permissions. Amazon DocumentDB defines the permissions of its service-linked roles, and unless defined otherwise, only Amazon DocumentDB can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

You can delete the roles only after first deleting their related resources. This protects your Amazon DocumentDB resources because you can't inadvertently remove permission to access the resources.

For information about other services that support service-linked roles, see [AWS Services That Work with IAM](#) and look for the services that have **Yes** in the **Service-Linked Role** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

Amazon DocumentDB Service-Linked Role Permissions

Amazon DocumentDB (with MongoDB compatibility) uses the service-linked role named **AWSServiceRoleForRDS** to allow Amazon DocumentDB to call AWS services on behalf of your clusters.

The **AWSServiceRoleForRDS** service-linked role trusts the following services to assume the role:

- `docdb.amazonaws.com`

The role permissions policy allows Amazon DocumentDB to complete the following actions on the specified resources:

- Actions on `ec2`:
 - `AssignPrivateIpAddresses`
 - `AuthorizeSecurityGroupIngress`
 - `CreateNetworkInterface`
 - `CreateSecurityGroup`
 - `DeleteNetworkInterface`
 - `DeleteSecurityGroup`
 - `DescribeAvailabilityZones`
 - `DescribeInternetGateways`
 - `DescribeSecurityGroups`
 - `DescribeSubnets`
 - `DescribeVpcAttribute`

- `DescribeVpcs`
- `ModifyNetworkInterfaceAttribute`
- `RevokeSecurityGroupIngress`
- `UnassignPrivateIpAddresses`
- Actions on sns:
 - `ListTopic`
 - `Publish`
- Actions on cloudwatch:
 - `PutMetricData`
 - `GetMetricData`
 - `CreateLogStream`
 - `PullLogEvents`
 - `DescribeLogStreams`
 - `CreateLogGroup`

Note

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. You might encounter the following error message:

Unable to create the resource. Verify that you have permission to create service linked role. Otherwise wait and try again later.

If you see this error, ensure that you have the following permissions enabled:

```
{  
    "Action": "iam:CreateServiceLinkedRole",  
    "Effect": "Allow",  
    "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/  
AWSServiceRoleForRDS",  
    "Condition": {  
        "StringLike": {  
            "iam:AWSServiceName": "rds.amazonaws.com"  
        }  
    }  
}
```

For more information, see [Service-Linked Role Permissions](#) in the *IAM User Guide*.

Creating an Amazon DocumentDB Service-Linked Role

You don't need to manually create a service-linked role. When you create a cluster, Amazon DocumentDB creates the service-linked role for you.

If you delete this service-linked role and then need to create it again, you can use the same process to re-create the role in your account. When you create a cluster, Amazon DocumentDB creates the service-linked role for you again.

Modifying an Amazon DocumentDB Service-Linked Role

Amazon DocumentDB does not allow you to modify the AWSServiceRoleForRDS service-linked role. After you create a service-linked role, you cannot change the name of the role because various entities might

reference the role. However, you can modify the description of the role using IAM. For more information, see [Editing a Service-Linked Role](#) in the *IAM User Guide*.

Deleting an Amazon DocumentDB Service-Linked Role

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored or maintained. However, you must delete all of your clusters before you can delete the service-linked role.

Cleaning Up an Amazon DocumentDB Service-Linked Role

Before you can use IAM to delete a service-linked role, you must first confirm that the role has no active sessions and remove any resources used by the role.

To check whether the service-linked role has an active session using the console

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane of the IAM console, choose **Roles**, and then choose the name (not the check box) of the **AWSServiceRoleForRDS** role.
3. On the **Summary** page for the selected role, choose the **Access Advisor** tab.
4. On the **Access Advisor** tab, review the recent activity for the service-linked role.

Note

If you are unsure whether Amazon DocumentDB is using the **AWSServiceRoleForRDS** role, you can try to delete the role. If the service is using the role, then the deletion fails and you can view the Regions where the role is being used. If the role is being used, then you must wait for the session to end before you can delete the role. You cannot revoke the session for a service-linked role.

If you want to remove the **AWSServiceRoleForRDS** role, you must first delete *all* your instances and clusters. For information about deleting instances and clusters, see the following topics:

- [Deleting an Amazon DocumentDB Instance \(p. 331\)](#)
- [Deleting an Amazon DocumentDB Cluster \(p. 296\)](#)

Supported Regions for Amazon DocumentDB Service-Linked Roles

Amazon DocumentDB supports using service-linked roles in all of the Regions where the service is available. For more information, see <https://docs.aws.amazon.com/documentdb/latest/developerguide/regions-and-azs.html#regions-and-azs-availability>.

Monitoring Amazon DocumentDB

Monitoring your AWS services is an important part of keeping your systems healthy and functioning optimally. It's wise to collect monitoring data from all parts of your AWS solution so that you can more easily debug and fix failures or degradations, should they occur. Before you begin monitoring your AWS solutions, we recommend that you consider and formulate answers for the following questions:

- What are your monitoring goals?
- What resources are you going to monitor?
- How frequently will you monitor these resources?
- What monitoring tools will you use?
- Who is responsible for doing the monitoring?
- Who is to be notified and by what means if something goes wrong?

To understand your current performance patterns, identify performance anomalies, and formulate methods to address issues, you should establish baseline performance metrics for various times and under differing load conditions. As you monitor your AWS solution, we recommend that you store your historical monitoring data for future reference and for establishing your baselines.

In general, acceptable values for performance metrics depend on what your baseline looks like and what your application is doing. Investigate consistent or trending variances from your baseline. The following is advice about specific types of metrics:

- **High CPU or RAM use** — High values for CPU or RAM use might be appropriate, provided that they are in keeping with your goals for your application (like throughput or concurrency) and are expected.
- **Storage volume consumption** —Investigate storage consumption (`VolumeBytesUsed`) if space that is used is consistently at or above 85 percent of the total storage volume space. Determine whether you can delete data from the storage volume or archive data to a different system to free up space. For more information, see [Amazon DocumentDB Storage \(p. 10\)](#) and [Amazon DocumentDB Quotas and Limits \(p. 535\)](#).
- **Network traffic** — For network traffic, talk with your system administrator to understand what the expected throughput is for your domain network and internet connection. Investigate network traffic if throughput is consistently lower than expected.
- **Database connections** — Consider constraining database connections if you see high numbers of user connections in conjunction with decreases in instance performance and response time. The best number of user connections for your instance will vary based on your instance class and the complexity of the operations being performed.
- **IOPS metrics**—The expected values for IOPS metrics depend on disk specification and server configuration, so use your baseline to know what is typical. Investigate if values are consistently different from your baseline. For best IOPS performance, make sure that your typical working set fits into memory to minimize read and write operations.

Amazon DocumentDB (with MongoDB compatibility) provides a variety of Amazon CloudWatch metrics that you can monitor to determine the health and performance of your Amazon DocumentDB clusters and instances. You can view Amazon DocumentDB metrics using various tools, including the Amazon DocumentDB console, AWS CLI, CloudWatch API, and Performance Insights.

Topics

- [Monitoring an Amazon DocumentDB Cluster's Status \(p. 402\)](#)
- [Monitoring an Amazon DocumentDB Instance's Status \(p. 404\)](#)
- [Using Amazon DocumentDB Event Subscriptions \(p. 407\)](#)

- [Monitoring Amazon DocumentDB with CloudWatch \(p. 414\)](#)
- [Logging Amazon DocumentDB API Calls with AWS CloudTrail \(p. 426\)](#)
- [Profiling Amazon DocumentDB Operations \(p. 427\)](#)
- [Monitoring with Performance Insights \(p. 434\)](#)

Monitoring an Amazon DocumentDB Cluster's Status

The status of a cluster indicates the health of the cluster. You can view the status of a cluster by using the Amazon DocumentDB console or the AWS CLI `describe-db-clusters` command.

Topics

- [Cluster Status Values \(p. 402\)](#)
- [Monitoring a Cluster's Status Using the AWS Management Console \(p. 403\)](#)
- [Monitoring a Cluster's Status Using the AWS CLI \(p. 404\)](#)

Cluster Status Values

The following table lists the valid values for a cluster's status.

Cluster Status	Description
available	The cluster is healthy and available.
backing-up	The cluster is currently being backed up.
creating	The cluster is being created. It is inaccessible while it is being created.
deleting	The cluster is being deleted. It is inaccessible while it is being deleted.
failing-over	A failover from the primary instance to an Amazon DocumentDB replica is being performed.
inaccessible-encryption-credentials	The AWS KMS key used to encrypt or decrypt the cluster can't be accessed.
maintenance	A maintenance update is being applied to the cluster. This status is used for cluster-level maintenance that Amazon DocumentDB schedules well in advance.

Cluster Status	Description
migrating	A cluster snapshot is being restored to a cluster.
migration-failed	A migration failed.
modifying	The cluster is being modified because of a customer request to modify the cluster.
renaming	The cluster is being renamed because of a customer request to rename it.
resetting-master-credentials	The master credentials for the cluster are being reset because of a customer request to reset them.
upgrading	The cluster engine version is being upgraded.

Monitoring a Cluster's Status Using the AWS Management Console

When using the AWS Management Console to determine the status of a cluster, use the following procedure.

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Clusters**.
3. In the Clusters navigation box, you'll see the column **Cluster identifier**. Your instances are listed under clusters, similar to the screenshot below.

Cluster identifier	Role	Status
docdb-cloud9-getstarted	Cluster	Primary
robo3t	Cluster	Primary

4. In the **Cluster identifier** column, find the name of the instance that you are interested in. Then, to find the status of the instance, read across that row to the **Status** column, as shown below.

Clusters (1)				Action
<input type="text"/> Filter clusters				<
Cluster identifier	Engine version	Status	Instances	
docdb-2020-10-23-22-23-28	docdb 3.6.0	available	1	

Monitoring a Cluster's Status Using the AWS CLI

When using the AWS CLI to determine the status of a cluster, use the `describe-db-clusters` operation. The following code finds the status of the cluster `sample-cluster`.

For Linux, macOS, or Unix:

```
aws docdb describe-db-clusters \
--db-cluster-identifier sample-cluster \
--query 'DBClusters[*].[DBClusterIdentifier,Status]'
```

For Windows:

```
aws docdb describe-db-clusters ^
--db-cluster-identifier sample-cluster ^
--query 'DBClusters[*].[DBClusterIdentifier,Status]'
```

Output from this operation looks something like the following.

```
[  
  [  
    "sample-cluster",  
    "available"  
  ]  
]
```

Monitoring an Amazon DocumentDB Instance's Status

The status of an instance indicates the health of the instance. You can view the status of an instance in Amazon DocumentDB (with MongoDB compatibility) by using the AWS Management Console or the AWS CLI operation `describe-db-instances`.

Note

Amazon DocumentDB also uses another status called *maintenance status*, which is shown in the **Maintenance** column of the Amazon DocumentDB console. This value indicates the status of any maintenance patches that need to be applied to an instance. Maintenance status is independent of the Amazon DocumentDB instance status. For more information about maintenance status, see [Applying Amazon DocumentDB Updates \(p. 393\)](#).

Topics

- [Instance Status Values \(p. 405\)](#)
- [Monitoring an Instance's Status Using the AWS Management Console \(p. 406\)](#)

- [Monitoring an Instance's Status Using the AWS CLI \(p. 407\)](#)

Instance Status Values

The following table lists the possible status values for instances and how you are billed for each status. It shows if you will be billed for the instance and storage, only storage, or not billed. For all instance statuses, you are always billed for backup usage.

Valid Values for an Instance's Status

Instance status	Billed	Description
available	Billed	The instance is healthy and available.
backing-up	Billed	The instance is currently being backed up.
configuring-log-exports	Billed	Publishing log files to Amazon CloudWatch Logs is being enabled or disabled for this instance.
creating	Not billed	The instance is being created. The instance is not accessible while it is being created.
deleting	Not billed	The instance is being deleted.
failed	Not billed	The instance has failed and Amazon DocumentDB was unable to recover it. To recover the data, perform a point-in-time restore to the latest restorable time of the instance.
inaccessible-encryption-credentials	Not billed	The AWS KMS key that is used to encrypt or decrypt the instance could not be accessed.
incompatible-network	Not billed	Amazon DocumentDB is attempting to perform a recovery action on an instance but is unable to do so because the VPC is in a state that is preventing the action from being completed. This status can occur if, for example, all available IP addresses in a subnet were in use and Amazon DocumentDB was unable to get an IP address for the instance.
maintenance	Billed	Amazon DocumentDB is applying a maintenance update to the instance. This status is used for instance-level maintenance that Amazon DocumentDB schedules well in advance. We're evaluating ways to expose additional maintenance actions to customers through this status.
modifying	Billed	The instance is being modified because of a request to modify the instance.
rebooting	Billed	The instance is being rebooted because of a request or an Amazon DocumentDB process that requires the rebooting of the instance.

Instance status	Billed	Description
renaming	Billed	The instance is being renamed because of a request to rename it.
resetting-master-credentials	Billed	The master credentials for the instance are being reset because of a request to reset them.
restore-error	Billed	The instance encountered an error attempting to restore to a point-in-time or from a snapshot.
starting	Billed for storage	The instance is starting.
stopped	Billed for storage	The instance is stopped.
stopping	Billed for storage	The instance is being stopped.
storage-full	Billed	The instance has reached its storage capacity allocation. This is a critical status and should be remedied immediately; scale up your storage by modifying the instance. Set Amazon CloudWatch alarms to warn you when storage space is getting low so you don't run into this situation.

Monitoring an Instance's Status Using the AWS Management Console

When using the AWS CLI to determine the status of a cluster, use the following procedure.

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Clusters**.
3. Note that in the Clusters navigation box, the column **Cluster Identifier** shows both clusters and instances. Instances are listed underneath clusters, similar to the screenshot below.

Clusters (2)				
<input type="text"/> Filter Resources				
	Cluster identifier	Role	Engine version	Region & AZ
	docdb-cloud9-getstarted	Cluster	3.6.0	us-east-1
	docdb-cloud9-getstarted	Primary	3.6.0	us-east-1f
	robo3t	Cluster	3.6.0	us-east-1
	robo3t	Primary	3.6.0	us-east-1d

4. Find the name of the instance that you are interested in. Then, to find the status of the instance, read across that row to the **Status** column, as shown following.

Cluster identifier	Role	Engine version	Region & AZ	Status
docdb-cloud9-getstarted	Cluster	3.6.0	us-east-1	available
docdb-cloud9-getstarted	Primary	3.6.0	us-east-1f	available
robo3t	Cluster	3.6.0	us-east-1	available
robo3t	Primary	3.6.0	us-east-1d	available

Monitoring an Instance's Status Using the AWS CLI

When using the AWS CLI to determine the status of a cluster, use the `describe-db-instances` operation. The following code finds the status of the instance `sample-cluster-instance-01`.

For Linux, macOS, or Unix:

```
aws docdb describe-db-instances \
    --db-instance-identifier sample-cluster-instance-01 \
    --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceStatus]'
```

For Windows:

```
aws docdb describe-db-instances ^
    --db-instance-identifier sample-cluster-instance-01 ^
    --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceStatus]'
```

Output from this operation looks something like the following.

```
[  
  [  
    "sample-cluster-instance-01",  
    "available"  
  ]  
]
```

Using Amazon DocumentDB Event Subscriptions

Amazon DocumentDB uses Amazon Simple Notification Service (Amazon SNS) to provide notifications when an Amazon DocumentDB event occurs. These notifications can be in any form that is supported by Amazon SNS for an AWS Region, such as an email, a text message, or a call to an HTTP endpoint.

Amazon DocumentDB groups these events into categories that you can subscribe to so that you can be notified when an event in that category occurs. You can subscribe to an event category for an instance, cluster, snapshot, cluster snapshot, or for a parameter group. For example, if you subscribe to the `Backup` category for a given instance, you are notified whenever a backup-related event occurs that affects the instance. You also receive notification when an event subscription changes.

Events occur at both the cluster and the instance level, so you can receive events if you subscribe to a cluster or an instance.

Event subscriptions are sent to the addresses you provide when you create the subscription. You might want to create several different subscriptions, such as a subscription that receives all event notifications and another subscription that includes only critical events for your production instances. You can easily turn off notification without deleting a subscription. To do so, set the **Enabled** radio button to **No** in the Amazon DocumentDB console.

Important

Amazon DocumentDB doesn't guarantee the order of events sent in an event stream. The event order is subject to change.

Amazon DocumentDB uses the Amazon Resource Name (ARN) of an Amazon SNS topic to identify each subscription. The Amazon DocumentDB console creates the ARN for you when you create the subscription.

Billing for Amazon DocumentDB event subscriptions is through Amazon SNS. Amazon SNS fees apply when using event notification. For more information, see [Amazon Simple Notification Service Pricing](#). Other than Amazon SNS charges, Amazon DocumentDB does not bill for event subscriptions.

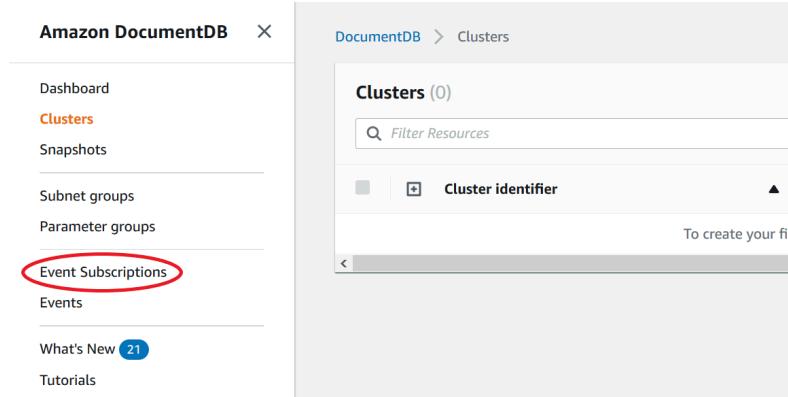
Topics

- [Subscribing to Amazon DocumentDB Event Subscriptions \(p. 408\)](#)
- [Managing Amazon DocumentDB Event Notification Subscriptions \(p. 409\)](#)
- [Amazon DocumentDB Event Categories and Messages \(p. 412\)](#)

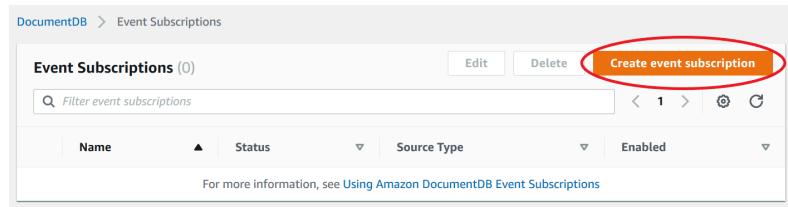
Subscribing to Amazon DocumentDB Event Subscriptions

You can use the Amazon DocumentDB console to subscribe to event subscriptions, as follows:

1. Sign in to the AWS Management Console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Event subscriptions**.



3. In the **Event subscriptions** pane, choose **Create event subscription**.



4. In the **Create event subscription** dialog box, do the following:

- For **Name**, enter a name for the event notification subscription.

DocumentDB > Event Subscriptions > Create event subscription

Create event subscription

Details

Name
Name of the subscription
Test

- For **Target**, choose where you want to send notifications to. You can choose an existing **ARN** or choose **New Email Topic** to enter the name of a topic and a list of recipients.

Target

Send notifications to

ARN
 New Email Topic

ARN
ARN to send notifications to
Choose ARN

- For **Source**, choose a source type. Depending on the source type you selected, choose the event categories and the sources that you want to receive event notifications from.

Source

Source Type
Source type of resource this subscription will consume events from
Choose source type

- Choose **Create**.

Source

Source Type
Source type of resource this subscription will consume events from
Instances

Instances to include
Instances that this subscription will consume events from

All instances
 Select specific instances

Event Categories to include
Event Categories that this subscription will consume events from

All event categories
 Select specific event categories

Cancel **Create**

Managing Amazon DocumentDB Event Notification Subscriptions

If you choose **Event subscriptions** in the navigation pane of the Amazon DocumentDB console, you can view subscription categories and a list of your current subscriptions. You can also modify or delete a specific subscription.

To modify your current Amazon DocumentDB event notification subscriptions

1. Sign in to the AWS Management Console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Event subscriptions**. The **Event subscriptions** pane shows all your event notification subscriptions.

Name	Status	Source Type
test	active	db-instance

3. In the **Event subscriptions** pane, choose the subscription that you want to modify and choose **Edit**.

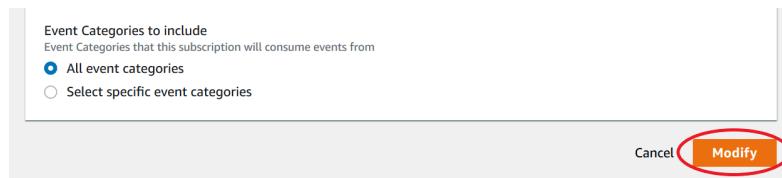
Name	Status	Source Type	Enabled
test	active	db-instance	true

4. Make your changes to the subscription in either the **Target** or **Source** section. You can add or remove source identifiers by selecting or deselecting them in the **Source** section.

Modify event subscription

Details	
Enabled	<input checked="" type="radio"/> Enabled <input type="radio"/> Disabled
Target	
Send notifications to	<input checked="" type="radio"/> ARN <input type="radio"/> New Email Topic
ARN	ARN to send notifications to Test

5. Choose **Modify**. The Amazon DocumentDB console indicates that the subscription is being modified.



Deleting an Amazon DocumentDB event notification subscription

You can delete a subscription when you no longer need it. All subscribers to the topic will no longer receive event notifications specified by the subscription.

1. Sign in to the AWS Management Console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Event subscriptions**.

Name	Status	Source Type
test	active	db-instance

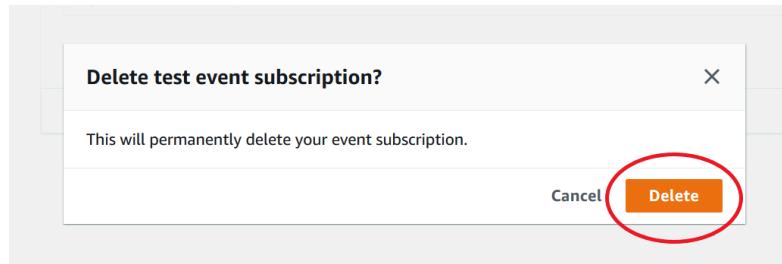
3. In the **Event subscriptions** pane, choose the subscription that you want to delete.

Name	Status	Source Type	Enabled
test	active	db-instance	true

4. Choose **Delete**.

Name	Status	Source Type	Enabled
test	active	db-instance	true

5. A pop-up window will appear asking you if you want to permanently delete this notification. Choose **Delete**.



Amazon DocumentDB Event Categories and Messages

Amazon DocumentDB generates a significant number of events in categories that you can subscribe to using the console. Each category applies to a source type, which can be an instance, snapshot, or parameter group.

Note

Amazon DocumentDB uses existing Amazon RDS event definitions and IDs.

Amazon DocumentDB events originating from instances

Category	Description
availability	The instance restarted.
availability	The instance shutdown.
configuration change	Applying modification to an instance class.
configuration change	Finished applying modification to an instance class.
configuration change	Reset master credentials.
creation	Instance created.
deletion	Instance deleted
failure	The instance has failed due to an incompatible configuration or an underlying storage issue. Begin a point-in-time-restore for the instance.
notification	Instance stopped.
notification	Instance started.
notification	Instance is being started due to it exceeding the maximum allowed time being stopped.
recovery	Recovery of the instance has started. Recovery time will vary with the amount of data to be recovered.
recovery	Recovery of the instance is complete.

Amazon DocumentDB events originating from a cluster

Category	Description
creation	Cluster created
deletion	Cluster deleted.
failover	Promoting previous primary again.
failover	Completed failover to instance.
failover	Started failover to DB instance: %s
failover	Started same AZ failover to DB instance: %s
failover	Started cross AZ failover to DB instance: %s
maintenance	Cluster has been patched.
maintenance	Database cluster is in a state that cannot be upgraded: %s
notification	The cluster stopped.
notification	The cluster started.
notification	The cluster stop failed.
notification	The cluster is being started due to it exceeding the maximum allowed time being stopped.
notification	Renamed cluster from %s to %s.

Amazon DocumentDB events originating from cluster snapshot

The following table shows the event category and a list of events when an Amazon DocumentDB cluster snapshot is the source type.

Category	Description
backup	Creating manual cluster snapshot.
backup	Manual cluster snapshot created.
backup	Creating automated cluster snapshot.
backup	Automated cluster snapshot created.

Amazon DocumentDB events originating from parameter group

The following table shows the event category and a list of events when a parameter group is the source type.

Category	Description
configuration change	Updated parameter %s to %s with apply method %s

Monitoring Amazon DocumentDB with CloudWatch

Amazon DocumentDB (with MongoDB compatibility) integrates with Amazon CloudWatch so that you can gather and analyze operational metrics for your clusters. You can monitor these metrics using the CloudWatch console, the Amazon DocumentDB console, the AWS Command Line Interface (AWS CLI), or the CloudWatch API.

CloudWatch also lets you set alarms so that you can be notified if a metric value breaches a threshold that you specify. You can even set up Amazon CloudWatch Events to take corrective action if a breach occurs. For more information about using CloudWatch and alarms, see the [Amazon CloudWatch documentation](#).

Topics

- [Amazon DocumentDB Metrics \(p. 414\)](#)
- [Viewing CloudWatch Data \(p. 422\)](#)
- [Amazon DocumentDB Dimensions \(p. 426\)](#)
- [Monitoring Opcounters \(p. 426\)](#)
- [Monitoring Database Connections \(p. 426\)](#)

Amazon DocumentDB Metrics

To monitor the health and performance of your Amazon DocumentDB cluster and instances, you can view the following metrics in the Amazon DocumentDB console.

Resource Utilization

Metric	Description
BackupRetentionPeriodStorageUsed	The total amount of backup storage in GiB used to support the point-in-time restore feature within the Amazon DocumentDB's retention window. Included in the total reported by the TotalBackupStorageBilled metric. Computed separately for each Amazon DocumentDB cluster.
ChangeStreamLogSize	The amount of storage used by your cluster to store the change stream log in megabytes. This value is a subset of the total storage for the cluster (VolumeBytesUsed) and affects

Metric	Description
	the cost of the cluster. For storage pricing information, see the Amazon DocumentDB product page . The change stream log size is a function of how much change is happening on your cluster and the change stream long retention duration. For more information on change streams, see Using Change Streams with Amazon DocumentDB (p. 486) .
CPUUtilization	The percentage of CPU used by an instance.
DatabaseConnections	The number of connections open on an instance taken at a one-minute frequency.
DatabaseConnectionsMax	The maximum number of open database connections on an instance in a one-minute period.
DatabaseCursors	The number of cursors open on an instance taken at a one-minute frequency.
DatabaseCursorsMax	The maximum number of open cursors on an instance in a one-minute period.
DatabaseCursorsTimedOut	The number of cursors that timed out in a one-minute period.
FreeableMemory	The amount of available random access memory, in bytes.
FreeLocalStorage	This metric reports the amount of storage available to each instance for temporary tables and logs. This value depends on the instance class. You can increase the amount of free storage space for an instance by choosing a larger instance class for your instance.
LowMemThrottleQueueDepth	The queue depth for requests that are throttled due to low available memory taken at a one-minute frequency.

Metric	Description
LowMemThrottleMaxQueueDepth	The maximum queue depth for requests that are throttled due to low available memory in a one-minute period.
LowMemNumOperationsThrottled	The number of requests that are throttled due to low available memory in a one-minute period.
SnapshotStorageUsed	The total amount of backup storage in GiB consumed by all snapshots for a given Amazon DocumentDB cluster outside its backup retention window. Included in the total reported by the TotalBackupStorageBilled metric. Computed separately for each Amazon DocumentDB cluster.
SwapUsage	The amount of swap space used on the instance.
TotalBackupStorageBilled	The total amount of backup storage in GiB for which you are billed for a given Amazon DocumentDB cluster. Includes the backup storage measured by the BackupRetentionPeriodStorageUsed and SnapshotStorageUsed metrics. Computed separately for each Amazon DocumentDB cluster.
TransactionsOpen	The number of transactions open on an instance taken at a one-minute frequency.
TransactionsOpenMax	The maximum number of transactions open on an instance in a one-minute period.
VolumeBytesUsed	The amount of storage used by your cluster in bytes. This value affects the cost of the cluster. For pricing information, see the Amazon DocumentDB product page .

Latency

Metric	Description	
DBClusterReplicaLagMaximum	The maximum amount of lag, in milliseconds, between the primary instance and each Amazon DocumentDB instance in the cluster.	
DBClusterReplicaLagMinimum	The minimum amount of lag, in milliseconds, between the primary instance and each replica instance in the cluster.	
DBInstanceReplicaLag	The amount of lag, in milliseconds, when replicating updates from the primary instance to a replica instance.	
ReadLatency	The average amount of time taken per disk I/O operation.	
WriteLatency	The average amount of time, in milliseconds, taken per disk I/O operation.	

Operations

Metric	Description	
DocumentsDeleted	The number of deleted documents in a one-minute period.	
DocumentsInserted	The number of inserted documents in a one-minute period.	
DocumentsReturned	The number of returned documents in a one-minute period.	
DocumentsUpdated	The number of updated documents in a one-minute period.	
OpcountersCommand	The number of commands issued in a one-minute period.	
OpcountersDelete	The number of delete operations issued in a one-minute period.	
OpcountersGetmore	The number of getmores issued in a one-minute period.	
OpcountersInsert	The number of insert operations issued in a one-minute period.	

Metric	Description
OpcountersQuery	The number of queries issued in a one-minute period.
OpcountersUpdate	The number of update operations issued in a one-minute period.
TransactionsStarted	The number of transactions started on an instance in a one-minute period.
TransactionsCommitted	The number of transactions committed on an instance in a one-minute period.
TransactionsAborted	The number of transactions aborted on an instance in a one-minute period.
TTLDeletedDocuments	The number of documents deleted by a TTLMonitor in a one-minute period.

Throughput

Metric	Description
NetworkReceiveThroughput	The amount of network throughput, in bytes per second, received from clients by each instance in the cluster. This throughput doesn't include network traffic between instances in the cluster and the cluster volume.
NetworkThroughput	The amount of network throughput, in bytes per second, both received from and transmitted to clients by each instance in the Amazon DocumentDB cluster. This throughput doesn't include network traffic between instances in the cluster and the cluster volume.
NetworkTransmitThroughput	The amount of network throughput, in bytes per second, sent to clients by each instance in the cluster. This throughput doesn't include network traffic between instances in the cluster and the cluster volume.

Metric	Description	
ReadIOPS	The average number of disk read I/O operations per second. Amazon DocumentDB reports read and write IOPS separately, and on one-minute intervals.	
ReadThroughput	The average number of bytes read from disk per second.	
VolumeReadIOPs	<p>The average number of billed read I/O operations from a cluster volume, reported at 5-minute intervals. Billed read operations are calculated at the cluster volume level, aggregated from all instances in the cluster, and then reported at 5-minute intervals. The value is calculated by taking the value of the read operations metric over a 5-minute period. You can determine the amount of billed read operations per second by taking the value of the billed read operations metric and dividing by 300 seconds.</p> <p>For example, if the VolumeReadIOPs returns 13,686, then the billed read operations per second is 45 ($13,686 / 300 = 45.62$).</p> <p>You accrue billed read operations for queries that request database pages that are not present in the buffer cache and therefore must be loaded from storage. You might see spikes in billed read operations as query results are read from storage and then loaded into the buffer cache.</p>	

Metric	Description	
VolumeWriteIOPs	<p>The average number of billed write I/O operations from a cluster volume, reported at 5-minute intervals. Billed write operations are calculated at the cluster volume level, aggregated from all instances in the cluster, and then reported at 5-minute intervals. The value is calculated by taking the value of the write operations metric over a 5-minute period. You can determine the amount of billed write operations per second by taking the value of the billed write operations metric and dividing by 300 seconds.</p> <p>For example, if the VolumeWriteIOPs returns 13,686, then the billed write operations per second is 45 ($13,686 / 300 = 45.62$).</p> <p>Note that VolumeReadIOPs and VolumeWriteIOPs metrics are calculated by the DocumentDB storage layer and it includes IOs performed by the primary and replica instances. The data is aggregated every 20-30 minutes and then reported at 5-minute intervals, thus emitting the same data point for the metric in the time period. If you are looking for a metric to correlate to your insert operations over a 1-minute interval, you can use the instance level WriteIOPS metric. The metric is available in the monitoring tab of your Amazon DocumentDB primary instance.</p>	
WriteIOPS	The average number of disk write I/O operations per second. When used on a cluster level, WriteIOPs are evaluated across all instances in the cluster. Read and write IOPS are reported separately, on 1-minute intervals.	
WriteThroughput	The average number of bytes written to disk per second.	

System

Metric	Description
BufferCacheHitRatio	The percentage of requests that are served by the buffer cache.
DiskQueueDepth	the number of concurrent write requests to the distributed storage volume.
EngineUptime	The amount of time, in seconds, that the instance has been running.
IndexBufferCacheHitRatio	The percentage of index requests that are served by the buffer cache. You might see a spike greater than 100% for the metric right after you drop an index, collection or database. This will automatically be corrected after 60 seconds. This limitation will be fixed in a future patch update.

T3 Instance Metrics

Metric	Description
CPUCreditUsage	The number of CPU credits spent during the measurement period.
CPUCreditBalance	The number of CPU credits that an instance has accrued. This balance is depleted when the CPU bursts and CPU credits are spent more quickly than they are earned.
CPUSurplusCreditBalance	The number of surplus CPU credits spent to sustain CPU performance when the CPUCreditBalance value is zero.
CPUSurplusCreditsCharged	The number of surplus CPU credits exceeding the maximum number of CPU credits that can be earned in a 24-hour period, and thus attracting an additional charge. For more information, see Monitoring your CPU credits .

Viewing CloudWatch Data

You can view Amazon CloudWatch data using the CloudWatch console, the Amazon DocumentDB console, AWS Command Line Interface (AWS CLI), or the CloudWatch API.

View CloudWatch Metrics Using the Amazon DocumentDB Management Console

To view CloudWatch metrics using the Amazon DocumentDB Management Console, complete the following steps.

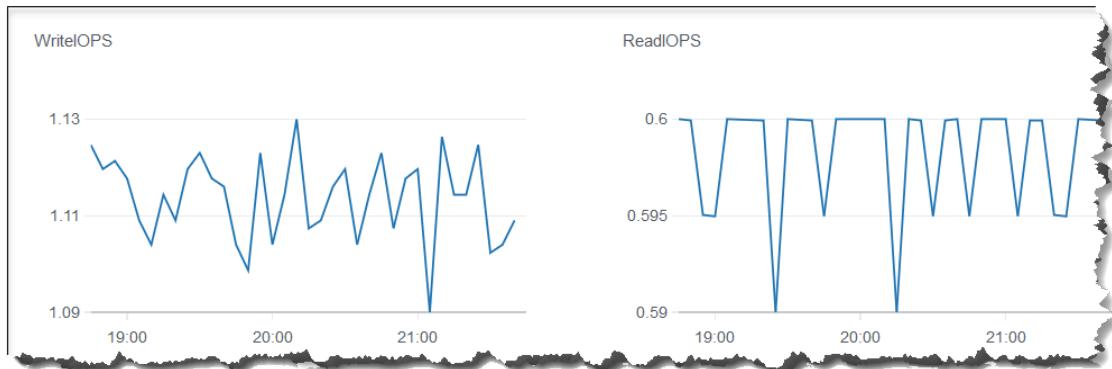
1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Clusters**.

Tip
If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.
3. In the Clusters navigation box, you'll see the column **Cluster Identifier**. Your instances are listed under clusters, similar to the screenshot below.

Cluster identifier	Role	Status
docdb-cloud9-getstarted	Cluster	Primary
robo3t	Cluster	Primary

4. From the list of instances, choose the name of the instance that you want metrics for.
5. In the resulting instance summary page, choose the **Monitoring** tab to view graphical representations of your Amazon DocumentDB instance's metrics. Because a graph must be generated for each metric, it might take a few minutes for the **CloudWatch** graphs to populate.

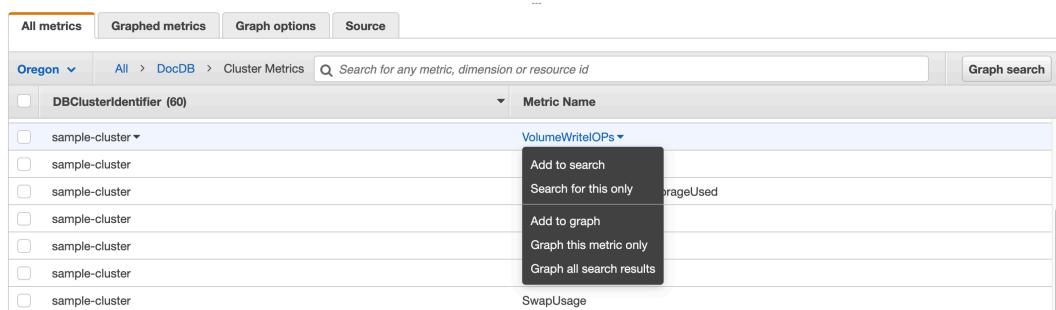
The following image shows the graphical representations of two CloudWatch metrics in the Amazon DocumentDB console, WriteIOPS and ReadIOPS.



View CloudWatch Metrics Using the CloudWatch Management Console

To view CloudWatch metrics using the CloudWatch Management Console, complete the following steps.

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/cloudwatch>.
2. In the navigation pane, choose **Metrics**. Then, from the list of service names, choose **DocDB**.
3. Choose a metric dimension (for example, **Cluster Metrics**).
4. The **All metrics** tab displays all metrics for that dimension in **DocDB**.
 - a. To sort the table, use the column heading.
 - b. To graph a metric, select the check box next to the metric. To select all metrics, select the check box in the heading row of the table.
 - c. To filter by metric, hover over the metric name and select the drop-down arrow next to the metric name. Then, choose **Add to search**, as shown in the image below.



View CloudWatch Metrics Using the AWS CLI

To view CloudWatch data for Amazon DocumentDB, use the `CloudWatch get-metric-statistics` operation with the following parameters.

Parameters

- **--namespace** — Required. The service namespace for which you want CloudWatch metrics. For Amazon DocumentDB, this must be `AWS/DocDB`.
- **--metric-name** — Required. The name of the metric for which you want data.
- **--start-time** — Required. The timestamp that determines the first data point to return.

The value specified is inclusive; results include data points with the specified timestamp. The timestamp must be in ISO 8601 UTC format (for example, `2016-10-03T23:00:00Z`).

- **--end-time** — Required. The timestamp that determines the last data point to return.

The value specified is inclusive; results include data points with the specified timestamp. The timestamp must be in ISO 8601 UTC format (for example, 2016-10-03T23:00:00Z).

- **--period** — Required. The granularity, in seconds, of the returned data points. For metrics with regular resolution, a period can be as short as one minute (60 seconds) and must be a multiple of 60. For high-resolution metrics that are collected at intervals of less than one minute, the period can be 1, 5, 10, 30, 60, or any multiple of 60.
- **--dimensions** — Optional. If the metric contains multiple dimensions, you must include a value for each dimension. CloudWatch treats each unique combination of dimensions as a separate metric. If a specific combination of dimensions was not published, you can't retrieve statistics for it. You must specify the same dimensions that were used when the metrics were created.
- **--statistics** — Optional. The metric statistics, other than percentile. For percentile statistics, use ExtendedStatistics. When calling GetMetricStatistics, you must specify either Statistics or ExtendedStatistics, but not both.

Permitted values:

- SampleCount
- Average
- Sum
- Minimum
- Maximum
- **--extended-statistics** — Optional. The percentile statistics. Specify values between p0.0 and p100. When calling GetMetricStatistics, you must specify either Statistics or ExtendedStatistics, but not both.
- **--unit** — Optional. The unit for a given metric. Metrics may be reported in multiple units. Not supplying a unit results in all units being returned. If you specify only a unit that the metric does not report, the results of the call are null.

Possible values:

- Seconds
- Microseconds
- Milliseconds
- Bytes
- Kilobytes
- Megabytes
- Gigabytes
- Terabytes
- Bits
- Kilobits
- 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

The following example finds the maximum CPUUtilization for a 2-hour period taking a sample every 60 seconds.

For Linux, macOS, or 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
```

For 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 from this operation look something like the following.

```
{
    "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": 4.25000000000485,
            "Timestamp": "2019-02-11T06:44:00Z"
        }
    ]
}
```

```
        "Maximum": 4.3333333331878,  
        "Timestamp": "2019-02-11T06:07:00Z"  
    }  
}  
]
```

Amazon DocumentDB Dimensions

The metrics for Amazon DocumentDB are qualified by the values for the account or operation. You can use the CloudWatch console to retrieve Amazon DocumentDB data filtered by any of the dimensions in the following table.

Dimension	Description
DBClusterIdentifier	Filters the data that you request for a specific Amazon DocumentDB cluster.
DBClusterIdentifier, Role	Filters the data that you request for a specific Amazon DocumentDB cluster, aggregating the metric by instance role (WRITER/READER). For example, you can aggregate metrics for all READER instances that belong to a cluster.
DBInstanceIdentifier	Filters the data that you request for a specific database instance.

Monitoring Opcounters

Opcounter metrics have a non-zero value (usually ~50) for idle clusters. This is because Amazon DocumentDB performs periodic health checks, internal operations, and metrics collection tasks.

Monitoring Database Connections

When you view the number of connections by using database engine commands such as `db.runCommand({ serverStatus: 1 })`, you might see up to 10 more connections than you see in DatabaseConnections through CloudWatch. This occurs because Amazon DocumentDB performs periodic health checks and metrics collection tasks that don't get accounted for in DatabaseConnections. DatabaseConnections represents customer-initiated connections only.

Logging Amazon DocumentDB API Calls with AWS CloudTrail

Amazon DocumentDB (with MongoDB compatibility) is integrated with AWS CloudTrail, a service that provides a record of actions taken by IAM users, IAM roles, or an AWS service in Amazon DocumentDB (with MongoDB compatibility). CloudTrail captures all AWS CLI API calls for Amazon DocumentDB as events, including calls from the Amazon DocumentDB console and from code calls to the Amazon DocumentDB SDK. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Amazon DocumentDB. If you don't configure a trail, you can still view the most recent events on the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to Amazon DocumentDB (with MongoDB compatibility), the IP address from which the request was made, who made the request, when it was made, and other details.

Important

For certain management features, Amazon DocumentDB uses operational technology that is shared with Amazon Relational Database Service (Amazon RDS). Amazon DocumentDB console, AWS CLI, and API calls are logged as calls made to the Amazon RDS API.

To learn more about AWS CloudTrail, see [AWS CloudTrail User Guide](#).

Amazon DocumentDB Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in Amazon DocumentDB (with MongoDB compatibility), that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see [Viewing Events with CloudTrail Event History](#).

For an ongoing record of events in your AWS account, including events for Amazon DocumentDB (with MongoDB compatibility), create a trail. A trail enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following topics in the *AWS CloudTrail User Guide*:

- [Overview for Creating a Trail](#)
- [CloudTrail Supported Services and Integrations](#)
- [Configuring Amazon SNS Notifications for CloudTrail](#)
- [Receiving CloudTrail Log Files from Multiple Regions](#)
- [Receiving CloudTrail Log Files from Multiple Accounts](#)

Every event or log entry includes information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or IAM user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the [CloudTrail userIdentity Element](#).

Profiling Amazon DocumentDB Operations

You can use the profiler in Amazon DocumentDB (with MongoDB compatibility) to log the execution time and details of operations that were performed on your cluster. The profiler is useful for monitoring the slowest operations on your cluster to help you improve individual query performance and overall cluster performance.

By default, the profiler feature is disabled. When enabled, the profiler logs operations that are taking longer than a customer-defined threshold value (for example, 100 ms) to Amazon CloudWatch Logs. Logged details include the profiled command, time, plan summary, and client metadata. After the operations are logged to CloudWatch Logs, you can use CloudWatch Logs Insights to analyze, monitor, and archive your Amazon DocumentDB profiling data. Common queries are provided in the section [Common Queries \(p. 434\)](#).

When enabled, the profiler uses additional resources in your cluster. We recommend that you start with a high threshold value (for example, 500 ms) and gradually lower the value to identify slow operations. Starting with a threshold value of 50 ms can cause performance issues on your cluster for

high throughput applications. The profiler is enabled at the cluster level and works on all instances and databases in a cluster. Amazon DocumentDB logs operations to Amazon CloudWatch Logs on a best-effort basis.

Although Amazon DocumentDB imposes no additional charge to enable the profiler, you are charged the standard rates for the usage of CloudWatch Logs. For information about CloudWatch Logs pricing, see [Amazon CloudWatch pricing](#).

Topics

- [Supported Operations \(p. 428\)](#)
- [Limitations \(p. 428\)](#)
- [Enabling the Amazon DocumentDB Profiler \(p. 428\)](#)
- [Disabling the Amazon DocumentDB Profiler \(p. 431\)](#)
- [Disabling Profiler Logs Export \(p. 432\)](#)
- [Accessing Your Amazon DocumentDB Profiler Logs \(p. 434\)](#)
- [Common Queries \(p. 434\)](#)

Supported Operations

Amazon DocumentDB profiler supports the following operations:

- aggregate
- count
- delete
- distinct
- find (OP_QUERY and command)
- findAndModify
- insert
- update

Limitations

The slow query profiler is only able to emit profiler logs if the entire result set of the query is able to fit in one batch, and if the result set is under 16MB (maximum BSON size). Result sets greater than 16MB are automatically split into multiple batches.

Most drivers or shells may set a default batch size that is small. You can specify the batch size as part of your query. For the purpose of capturing slow query logs, we recommend a batch size that exceeds the size of your expected result set. If you are unsure of the result set size, or if it varies, you can also set the batch size to a large number (e.g., 100k).

However, using a larger batch size means more results will have to be retrieved from the database before a response is sent to the client. For some queries, that may create longer delays before you get results. If you do not plan to consume the entire result set, it is possible that you will spend more I/Os to process the query and throw away the result.

Enabling the Amazon DocumentDB Profiler

Enabling the profiler on a cluster is a three-step process. Ensure that all steps are completed, or profiling logs will not be sent to CloudWatch Logs. Profiler is set at the cluster level and is performed on all of the cluster's databases and instances.

To enable the profiler on a cluster

1. Because you can't modify a default cluster parameter group, ensure that you have an available custom cluster parameter group. For more information, see [Creating Amazon DocumentDB Cluster Parameter Groups \(p. 360\)](#).
2. Using an available custom cluster parameter group, modify the following parameters: `profiler`, `profiler_threshold_ms`, and `profiler_sampling_rate`. For more information, see [Modifying Amazon DocumentDB Cluster Parameter Groups \(p. 362\)](#).
3. Create or modify your cluster to use the custom cluster parameter group and to enable exporting `profiler` logs to CloudWatch Logs.

The following sections show how to implement these steps using the AWS Management Console and the AWS Command Line Interface (AWS CLI).

Using the AWS Management Console

1. Before you begin, create a Amazon DocumentDB cluster and a custom cluster parameter group if you don't already have one. For more information, see [Creating Amazon DocumentDB Cluster Parameter Groups \(p. 360\)](#) and [Creating an Amazon DocumentDB Cluster \(p. 274\)](#).
2. Using an available custom cluster parameter group, modify the following parameters. For more information, see [Modifying Amazon DocumentDB Cluster Parameter Groups \(p. 362\)](#).
 - `profiler` — Enables or disables query profiling. Permitted values are `enabled` and `disabled`. The default value is `disabled`. To enable profiling, set the value to `enabled`.
 - `profiler_threshold_ms` — When `profiler` is set to `enabled`, all commands that are taking longer than `profiler-threshold-ms` are logged to CloudWatch. Permitted values are `[50-INT_MAX]`. The default value is `100`.
 - `profiler_sampling_rate` — The fraction of slow operations that should be profiled or logged. Permitted values are `[0.0-1.0]`. The default value is `1.0`.
3. Modify your cluster to use the custom cluster parameter group and set the profiler log exports to publish to Amazon CloudWatch.
 - a. In the navigation pane, choose **Clusters** to add your custom parameter group to a cluster.
 - b. Choose the button to the left of the name of the cluster that you want to associate your parameter group with. Select **Actions**, and then **Modify** to modify your cluster.
 - c. Under **Cluster options**, choose the custom parameter group from the step above to add it to your cluster.
 - d. Under **Log exports**, select **Profiler logs** to publish to Amazon CloudWatch.
 - e. Choose **Continue** to view a summary of your modifications.
 - f. After verifying your changes, you can apply them immediately or during the next maintenance window under **Scheduling of modifications**.
 - g. Choose **Modify cluster** to update your cluster with your new parameter group.

Using the AWS CLI

The following procedure enables the profiler on all supported operations for the cluster sample-cluster.

1. Before you begin, ensure that you have an available custom cluster parameter group by running the following command, and reviewing the output for a cluster parameter group that doesn't have `default` in the name and has `docdb3.6` as the parameter group family. If you don't have a non-default cluster parameter group, see [Creating Amazon DocumentDB Cluster Parameter Groups \(p. 360\)](#).

```
aws docdb describe-db-cluster-parameter-groups \
  --query 'DBClusterParameterGroups[*].
[DBClusterParameterGroupName,DBParameterGroupFamily]'
```

In the following output, only `sample-parameter-group` meets both criteria.

```
[[
  [
    "default.docdb3.6",
    "docdb3.6"
  ],
  [
    "sample-parameter-group",
    "docdb3.6"
  ]
]]
```

- Using your custom cluster parameter group, modify the following parameters:

- `profiler` — Enables or disables query profiling. Permitted values are `enabled` and `disabled`. The default value is `disabled`. To enable profiling, set the value to `enabled`.
- `profiler_threshold_ms` — When `profiler` is set to `enabled`, all commands taking longer than `profiler_threshold_ms` are logged to CloudWatch. Permitted values are `[0-INT_MAX]`. Setting this value to `0` profiles all supported operations. The default value is `100`.
- `profiler_sampling_rate` — The fraction of slow operations that should be profiled or logged. Permitted values are `[0.0-1.0]`. The default value is `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
```

- Modify your Amazon DocumentDB cluster so that it uses the `sample-parameter-group` custom cluster parameter group from the previous step and sets the parameter `--enable-cloudwatch-logs-exports` to `profiler`.

The following code modifies the cluster `sample-cluster` to use the `sample-parameter-group` from the previous step, and adds `profiler` to the enabled CloudWatch Logs exports.

```
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 from this operation looks something like the following.

```
{
  "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
}
```

Disabling the Amazon DocumentDB Profiler

To disable the profiler, you disable both the `profiler` parameter and the export of `profiler` logs to CloudWatch Logs.

Disabling the Profiler

You can disable the `profiler` parameter using either the AWS Management Console or AWS CLI, as follows.

Using the AWS Management Console

The following procedure uses the AWS Management Console to disable Amazon DocumentDB profiler.

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Parameter groups**. Then choose the name of the cluster parameter group that you want to disable the profiler on.
3. In the resulting **Cluster parameters** page, select the button to the left of the profiler parameter and choose **Edit**.
4. In the **Modify profiler** dialog box, choose disabled in the list.
5. Choose **Modify cluster parameter**.

Using the AWS CLI

To disable profiler on a cluster using the AWS CLI, modify the cluster as follows.

```
aws docdb modify-db-cluster-parameter-group \
--db-cluster-parameter-group-name sample-parameter-group \
--parameters ParameterName=profiler,ParameterValue=disabled,ApplyMethod=immediate
```

Disabling Profiler Logs Export

You can disable exporting profiler logs to CloudWatch Logs by using either the AWS Management Console or AWS CLI, as follows.

Using the AWS Management Console

The following procedure uses the AWS Management Console to disable Amazon DocumentDB exporting logs to CloudWatch.

1. Open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Clusters**. Choose the button to the left of the name of the cluster for which you want to disable exporting logs.
3. On the **Actions** menu, choose **Modify**.
4. Scroll down to the **Log exports** section and unselect **Profiler logs**.
5. Choose **Continue**.
6. Review your changes, and then choose when you want this change applied to your cluster:
 - **Apply during the next scheduled maintenance window**
 - **Apply immediately**
7. Choose **Modify cluster**.

Using the AWS CLI

The following code modifies the cluster `sample-cluster` and disables CloudWatch profiler logs.

Example

For Linux, macOS, or Unix:

```
aws docdb modify-db-cluster \
--db-cluster-identifier sample-cluster \
```

```
--cloudwatch-logs-export-configuration '{"DisableLogTypes":["profiler"]}'
```

For Windows:

```
aws docdb modify-db-cluster ^
--db-cluster-identifier sample-cluster ^
--cloudwatch-logs-export-configuration '{"DisableLogTypes":["profiler"]}'
```

Output from this operation looks something like the following.

```
{
  "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
  }
}
```

Accessing Your Amazon DocumentDB Profiler Logs

Follow these steps to access your profile logs on Amazon CloudWatch.

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. Make sure that you are in the same Region as your Amazon DocumentDB cluster.
3. In the navigation pane, choose **Logs**.
4. To find the profiler logs for your cluster, in the list, choose /aws/docdb/*yourClusterName*/profiler.

The profile logs for each of your instances are available under each of the respective instance names.

Common Queries

The following are some common queries you can use to analyze your profiled commands. For more information about CloudWatch Logs Insights, see [Analyzing Log Data with CloudWatch Logs Insights](#) and [Sample Queries](#).

Get the 10 Slowest Operations on a Specified Collection

```
filter ns="test.foo" | sort millis desc | limit 10
```

Get All the Update Operations on a Collection That Took More Than 60 ms

```
filter millis > 60 and op = "update"
```

Get the 10 Slowest Operations in the Last Month

```
sort millis desc | limit 10
```

Get All the Queries with a COLLSCAN Plan Summary

```
filter planSummary="COLLSCAN"
```

Monitoring with Performance Insights

Performance Insights adds to the existing Amazon DocumentDB monitoring features to illustrate your cluster performance and help you analyze any issues that affect it. With the Performance Insights dashboard, you can visualize the database load and filter the load by waits, query statements, hosts, or application.

How is it useful?

- Visualize database performance — Visualize the load to determine when and where the load is on the database

- Determine what is causing load on database — Determine which queries, hosts, and applications are contributing to the load on your instance
- Determine when there is load on your database — Zoom in on the Performance Insights dashboard to focus on specific events or zoom out to look at trends across a larger time span
- Alert on database load — Access new database load metrics automatically from CloudWatch where you can monitor the DB load metrics alongside other DocumentDB metrics and set alerts on them

What are the limitations of Amazon DocumentDB Performance Insights?

- Performance Insights in AWS GovCloud (US-West) region are not yet available
- Performance Insights for DocumentDB retains up to 7 days of performance data
- Queries longer than 1024kb are not aggregated in Performance Insights

Topics

- [Performance Insights concepts \(p. 435\)](#)
- [Enabling and disabling Performance Insights \(p. 437\)](#)
- [Configuring access policies for Performance Insights \(p. 439\)](#)
- [Analyzing metrics with the Performance Insights dashboard \(p. 443\)](#)
- [Retrieving metrics with the Performance Insights API \(p. 456\)](#)
- [Amazon CloudWatch metrics for Performance Insights \(p. 466\)](#)
- [Performance Insights for counter metrics \(p. 467\)](#)

Performance Insights concepts

Topics

- [Average active sessions \(p. 435\)](#)
- [Dimensions \(p. 436\)](#)
- [Max vCPU \(p. 437\)](#)

Average active sessions

Database load (DB load) measures the level of activity in your database. The key metric in Performance Insights is DB Load, which is collected every second. The unit for the DBLoad metric is the *Average Active Sessions (AAS)* for a DocumentDB instance.

An *active* session is a connection that has submitted work to the DocumentDB instance and is waiting for a response. For example, if you submit a query to a DocumentDB instance, the database session is active while the instance is processing the query.

To obtain the average active sessions, Performance Insights samples the number of sessions concurrently running a query. The AAS is the total number of sessions divided by the total number of samples. The following table shows five consecutive samples of a running query.

Sample	Number of sessions running query	AAS	Calculation
1	2	2	2 sessions / 1 sample
2	0	1	2 sessions / 2 samples

Sample	Number of sessions running query	AAS	Calculation
3	4	2	6 sessions / 3 samples
4	0	1.5	6 sessions / 4 samples
5	4	2	10 sessions / 5 samples

In the preceding example, the DB Load for the time interval from 1-5 is 2 AAS. An increase in DB load means that, on average, more sessions are running on the database.

Dimensions

The DB Load metric is different from the other time-series metrics because you can break it into subcomponents called dimensions. You can think of dimensions as categories for the different characteristics of the DB Load metric. When you are diagnosing performance issues, the most useful dimensions are **wait states** and **top query**.

wait states

A *wait state* causes a query statement to wait for a specific event to happen before it can continue running. For example, a query statement might wait until a locked resource is unlocked. By combining DB Load with wait states, you can get a complete picture of the session state. Here are various DocumentDB wait states:

DocumentDB wait state	Wait State Description
Latch	The Latch wait state occurs when the session is waiting to page the buffer pool. Frequent paging in and out of the buffer pool can happen more often when there are frequent large queries being processed by the system, collection scans, or when the buffer pool is too small to handle the working set.
CPU	The CPU wait state occurs when the session is waiting on CPU.
CollectionLock	The CollectionLock wait state occurs when the session is waiting to acquire a lock on the collection. These events occur when there are DDL operations on the collection.
DocumentLock	The DocumentLock wait state occurs when the session is waiting to acquire a lock on a document. High number of concurrent writes to the same document will contribute to more DocumentLock wait states on that document.
SystemLock	The SystemLock wait state occurs when the session is waiting on the system. This can occur when there are frequent long running queries, long running transactions, or high concurrency on the system.

DocumentDB wait state	Wait State Description
IO	The IO wait state occurs when the session waiting on IO to complete.
BufferLock	The BufferLock wait state occurs when the session is waiting to acquire a lock on a shared page in the buffer. BufferLock wait states can be prolonged if other processes are holding open cursors on the requested pages.
LowMemThrottle	The LowMemThrottle wait state occurs when the session is waiting due to heavy memory pressure on the Amazon DocumentDB instance. If this state persists for a long time, consider scaling up the instance to provide additional memory. For more information, see Resource Governor .
BackgroundActivity	The BackgroundActivity wait state occurs when the session is waiting on internal system processes.
Other	The Other wait state is an internal wait state. If this state persists for a long time, consider terminating this query. For more information, see How Do I Find and Terminate Long Running or Blocked Queries?

Top queries

Whereas wait states show bottlenecks, top queries show which queries are contributing the most to DB load. For example, many queries might be currently running on the database, but a single query might consume 99% of the DB load. In this case, the high load might indicate a problem with the query.

Max vCPU

In the dashboard, the **Database load** chart collects, aggregates, and displays session information. To see whether active sessions are exceeding the maximum CPU, look at their relationship to the **Max vCPU** line. The **Max vCPU** value is determined by the number of vCPU (virtual CPU) cores for your DocumentDB instance.

If the DB load is often above the **Max vCPU** line, and the primary wait state is CPU, the CPU is overloaded. In this case, you might want to throttle connections to the instance, tune any queries with a high CPU load, or consider a larger instance class. High and consistent instances of any wait state indicate that there might be bottlenecks or resource contention issues to resolve. This can be true even if the DB load doesn't cross the **Max vCPU** line.

Enabling and disabling Performance Insights

To use Performance Insights, enable it on your DB instance. You can disable it later if necessary. Enabling and disabling Performance Insights doesn't cause downtime, a reboot, or a failover.

The Performance Insights agent consumes limited CPU and memory on the DB host. When the DB load is high, the agent limits the performance impact by collecting data less frequently.

Enabling Performance Insights when creating a cluster

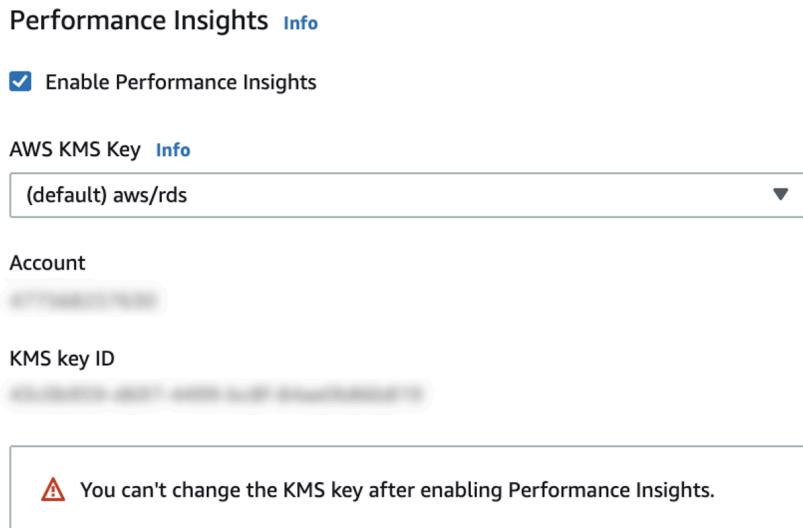
In the console, you can enable or disable Performance Insights when you create or modify a new DB instance.

Using the AWS Management Console

In the console, you can enable Performance Insights when you create a DocumentDB cluster. When you create a new DocumentDB cluster, enable Performance Insights by choosing **Enable Performance Insights** in the **Performance Insights** section.

Console instructions

1. To create a cluster, follow the instructions for [Creating an Amazon DocumentDB cluster](#).
2. Select **Enable Performance Insights** in the Performance Insights section.



Note

The Performance Insights data retention period will be seven days.

AWS KMS key — Specify your AWS KMS key. Performance Insights encrypts all potentially sensitive data using your AWS KMS key. Data is encrypted in flight and at rest. For more information, see [Configuring an AWS AWS KMS policy for Performance Insights](#).

Enabling and disabling when modifying an instance

You can modify a DB instance to enable or disable Performance Insights using the console or AWS CLI.

Using the AWS Management Console

Console instructions

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. Choose **Clusters**.
3. Choose a DB instance, and choose **Modify**.
4. In the Performance Insights section, choose either **Enable Performance Insights** or **Disable Performance Insights**.

Note

If you choose **Enable Performance Insights**, you can specify your AWS AWS KMS key. Performance Insights encrypts all potentially sensitive data using your AWS KMS key. Data is encrypted in flight and at rest. For more information, see [Encrypting Amazon DocumentDB Data at Rest](#).

5. Choose **Continue**.
6. For **Scheduling of Modifications**, choose **Apply immediately**. If you choose **Apply during the next scheduled maintenance window**, your instance ignores this setting and enables Performance Insights immediately.
7. Choose **Modify instance**.

Using the AWS CLI

When you use the `create-db-instance` or `modify-db-instance` AWS AWS CLI commands, you can enable Performance Insights by specifying `--enable-performance-insights`, or disable it by specifying `--no-enable-performance-insights`.

The following procedure describes how to enable or disable Performance Insights for a DB instance using the AWS AWS CLI.

AWS AWS CLI instructions

Call the `modify-db-instance` AWS AWS CLI command and provide the following values:

- `--db-instance-identifier` — The name of the DB instance
- `--enable-performance-insights` to enable or `--no-enable-performance-insights` to disable

Example

The following example enables Performance Insights for `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
```

Configuring access policies for Performance Insights

To access Performance Insights, you must have the appropriate permissions from AWS Identity and Access Management (IAM). You have the following options for granting access:

- Attach the `AmazonRDSPerformanceInsightsReadOnly` managed policy to an IAM user or role.
- Create a custom IAM policy and attach it to an IAM user or role.

Also, if you specified a customer managed key when you turned on Performance Insights, make sure that users in your account have the `kms:Decrypt` and `kms:GenerateDataKey` permissions on the KMS key.

Note

For encryption-at-rest with AWS KMS keys and security groups management, Amazon DocumentDB leverages operational technology that is shared with [Amazon RDS](#).

Attaching the `AmazonRDSPerformanceInsightsReadOnly` policy to an IAM principal

`AmazonRDSPerformanceInsightsReadOnly` is an AWS-managed policy that grants access to all read-only operations of the Amazon DocumentDB Performance Insights API. Currently, all operations in this API are read-only. If you attach `AmazonRDSPerformanceInsightsReadOnly` to an IAM user or role, the recipient can use Performance Insights with other console features.

Creating a custom IAM policy for Performance Insights

For users who don't have the `AmazonRDSPerformanceInsightsReadOnly` policy, you can grant access to Performance Insights by creating or modifying a user-managed IAM policy. When you attach the policy to an IAM user or role, the recipient can use Performance Insights.

To create a custom policy

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Policies**.
3. Choose **Create policy**.
4. On the **Create Policy** page, choose the JSON tab.
5. Copy and paste the following text, replacing `us-east-1` with the name of your AWS Region and `111122223333` with your customer account number.

```
{  
    "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. Choose **Review policy**.
7. Provide a name for the policy and optionally a description, and then choose **Create policy**.

You can now attach the policy to an IAM user or role. The following procedure assumes that you already have an IAM user available for this purpose.

To attach the policy to an IAM user

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Users**.
3. Choose an existing user from the list.

Important

To use Performance Insights, make sure that you have access to Amazon DocumentDB in addition to the custom policy. For example, the **AmazonDocDBReadOnlyAccess** predefined policy provides read-only access to Amazon DocDB. For more information, see [Managing access using policies](#).

4. On the **Summary** page, choose **Add permissions**.
5. Choose **Attach existing policies directly**. For **Search**, type the first few characters of your policy name, as shown following.

	Policy name ▾	Type	Used as
<input type="checkbox"/>	PerformanceInsightsCustomPolicy	Customer managed	None

6. Choose your policy, and then choose **Next: Review**.
7. Choose **Add permissions**.

Configuring an AWS KMS policy for Performance Insights

Performance Insights uses an AWS KMS key to encrypt sensitive data. When you enable Performance Insights through the API or the console, you have the following options:

- Choose the default AWS managed key.

Amazon DocumentDB uses the AWS managed key for your new DB instance. Amazon DocumentDB creates an AWS managed key for your AWS account. Your AWS account has a different AWS managed key for Amazon DocumentDB for each AWS Region.

- Choose a customer managed key.

If you specify a customer managed key, users in your account that call the Performance Insights API need the `kms:Decrypt` and `kms:GenerateDataKey` permissions on the KMS key. You can configure these permissions through IAM policies. However, we recommend that you manage these permissions through your KMS key policy. For more information, see [Using key policies in AWS KMS](#).

Example

The following sample key policy shows how to add statements to your KMS key policy. These statements allow access to Performance Insights. Depending on how you use the AWS KMS, you might want to change some restrictions. Before adding statements to your policy, remove all comments.

```
{  
    "Version" : "2012-10-17",  
    "Id" : "your-policy",  
    "Statement" : [ {  
        //This represents a statement that currently exists in your policy.  
    }  
    ....  
    //Starting here, add new statement to your policy for Performance Insights.  
    //We recommend that you add one new statement for every RDS/DocumentDB instance  
    {  
        "Sid" : "Allow viewing RDS Performance Insights",  
        "Effect": "Allow",  
        "Principal": {  
            "AWS": [  
                //One or more principals allowed to access Performance Insights  
                "arn:aws:iam::444455556666:role/Role1"  
            ]  
        },  
        "Action": [  
            "kms:Decrypt",  
            "kms:GenerateDataKey"  
        ],  
        "Resource": "*",  
        "Condition" :{  
            "StringEquals" : {  
                //Restrict access to only RDS APIs (including Performance Insights).  
                //Replace *region* with your AWS Region.  
                //For example, specify us-west-2.  
                "kms:ViaService" : "rds.*region*.amazonaws.com"  
            },  
            "ForAnyValue:StringEquals": {  
                //Restrict access to only data encrypted by Performance Insights.  
                "kms:EncryptionContext:aws:pi:service": "rds",  
                "kms:EncryptionContext:service": "pi",  
  
                //Restrict access to a specific DocDB instance.  
                //The value is a DbiResourceId.  
                "kms:EncryptionContext:aws:rds:db-id": "db-AAAAABBBBCCCCDDDDDEEEEE"  
            }  
        }  
    }  
}
```

```
}
```

Analyzing metrics with the Performance Insights dashboard

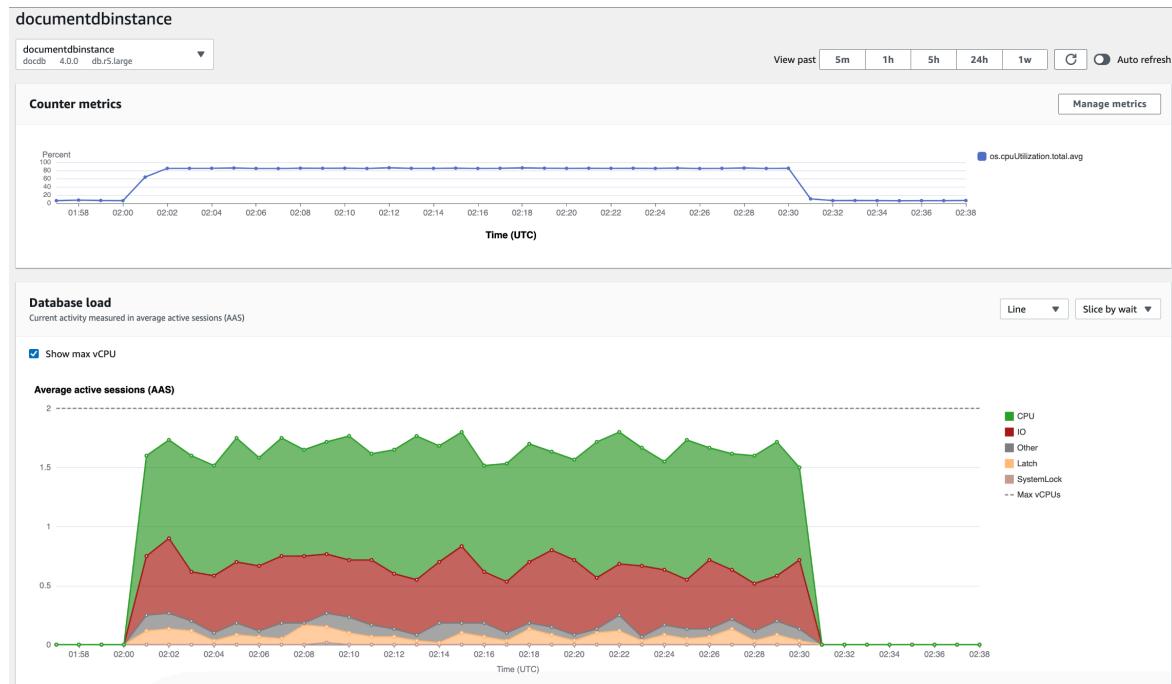
The Performance Insights dashboard contains database performance information to help you analyze and troubleshoot performance issues. On the main dashboard page, you can view information about the database load (DB load). You can "slice" DB load by dimensions such as wait states or query.

Topics

- [Overview of the Performance Insights dashboard \(p. 443\)](#)
- [Opening the Performance Insights dashboard \(p. 448\)](#)
- [Analyzing database load by wait states \(p. 449\)](#)
- [Overview of the Top queries tab \(p. 451\)](#)
- [Zooming in on the database load chart \(p. 455\)](#)

Overview of the Performance Insights dashboard

The dashboard is the easiest way to interact with Performance Insights. The following example shows the dashboard for an Amazon DocumentDB instance. By default, the Performance Insights dashboard shows data for the last hour.



The dashboard is divided into the following parts:

1. **Counter metrics** – Shows data for specific performance counter metrics.
2. **Database load** – Shows how the DB load compares to DB instance capacity as represented by the **Max vCPUs** line.

3. **Top dimensions** – Shows the top dimensions contributing to DB load. These dimensions include waits, queries, hosts, databases, and applications.

Topics

- [Counter metrics chart \(p. 444\)](#)
- [Database load chart \(p. 445\)](#)
- [Top dimensions table \(p. 446\)](#)

Counter metrics chart

With counter metrics, you can customize the Performance Insights dashboard to include up to 10 additional graphs. These graphs show a selection of dozens of operating system metrics. You can correlate this information with DB load to help identify and analyze performance problems.

The **Counter metrics** chart displays data for performance counters.



To change the performance counters, choose **Manage metrics**. You can select multiple **OS metrics** as shown in the following screenshot. To see details for any metric, hover over the metric name.

Select metrics shown on the graph X

Check the metrics that you want to see on the Performance Insights dashboard.

Find metrics

OS metrics (4) Clear all selections

general

numVCpus

cpuUtilization

idle system total
 user wait

loadAverageMinute

fifteen five one

memory

active buffers cached
 dirty free inactive

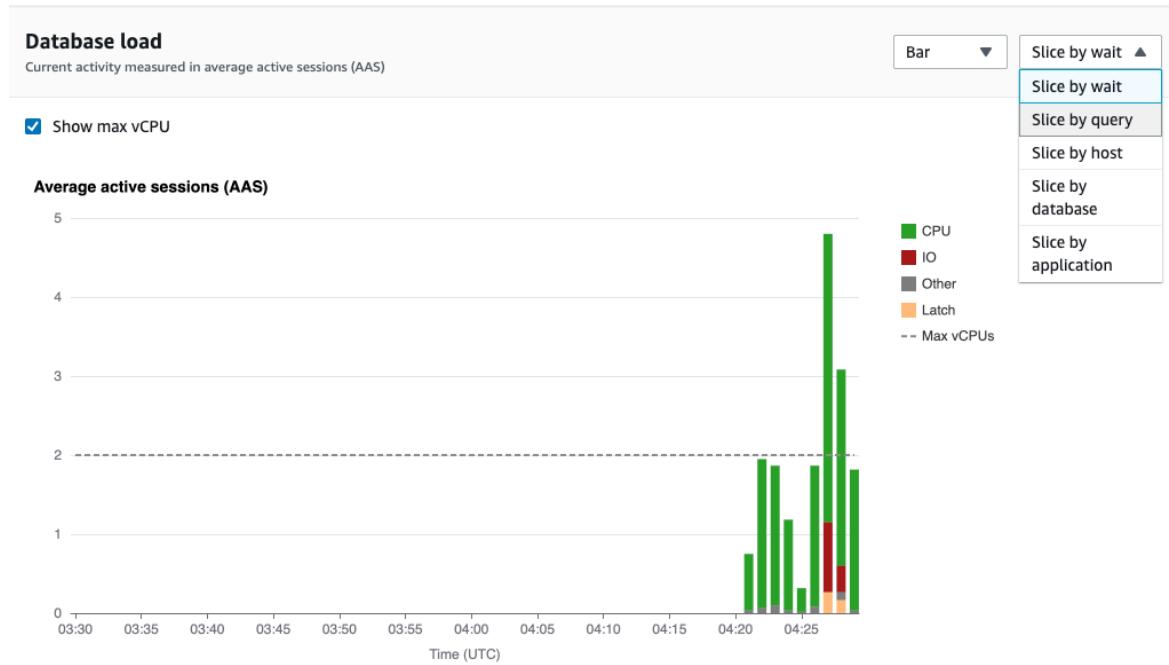
Database load chart

The **Database load** chart shows how the database activity compares to instance capacity as represented by the **Max vCPU** line. By default, the stacked line chart represents DB load as average active sessions per unit of time. The DB load is sliced (grouped) by wait states.



DB load sliced by dimensions

You can choose to display load as active sessions grouped by any supported dimensions. The following image shows the dimensions for the Amazon DocumentDB instance.



DB load details for a dimension item

To see details about a DB load item within a dimension, hover over the item name. The following image shows details for a query statement.



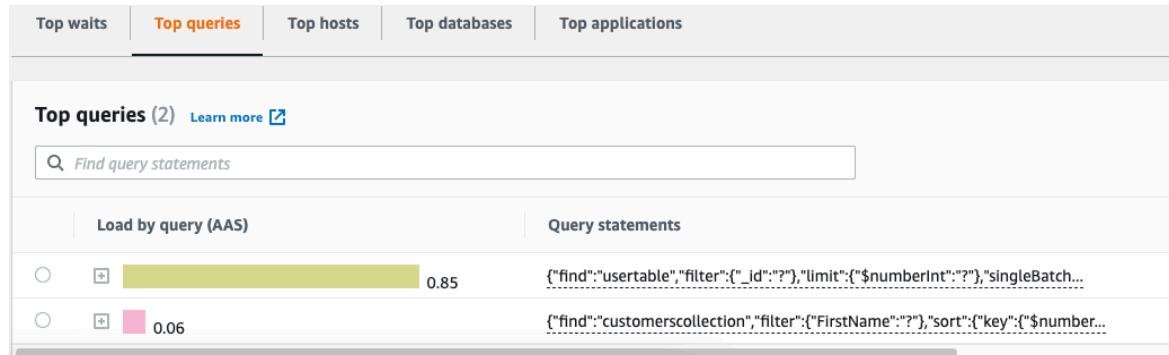
To see details for any item for the selected time period in the legend, hover over that item.



Top dimensions table

The **Top dimensions table** slices DB load by different dimensions. A dimension is a category or "slice by" for different characteristics of DB load. If the dimension is query, **Top queries** shows the query statements that contribute most to the DB load.

Choose any of the following dimension tabs.



The following table provides a brief description of each tab.

Description	Top waits	Top queries	Top hosts	Top databases
for which the database backend is waiting	for which the database backend is waiting	statements that are currently running	IP and port of the connected client	of the database to which the client is connected
for which the database backend is waiting	for which the database backend is waiting	statements that are currently running	IP and port of the connected client	of the database to which the client is connected
for which the database backend is waiting	for which the database backend is waiting	statements that are currently running	IP and port of the connected client	of the database to which the client is connected
for which the database backend is waiting	for which the database backend is waiting	statements that are currently running	IP and port of the connected client	of the database to which the client is connected

Description

The applications of the application that is connected to the database

To learn how to analyze queries by using the **Top queries** tab, see [Overview of the Top queries tab \(p. 451\)](#).

Opening the Performance Insights dashboard

To view the Performance Insights dashboard in the AWS Management Console, use the following steps:

1. Open the Performance Insights console at <https://console.aws.amazon.com/docdb/>.
2. Choose a DB instance. The Performance Insights dashboard is shown for that Amazon DocumentDB instance.

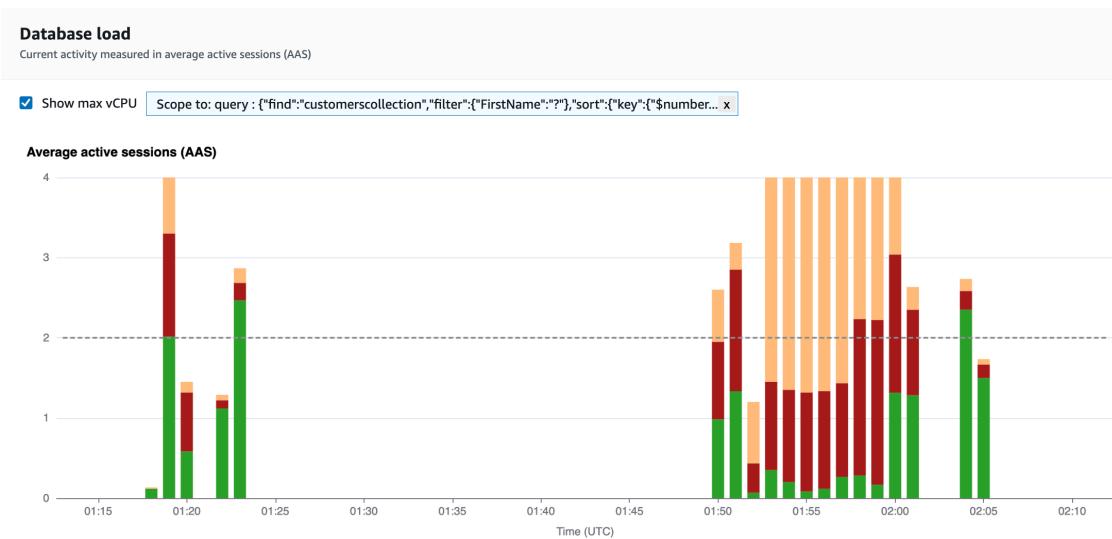
For Amazon DocumentDB instances with Performance Insights enabled, you can also reach the dashboard by choosing the **Sessions** item in the list of instances. Under **Current activity**, the **Sessions** item shows the database load in average active sessions over the last five minutes. The bar graphically shows the load. When the bar is empty, the instance is idle. As the load increases, the bar fills with blue. When the load passes the number of virtual CPUs (vCPUs) on the instance class, the bar turns red, indicating a potential bottleneck.

Clusters (1)								<input type="radio"/> Group Resources	Actions ▾	Create	
		Cluster identifier		Role	Engine version	Region & AZ	Status	CPU	Current activity		
<input type="checkbox"/>	<input type="checkbox"/>	documentdbinstance		Regional cluster	4.0.0	ap-south-1	available	-			
<input type="checkbox"/>	<input type="checkbox"/>	documentdbinstance		Primary instance	4.0.0	ap-south-1c	available	<div style="width: 84.99%;">84.99%</div>	5 Connections		
<input type="checkbox"/>	<input type="checkbox"/>	documentdbinstance2		Replica instance	4.0.0	ap-south-1b	available	<div style="width: 15.37%;">15.37%</div>	2 Connections		
<input type="checkbox"/>	<input type="checkbox"/>	documentdbinstance3		Replica instance	4.0.0	ap-south-1a	available	<div style="width: 14.84%;">14.84%</div>	2 Connections		

3. (Optional) Choose a different time interval by selecting a button in the upper right. For example, to change the interval to 1 hour, select **1h**.



In the following screenshot, the DB load interval is 1 hour.



4. To refresh your data automatically, enable **Auto refresh**.



The Performance Insight dashboard automatically refreshes with new data. The refresh rate depends on the amount of data displayed:

- 5 minutes refreshes every 5 seconds.
- 1 hour refreshes every minute.
- 5 hours refreshes every minute.
- 24 hours refreshes every 5 minutes.
- 1 week refreshes every hour.

Analyzing database load by wait states

If the **Database load (DB load)** chart shows a bottleneck, you can find out where the load is coming from. To do so, look at the top load items table below the **Database load** chart. Choose a particular item, like a query or an application, to drill down into that item and see details about it.

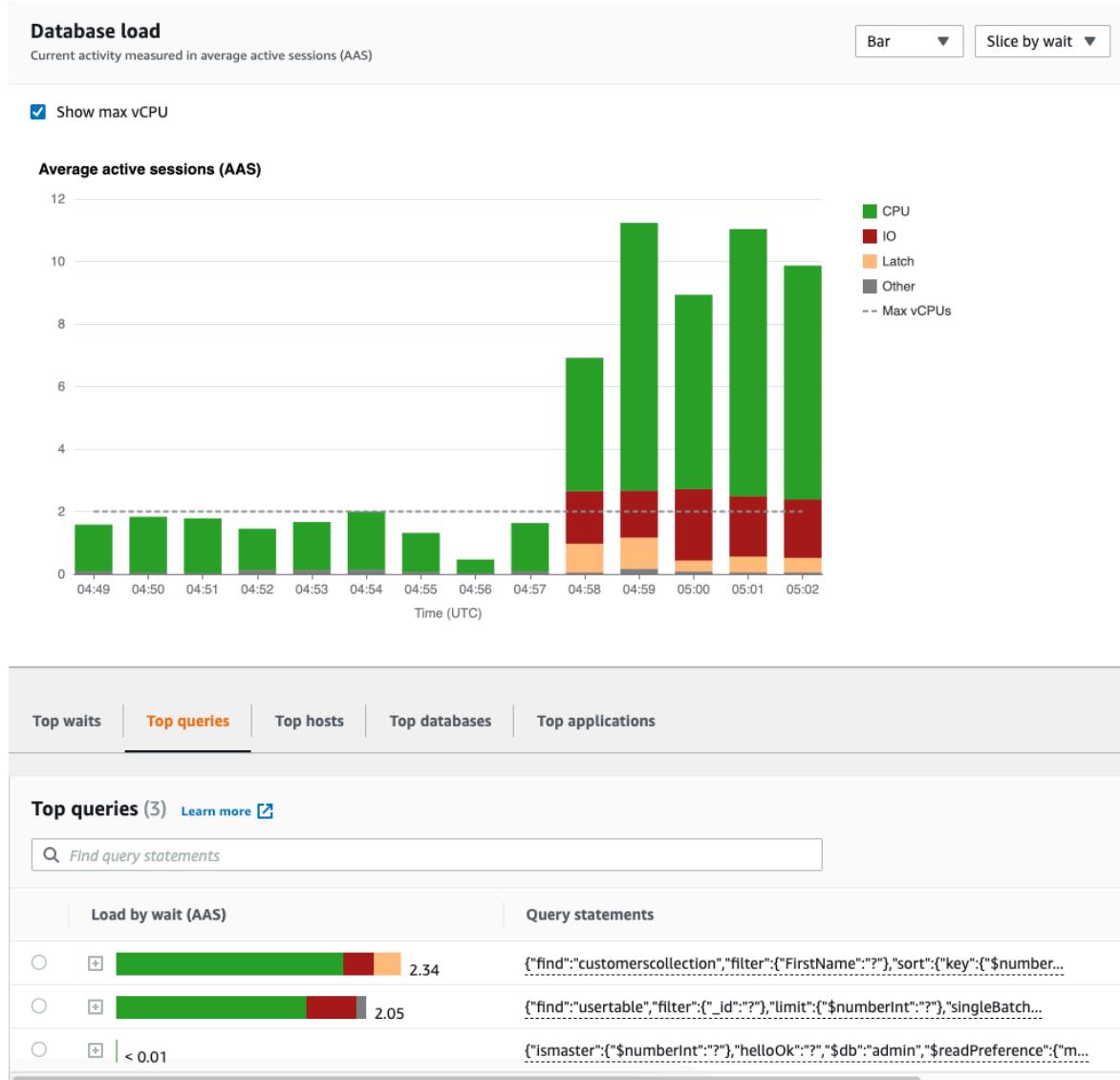
DB load grouped by waits and top queries typically provides the most insight into performance issues. DB load grouped by waits shows if there are any resource or concurrency bottlenecks in the database. In this case, the **Top queries** tab of the top load items table shows which queries are driving that load.

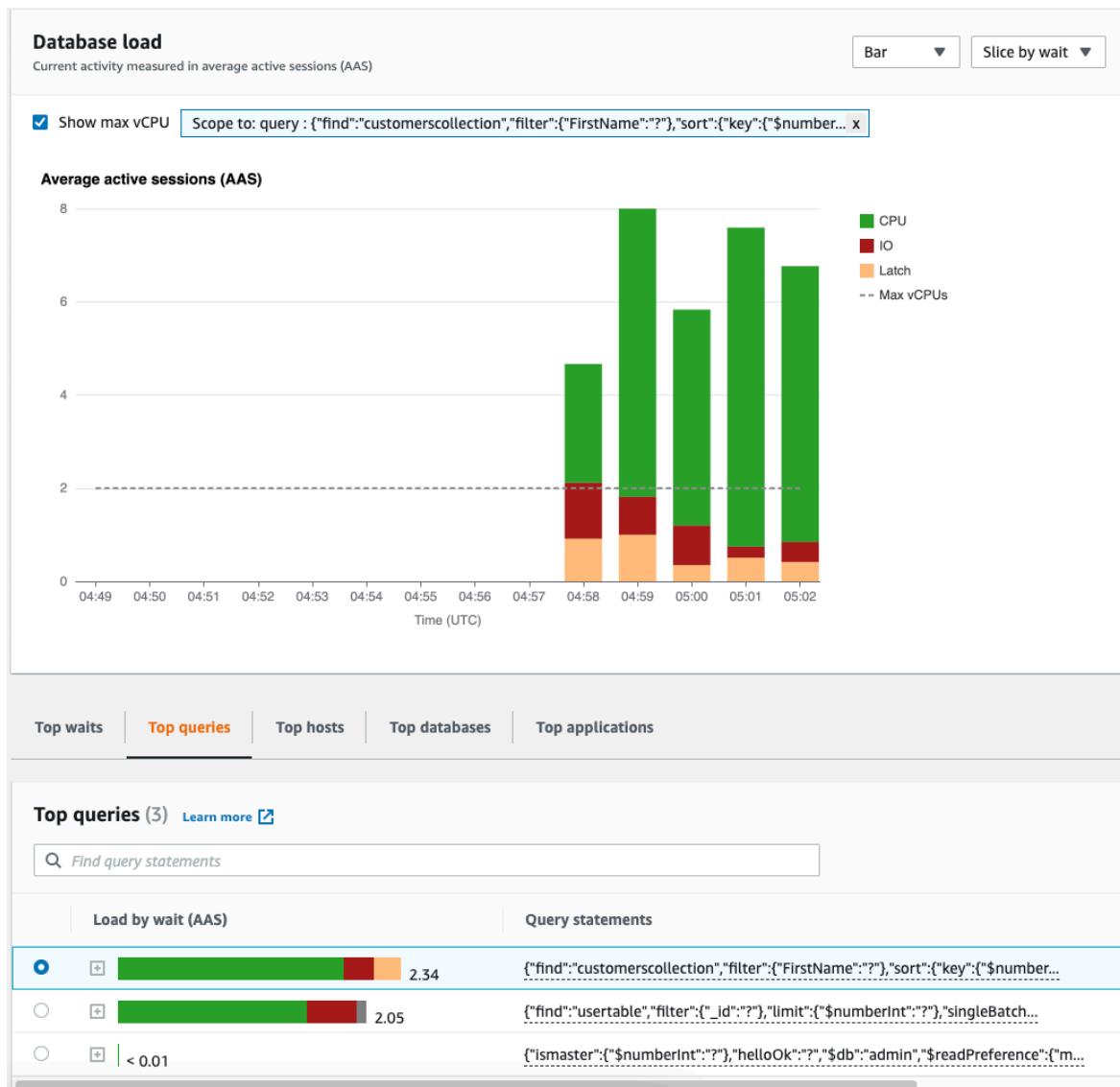
Your typical workflow for diagnosing performance issues is as follows:

1. Review the **Database load** chart and see if there are any incidents of database load exceeding the **Max CPU** line.
2. If there is, look at the **Database load** chart and identify which wait state or states are primarily responsible.
3. Identify the digest queries causing the load by seeing which of the queries the **Top queries** tab on the top load items table are contributing most to those wait states. You can identify these by the **Load by Wait (AAS)** column.
4. Choose one of these digest queries in the **Top queries** tab to expand it and see the child queries that it is composed of.

You can also see which hosts or applications are contributing the most load by selecting **Top hosts** or **Top applications**, respectively. Application names are specified in the connection string to the Amazon DocumentDB instance. Unknown indicates that the application field was not specified.

For example, in the following dashboard, **CPU** waits account for most of the DB load. Selecting the top query under **Top queries** will scope the Database load chart to focus on the most load that is being contributed by the select query.





Overview of the Top queries tab

By default, the **Top query** tab shows the queries that are contributing the most to DB load. You can analyze the query text to help tune your queries.

Topics

- [Query digests \(p. 452\)](#)
- [Load by waits \(AAS\) \(p. 452\)](#)
- [Viewing detailed query information \(p. 453\)](#)
- [Accessing statement query text \(p. 454\)](#)
- [Viewing and downloading statement query text \(p. 454\)](#)

Query digests

A *query digest* is a composite of multiple actual queries that are structurally similar but might have different literal values. The digest replaces hardcoded values with a question mark. For example, a query digest might look like this:

```
{"find":"customerscollection","filter":{"FirstName":"?"}, "sort":{"key": "$numberInt":"?"}, "limit":{"$numberInt":"?"}}
```

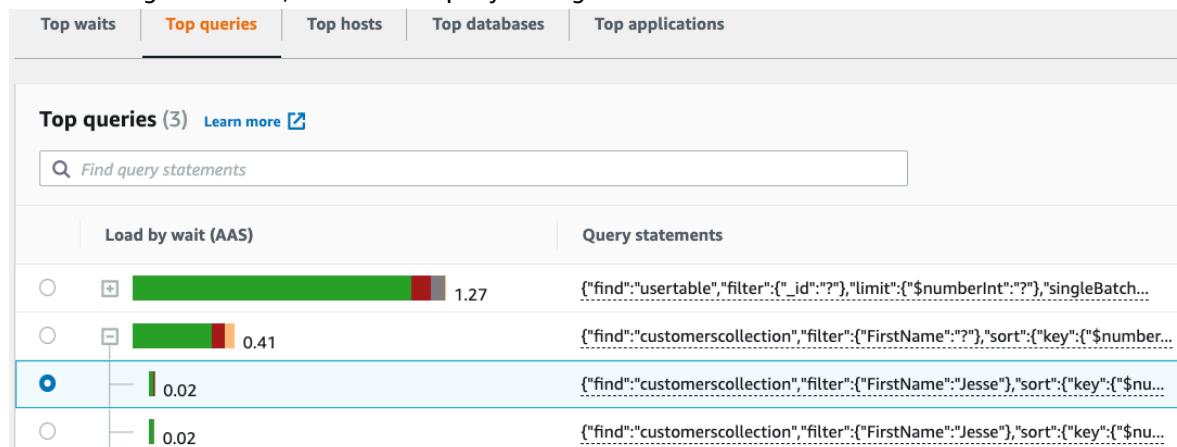
This digest might include the following child queries:

```
{"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"}}
```

To see the literal query statements in a digest, select the query, and then choose the plus symbol (+). In the following screenshot, the selected query is a digest.

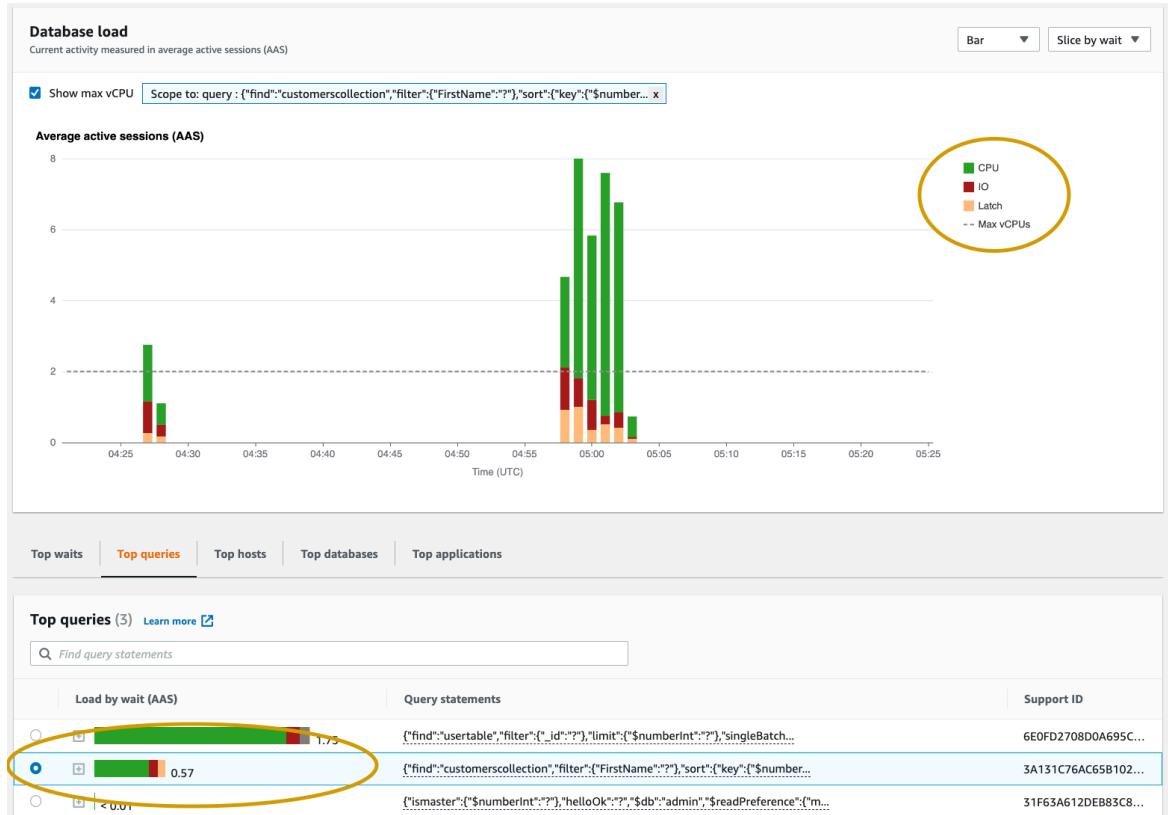


Note

A query digest groups similar query statements, but does not redact sensitive information.

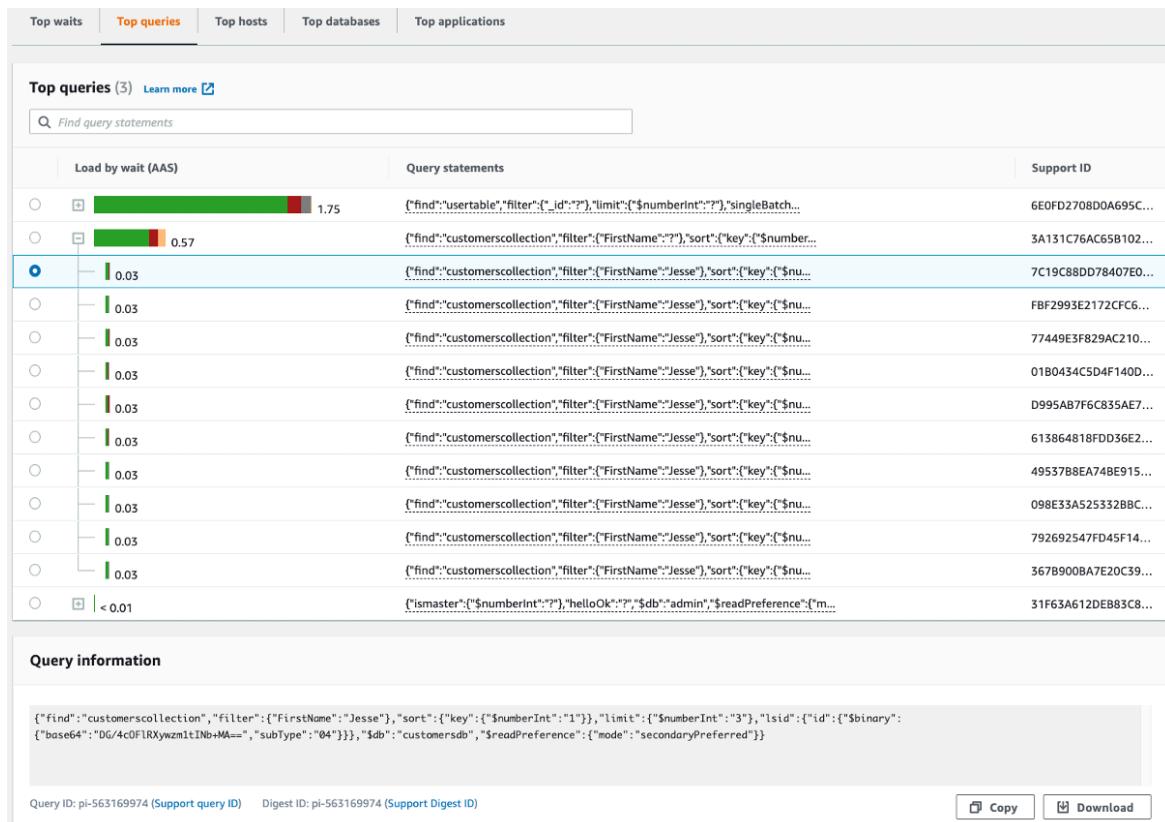
Load by waits (AAS)

In **Top queries**, the **Load by waits (AAS)** column illustrates the percentage of the database load associated with each top load item. This column reflects the load for that item by whatever grouping is currently selected in the **DB load chart**. For example, you might group the **DB load chart** by wait states. In this case, the **DB Load by Waits** bar is sized, segmented, and color-coded to show how much of a given wait state that query is contributing to. It also shows which wait states are affecting the selected query.



Viewing detailed query information

In the **Top query** table, you can open a *digest statement* to view its information. The information appears in the bottom pane.



The following types of identifiers (IDs) are associated with query statements:

- Support query ID** – A hash value of the query ID. This value is only for referencing a query ID when you are working with AWS Support. AWS Support doesn't have access to your actual query IDs and query text.
- Support digest ID** – A hash value of the digest ID. This value is only for referencing a digest ID when you are working with AWS Support. AWS Support doesn't have access to your actual digest IDs and query text.

Accessing statement query text

By default, each row in the **Top queries** table shows 500 bytes of query text for each query statement. When a digest statement exceeds 500 bytes, you can view more text by opening the statement in the Performance Insights dashboard. In this case, the maximum length for the displayed query is 1 KB. If you view a full query statement, you can also choose **Download**.

Viewing and downloading statement query text

In the Performance Insights dashboard, you can view or download query text.

To view more query text in the Performance Insights dashboard

- Open the Amazon DocumentDB console at: <https://console.aws.amazon.com/docdb/>
- In the navigation pane, choose **Performance Insights**.
- Choose a DB instance. The Performance Insights dashboard is displayed for that DB instance.

Query statements with text larger than 500 bytes will look like the following image:

Top queries (3) Learn more			
Load by wait (AAS)		Query statements	
<input type="radio"/>	<input checked="" type="checkbox"/>	1.75 {"find": "usertable", "filter": {"_id": "?"}, "limit": {"\$numberInt": "?"}, "singleBatch": true}	6E0FD2708D0A695C...
<input type="radio"/>	<input checked="" type="checkbox"/>	0.57 {"find": "customerscollection", "filter": {"FirstName": "?"}, "sort": {"key": {"\$numberInt": "?"}, "order": "asc"}, "batchSize": 1}	3A131C76AC65B102...
<input checked="" type="radio"/>	<input type="checkbox"/>	0.03 {"find": "customerscollection", "filter": {"FirstName": "Jesse"}, "sort": {"key": {"\$numberInt": "?"}, "order": "asc"}, "batchSize": 1}	7C19C88DD78407E0...
<input type="radio"/>	<input type="checkbox"/>	0.03 {"find": "customerscollection", "filter": {"FirstName": "Jesse"}, "sort": {"key": {"\$numberInt": "?"}, "order": "asc"}, "batchSize": 1}	FBF2993E2172FC0...

4. Examine the query information section to view more of the query text.

Query information

```
{"find": "customerscollection", "filter": {"FirstName": "Jesse"}, "sort": {"key": {"$numberInt": "1"}}, "limit": {"$numberInt": "3"}, "lsid": {"id": {"$binary": "DG/4c0FlRXywm1tNB+MA==", "subType": "04"}}, "$db": "customersdb", "$readPreference": {"mode": "secondaryPreferred"}}
```

Query ID: pl-563169974 ([Support query ID](#)) Digest ID: pl-563169974 ([Support Digest ID](#))

[Copy](#) [Download](#)

The Performance Insights dashboard can display up to 1 KB for each full query statement.

Note

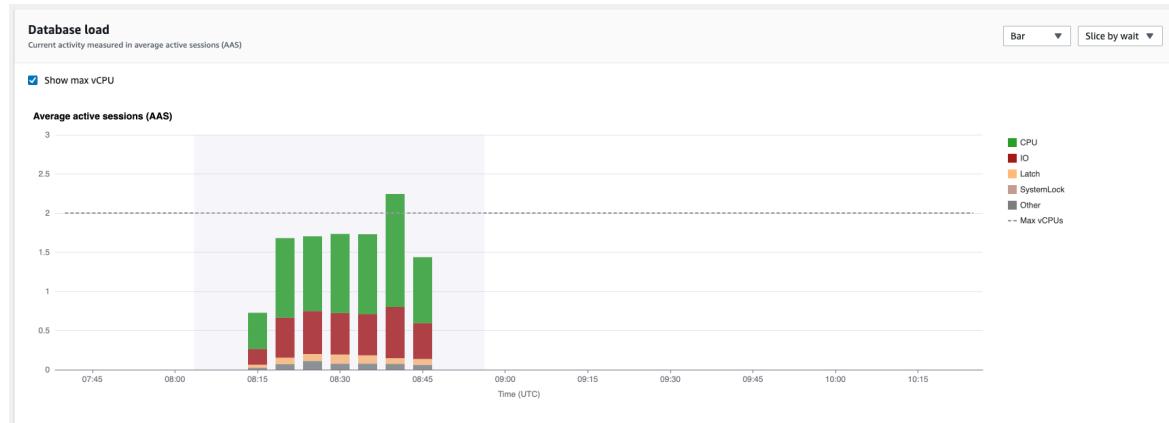
To copy or download the query statement, disable any pop-up blockers.

Zooming in on the database load chart

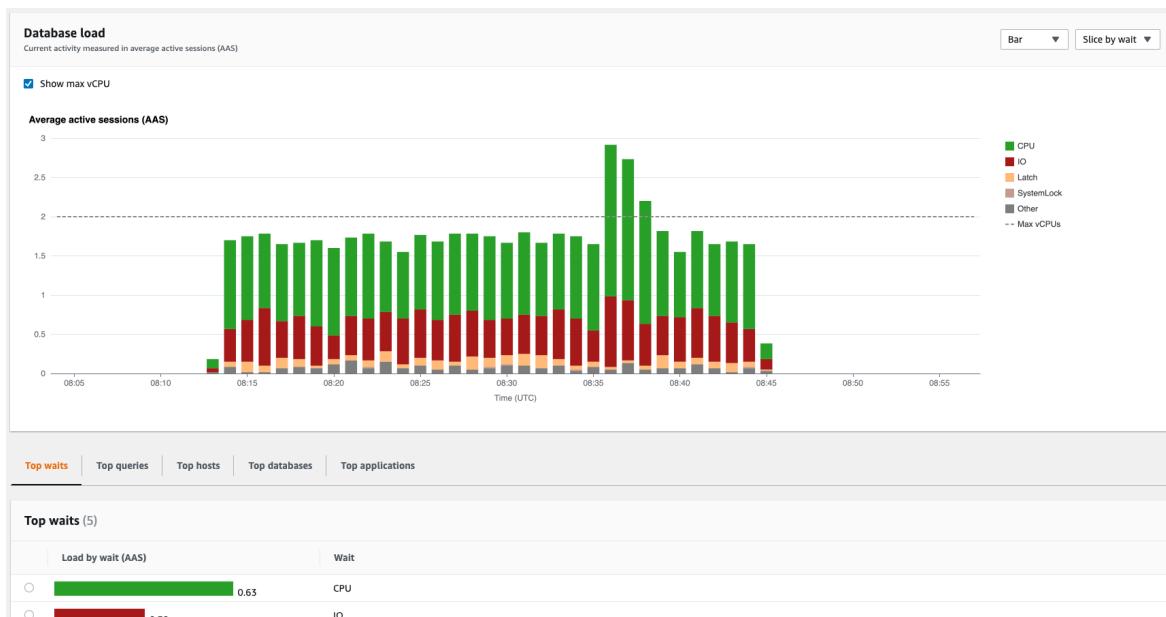
You can use other features of the Performance Insights user interface to help analyze performance data.

Click-and-Drag Zoom In

In the Performance Insights interface, you can choose a small portion of the load chart and zoom in on the detail.



To zoom in on a portion of the load chart, choose the start time and drag to the end of the time period you want. When you do this, the selected area is highlighted. When you release the mouse, the load chart zooms in on the selected area, and the **Top items** table is recalculated.



Retrieving metrics with the Performance Insights API

When Performance Insights is enabled, the API provides visibility into instance performance. Amazon CloudWatch Logs provides the authoritative source for vended monitoring metrics for AWS services.

Performance Insights offers a domain-specific view of database load measured as average active sessions (AAS). This metric appears to API consumers as a two-dimensional time-series dataset. The time dimension of the data provides DB load data for each time point in the queried time range. Each time point decomposes overall load in relation to the requested dimensions, such as Query, Wait-state, Application, or Host, measured at that time point.

Amazon DocumentDB Performance Insights monitors your Amazon DocumentDB DB instance so that you can analyze and troubleshoot database performance. One way to view Performance Insights data is in the AWS Management Console. Performance Insights also provides a public API so that you can query your own data. You can use the API to do the following:

- Offload data into a database
- Add Performance Insights data to existing monitoring dashboards
- Build monitoring tools

To use the Performance Insights API, enable Performance Insights on one of your Amazon DocumentDB instances. For information about enabling Performance Insights, see [Enabling and disabling Performance Insights \(p. 437\)](#). For more information about the Performance Insights API, see the [Performance Insights API Reference](#).

The Performance Insights API provides the following operations.

Performance Insights action	AWS CLI command	Description
DescribeDimensionKeys	<code>aws pi describe-dimension-keys</code>	Retrieves the top N dimension keys for a metric for a specific time period.

Performance Insights action	AWS CLI command	Description
GetDimensionKeyDetails	<code>aws pi get-dimension-key-details</code>	Retrieves the attributes of the specified dimension group for a DB instance or data source. For example, if you specify a query ID, and if the dimension details are available, <code>GetDimensionKeyDetails</code> retrieves the full text of the dimension db.query.statement associated with this ID. This operation is useful because <code>GetResourceMetrics</code> and <code>DescribeDimensionKeys</code> don't support retrieval of large query statement text.
GetResourceMetadata	<code>aws pi get-resource-metadata</code>	Retrieve the metadata for different features. For example, the metadata might indicate that a feature is turned on or off on a specific DB instance.
GetResourceMetrics	<code>aws pi get-resource-metrics</code>	Retrieves Performance Insights metrics for a set of data sources over a time period. You can provide specific dimension groups and dimensions, and provide aggregation and filtering criteria for each group.
ListAvailableResourceDimensions	<code>aws pi list-available-resource-dimensions</code>	Retrieve the dimensions that can be queried for each specified metric type on a specified instance.
ListAvailableResourceMetrics	<code>aws pi list-available-resource-metrics</code>	Retrieve all available metrics of the specified metric types that can be queried for a specified DB instance.

Topics

- [AWS CLI for Performance Insights \(p. 457\)](#)
- [Retrieving time-series metrics \(p. 458\)](#)
- [AWS CLI examples for Performance Insights \(p. 459\)](#)

AWS CLI for Performance Insights

You can view Performance Insights data using the AWS CLI. You can view help for the AWS CLI commands for Performance Insights by entering the following on the command line.

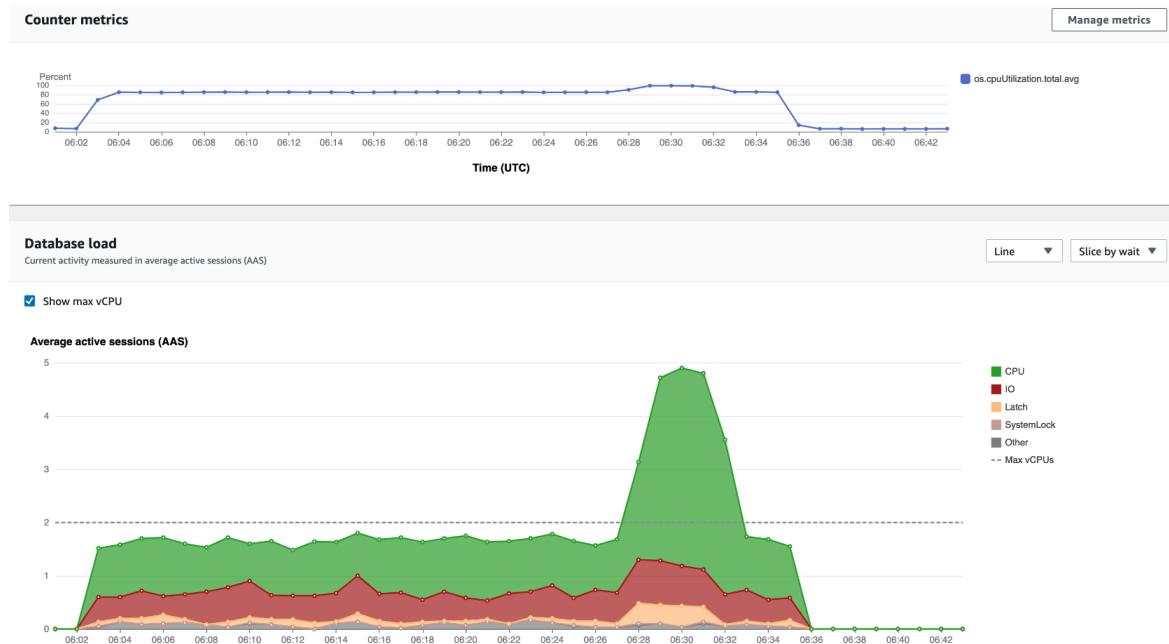
```
aws pi help
```

If you don't have the AWS CLI installed, see [Installing the AWS Command Line Interface](#) in the *AWS CLI User Guide* for information about installing it.

Retrieving time-series metrics

The `GetResourceMetrics` operation retrieves one or more time-series metrics from the Performance Insights data. `GetResourceMetrics` requires a metric and time period, and returns a response with a list of data points.

For example, the AWS Management Console uses `GetResourceMetrics` to populate the **Counter Metrics** chart and the **Database Load** chart, as seen in the following image.



All metrics returned by `GetResourceMetrics` are standard time-series metrics, with the exception of `db.load`. This metric is displayed in the **Database Load** chart. The `db.load` metric is different from the other time-series metrics because you can break it into subcomponents called *dimensions*. In the previous image, `db.load` is broken down and grouped by the waits states that make up the `db.load`.

Note

`GetResourceMetrics` can also return the `db.sampleload` metric, but the `db.load` metric is appropriate in most cases.

For information about the counter metrics returned by `GetResourceMetrics`, see [Performance Insights for counter metrics \(p. 467\)](#).

The following calculations are supported for the metrics:

- Average – The average value for the metric over a period of time. Append `.avg` to the metric name.
- Minimum – The minimum value for the metric over a period of time. Append `.min` to the metric name.
- Maximum – The maximum value for the metric over a period of time. Append `.max` to the metric name.
- Sum – The sum of the metric values over a period of time. Append `.sum` to the metric name.
- Sample count – The number of times the metric was collected over a period of time. Append `.sample_count` to the metric name.

For example, assume that a metric is collected for 300 seconds (5 minutes), and that the metric is collected one time each minute. The values for each minute are 1, 2, 3, 4, and 5. In this case, the following calculations are returned:

- Average – 3
- Minimum – 1
- Maximum – 5
- Sum – 15
- Sample count – 5

For information about using the `get-resource-metrics` AWS CLI command, see [get-resource-metrics](#).

For the `--metric-queries` option, specify one or more queries that you want to get results for. Each query consists of a mandatory Metric and optional GroupBy and Filter parameters. The following is an example of a `--metric-queries` option specification.

```
{  
    "Metric": "string",  
    "GroupBy": {  
        "Group": "string",  
        "Dimensions": ["string", ...],  
        "Limit": integer  
    },  
    "Filter": {"string": "string"  
    ...}}
```

AWS CLI examples for Performance Insights

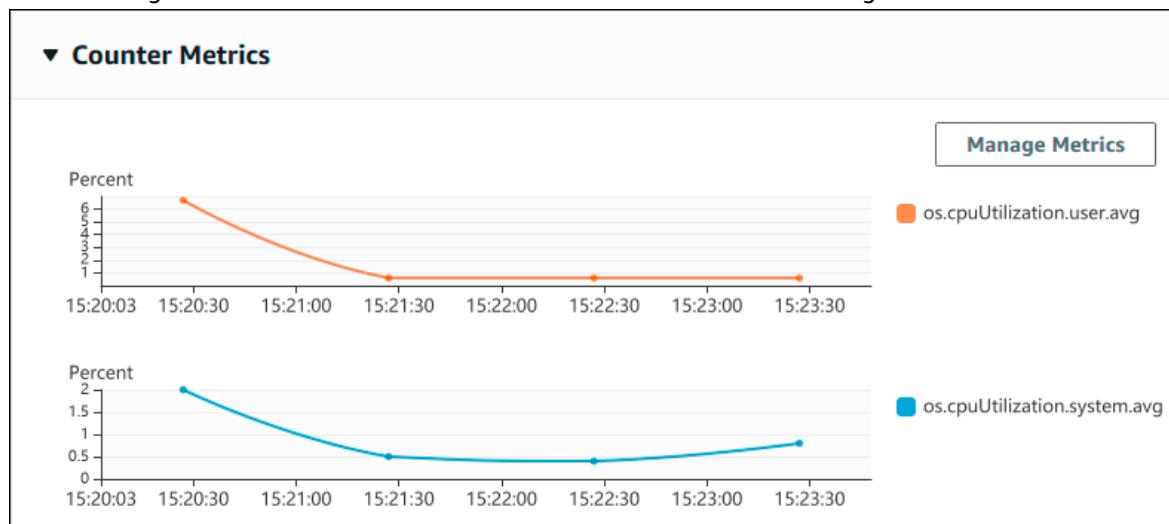
The following examples show how to use the AWS CLI for Performance Insights.

Topics

- [Retrieving counter metrics \(p. 459\)](#)
- [Retrieving the DB load average for top wait states \(p. 462\)](#)
- [Retrieving the DB load average for top Query \(p. 463\)](#)
- [Retrieving the DB load average filtered by Query \(p. 465\)](#)

Retrieving counter metrics

The following screenshot shows two counter metrics charts in the AWS Management Console.



The following example shows how to gather the same data that the AWS Management Console uses to generate the two counter metric charts.

For Linux, macOS, or 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"}]'
```

For 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"}]'
```

You can also make a command easier to read by specifying a file for the `--metrics-query` option. The following example uses a file called `query.json` for the option. The file has the following contents.

```
[  
  {  
    "Metric": "os.cpuUtilization.user.avg"  
  },  
  {  
    "Metric": "os.cpuUtilization.idle.avg"  
  }]
```

Run the following command to use the file.

For Linux, macOS, or 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
```

For 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
```

The preceding example specifies the following values for the options:

- **--service-type** – DOCDB for Amazon DocumentDB
- **--identifier** – The resource ID for the DB instance
- **--start-time** and **--end-time** – The ISO 8601 DateTime values for the period to query, with multiple supported formats

It queries for a one-hour time range:

- **--period-in-seconds** – 60 for a per-minute query
- **--metric-queries** – An array of two queries, each just for one metric.

The metric name uses dots to classify the metric in a useful category, with the final element being a function. In the example, the function is avg for each query. As with Amazon CloudWatch, the supported functions are min, max, total, and avg.

The response looks similar to the following.

```
{
    "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
```

The response has an Identifier, AlignedStartTime, and AlignedEndTime. B the --period-in-seconds value was 60, the start and end times have been aligned to the minute. If the --period-in-seconds was 3600, the start and end times would have been aligned to the hour.

The MetricList in the response has a number of entries, each with a Key and a DataPoints entry. Each DataPoint has a Timestamp and a Value. Each Datapoints list has 60 data points because the

queries are for per-minute data over an hour, with `Timestamp1/Minute1`, `Timestamp2/Minute2`, and so on, up to `Timestamp60/Minute60`.

Because the query is for two different counter metrics, there are two elements in the response `MetricList`.

Retrieving the DB load average for top wait states

The following example is the same query that the AWS Management Console uses to generate a stacked area line graph. This example retrieves the `db.load.avg` for the last hour with load divided according to the top seven wait states. The command is the same as the command in [Retrieving counter metrics \(p. 459\)](#). However, the `query.json` file has the following contents.

```
[  
  {  
    "Metric": "db.load.avg",  
    "GroupBy": { "Group": "db.wait_state", "Limit": 7 }  
  }  
]
```

Run the following command.

For Linux, macOS, or 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
```

For 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
```

The example specifies the metric of `db.load.avg` and a `GroupBy` of the top seven wait states. For details about valid values for this example, see [DimensionGroup](#) in the *Performance Insights API Reference*.

The response looks similar to the following.

```
{  
  "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": [  
        {  
          "AlignedTime": "2022-04-04T06:00:00+00:00",  
          "Value": 100  
        },  
        {  
          "AlignedTime": "2022-04-04T06:01:00+00:00",  
          "Value": 100  
        },  
        {  
          "AlignedTime": "2022-04-04T06:02:00+00:00",  
          "Value": 100  
        },  
        {  
          "AlignedTime": "2022-04-04T06:03:00+00:00",  
          "Value": 100  
        },  
        {  
          "AlignedTime": "2022-04-04T06:04:00+00:00",  
          "Value": 100  
        },  
        {  
          "AlignedTime": "2022-04-04T06:05:00+00:00",  
          "Value": 100  
        },  
        {  
          "AlignedTime": "2022-04-04T06:06:00+00:00",  
          "Value": 100  
        },  
        {  
          "AlignedTime": "2022-04-04T06:07:00+00:00",  
          "Value": 100  
        },  
        {  
          "AlignedTime": "2022-04-04T06:08:00+00:00",  
          "Value": 100  
        },  
        {  
          "AlignedTime": "2022-04-04T06:09:00+00:00",  
          "Value": 100  
        },  
        {  
          "AlignedTime": "2022-04-04T06:10:00+00:00",  
          "Value": 100  
        },  
        {  
          "AlignedTime": "2022-04-04T06:11:00+00:00",  
          "Value": 100  
        },  
        {  
          "AlignedTime": "2022-04-04T06:12:00+00:00",  
          "Value": 100  
        },  
        {  
          "AlignedTime": "2022-04-04T06:13:00+00:00",  
          "Value": 100  
        },  
        {  
          "AlignedTime": "2022-04-04T06:14:00+00:00",  
          "Value": 100  
        },  
        {  
          "AlignedTime": "2022-04-04T06:15:00+00:00",  
          "Value": 100  
        }  
      ]  
    }  
  ]  
}
```

```

//Each list of datapoints has the same timestamps and same number of items
{
    "Timestamp": "2022-04-04T06:01:00+00:00", //Minute1
    "Value": 0.0
},
{
    "Timestamp": "2022-04-04T06:02:00+00:00", //Minute2
    "Value": 0.0
},
//... 60 datapoints for the total db.load.avg key
]
},
{
    "Key": {
        //Another key. This is db.load.avg broken down by CPU
        "Metric": "db.load.avg",
        "Dimensions": {
            "db.wait_state.name": "CPU"
        }
    },
    "DataPoints": [
        {
            "Timestamp": "2022-04-04T06:01:00+00:00", //Minute1
            "Value": 0.0
        },
        {
            "Timestamp": "2022-04-04T06:02:00+00:00", //Minute2
            "Value": 0.0
        },
        //... 60 datapoints for the CPU key
    ]
}, //... In total we have 3 key/datapoints entries, 1) total, 2-3) Top Wait States
] //end of MetricList
} //end of response

```

In this response, there are three entries in the MetricList. There is one entry for the total db.load.avg, and three entries each for the db.load.avg divided according to one of the top three wait states. Since there was a grouping dimension (unlike the first example), there must be one key for each grouping of the metric. There can't be only one key for each metric, as in the basic counter metric use case.

Retrieving the DB load average for top Query

The following example groups db.wait_state by the top 10 query statements. There are two different groups for query statements:

- db.query – The full query statement, such as `{"find": "customers", "filter": {"FirstName": "Jesse"}, "sort": {"key": {"$numberInt": "1"}}}`
- db.query_tokenized – The tokenized query statement, such as `{"find": "customers", "filter": {"FirstName": "?"}, "sort": {"key": {"$numberInt": "?"}}, "limit": {"$numberInt": "?"}}`

When analyzing database performance, it can be useful to consider query statements that only differ by their parameters as one logic item. So, you can use db.query_tokenized when querying. However, especially when you're interested in `explain()`, sometimes it's more useful to examine full query statements with parameters. There is a parent-child relationship between tokenized and full queries, with multiple full queries (children) grouped under the same tokenized query (parent).

The command in this example is the similar to the command in [Retrieving the DB load average for top wait states \(p. 462\)](#). However, the query.json file has the following contents.

```
[  
  {  
    "Metric": "db.load.avg",  
    "GroupBy": { "Group": "db.query_tokenized", "Limit": 10 }  
  }  
]
```

The following example uses db.query_tokenized.

For Linux, macOS, or 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
```

For 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
```

This example queries over 1 hour, with a one minute period-in-seconds.

The example specifies the metric of db.load.avg and a GroupBy of the top seven wait states. For details about valid values for this example, see [DimensionGroup](#) in the *Performance Insights API Reference*.

The response looks similar to the following.

```
{  
  "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

```

This response has 11 entries in the MetricList (1 total, 10 top tokenized query), with each entry having 24 per-hour DataPoints.

For tokenized queries, there are three entries in each dimensions list:

- db.query_tokenized.statement – The tokenized query statement.
- db.query_tokenized.db_id – The synthetic ID that Performance Insights generates for you. This example returns the pi-1064184600 synthetic ID.
- db.query_tokenized.id – The ID of the query inside Performance Insights.

In the AWS Management Console, this ID is called the Support ID. It's named this because the ID is data that AWS Support can examine to help you troubleshoot an issue with your database. AWS takes the security and privacy of your data extremely seriously, and almost all data is stored encrypted with your AWS KMS customer master key (CMK). Therefore, nobody inside AWS can look at this data. In the example preceding, both the tokenized.statement and the tokenized.db_id are stored encrypted. If you have an issue with your database, AWS Support can help you by referencing the Support ID.

When querying, it might be convenient to specify a Group in GroupBy. However, for finer-grained control over the data that's returned, specify the list of dimensions. For example, if all that is needed is the db.query_tokenized.statement, then a Dimensions attribute can be added to the query.json file.

```

[
    {
        "Metric": "db.load.avg",
        "GroupBy": {
            "Group": "db.query_tokenized",
            "Dimensions": ["db.query_tokenized.statement"],
            "Limit": 10
        }
    }
]

```

Retrieving the DB load average filtered by Query

The corresponding API query in this example is similar to the command in [Retrieving the DB load average for top Query \(p. 463\)](#). However, the query.json file has the following contents.

```

[
    {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.wait_state", "Limit": 5 },
        "Filter": { "db.query_tokenized.id": "AKIAIOSFODNN7EXAMPLE" }
    }
]

```

In this response, all values are filtered according to the contribution of tokenized query AKIAIOSFODNN7EXAMPLE specified in the query.json file. The keys also might follow a different order than a query without a filter, because it's the top five wait states that affected the filtered query.

Amazon CloudWatch metrics for Performance Insights

Performance Insights automatically publishes metrics to Amazon CloudWatch. The same data can be queried from Performance Insights, but having the metrics in CloudWatch makes it easy to add CloudWatch alarms. It also makes it easy to add the metrics to existing CloudWatch Dashboards.

Metric	Description
DBLoad	The number of active sessions for Amazon DocumentDB. Typically, you want the data for the average number of active sessions. In Performance Insights, this data is queried as db.load.avg.
DBLoadCPU	The number of active sessions where the wait state type is CPU. In Performance Insights, this data is queried as db.load.avg, filtered by the wait state type CPU.
DBLoadNonCPU	The number of active sessions where the wait state type is not CPU.

Note

These metrics are published to CloudWatch only if there is load on the DB instance.

You can examine these metrics using the CloudWatch console, the AWS CLI, or the CloudWatch API.

For example, you can get the statistics for the DBLoad metric by running the [get-metric-statistics](#) command.

```
aws cloudwatch get-metric-statistics \
--region ap-south-1 \
--namespace AWS/DocDB \
--metric-name DBLoad \
--period 360 \
--statistics Average \
--start-time 2022-03-14T08:00:00Z \
--end-time 2022-03-14T09:00:00Z \
--dimensions Name=DBInstanceIdentifier,Value=documentdbinstance
```

This example generates output similar to the following.

```
{
  "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"
}

```

For more information about CloudWatch, see [What is Amazon CloudWatch?](#) in the *Amazon CloudWatch User Guide*.

Performance Insights for counter metrics

Counter metrics are operating system metrics in the Performance Insights dashboard. To help identify and analyze performance problems, you can correlate counter metrics with DB load.

Performance Insights operating system counters

The following operating system counters are available with DocumentDB Performance Insights.

Counter	Type	Metric
active	memory	os.memory.active
buffers	memory	os.memory.buffers
cached	memory	os.memory.cached
dirty	memory	os.memory.dirty
free	memory	os.memory.free
inactive	memory	os.memory.inactive
mapped	memory	os.memory.mapped
pageTables	memory	os.memory.pageTables
slab	memory	os.memory.slab
total	memory	os.memory.total
writeback	memory	os.memory.writeback
idle	cpuUtilization	os.cpuUtilization.idle

Counter	Type	Metric
system	cpuUtilization	os.cpuUtilization.system
total	cpuUtilization	os.cpuUtilization.total
user	cpuUtilization	os.cpuUtilization.user
wait	cpuUtilization	os.cpuUtilization.wait
one	loadAverageMinute	os.loadAverageMinute.one
fifteen	loadAverageMinute	os.loadAverageMinute.fifteen
five	loadAverageMinute	os.loadAverageMinute.five
cached	swap	os.swap.cached
free	swap	os.swap.free
in	swap	os.swap.in
out	swap	os.swap.out
total	swap	os.swap.total
rx	network	os.network.rx
tx	network	os.network.tx
numVCPU	general	os.general.numVCPU

Developing with Amazon DocumentDB

These sections cover development using Amazon DocumentDB (with MongoDB compatibility).

Topics

- [Connecting Programmatically to Amazon DocumentDB \(p. 469\)](#)
- [Using Change Streams with Amazon DocumentDB \(p. 486\)](#)
- [Connecting to Amazon DocumentDB as a Replica Set \(p. 496\)](#)
- [Connecting to an Amazon DocumentDB Cluster from Outside an Amazon VPC \(p. 499\)](#)
- [Connecting to an Amazon DocumentDB Cluster from Robo 3T \(p. 500\)](#)
- [Connecting to an Amazon DocumentDB Cluster from Studio 3T \(p. 503\)](#)
- [Connect Using Amazon EC2 \(p. 509\)](#)
- [Connect Using Amazon DocumentDB JDBC Driver \(p. 523\)](#)

Connecting Programmatically to Amazon DocumentDB

This section contains code examples that demonstrate how to connect to Amazon DocumentDB (with MongoDB compatibility) using several different languages. The examples are separated into two sections based on whether you are connecting to a cluster that has Transport Layer Security (TLS) enabled or disabled. By default, TLS is enabled on Amazon DocumentDB clusters. However, you can turn off TLS if you want. For more information, see [Encrypting Data in Transit \(p. 147\)](#).

If you are attempting to connect to your Amazon DocumentDB from outside the VPC in which your cluster resides, please see [Connecting to an Amazon DocumentDB Cluster from Outside an Amazon VPC \(p. 499\)](#).

Before you connect to your cluster, you must know whether TLS is enabled on the cluster. The next section shows you how to determine the value of your cluster's `tls` parameter using either the AWS Management Console or the AWS CLI. Following that, you can continue by finding and applying the appropriate code example.

Topics

- [Determining the Value of Your `tls` Parameter \(p. 469\)](#)
- [Connecting with TLS Enabled \(p. 471\)](#)
- [Connecting with TLS Disabled \(p. 480\)](#)

Determining the Value of Your `tls` Parameter

Determining whether your cluster has TLS enabled is a two-step process that you can perform using either the AWS Management Console or AWS CLI.

1. **Determine which parameter group is governing your cluster.**

Using the AWS Management Console

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the left navigation pane, choose **Clusters**.
3. In the list of clusters, select the name of your cluster.
4. The resulting page shows the details of the cluster that you selected. Scroll down to **Cluster details**. At the bottom of that section, locate the parameter group's name below **Cluster parameter group**.

Using the AWS CLI

The following AWS CLI code determines which parameter is governing your cluster. Make sure you replace `sample-cluster` with the name of your cluster.

```
aws docdb describe-db-clusters \
--db-cluster-identifier sample-cluster \
--query 'DBClusters[*].[DBClusterIdentifier,DBClusterParameterGroup]'
```

Output from this operation looks something like the following:

```
[  
  [  
    "sample-cluster",  
    "sample-parameter-group"  
  ]  
]
```

2. **Determine the value of the `tls` parameter in your cluster's parameter group.**

Using the AWS Management Console

1. In the navigation pane, choose **Parameter groups**.
2. In the **Cluster parameter groups** window, select your cluster parameter group.
3. The resulting page shows your cluster parameter group's parameters. You can see the value of the `tls` parameter here. For information on modifying this parameter, see [Modifying Amazon DocumentDB Cluster Parameter Groups \(p. 362\)](#).

Using the AWS CLI

You can use the `describe-db-cluster-parameters` AWS CLI command to view the details of the parameters in your cluster parameter group.

- **--describe-db-cluster-parameters** — To list all the parameters inside a parameter group and their values.
- **--db-cluster-parameter-group name** — Required. The name of your cluster parameter group.

```
aws docdb describe-db-cluster-parameters \
--db-cluster-parameter-group-name sample-parameter-group
```

Output from this operation looks something like the following:

```
{  
    "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",  
            "IsModifiable": true,  
            "ApplyMethod": "pending-reboot"  
        }  
    ]  
}
```

After determining the value of your `tls` parameter, continue connecting to your cluster by using one of the code examples in the following sections.

- [Connecting with TLS Enabled \(p. 471\)](#)
- [Connecting with TLS Disabled \(p. 480\)](#)

Connecting with TLS Enabled

To view a code example for programmatically connecting to a TLS-enabled Amazon DocumentDB cluster, choose the appropriate tab for the language that you want to use.

To encrypt data in transit, download the public key for Amazon DocumentDB named `rds-combined-ca-bundle.pem` using the following operation.

```
wget https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem
```

Python

The following code demonstrates how to connect to Amazon DocumentDB using Python when TLS is enabled.

```
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=rds-combined-ca-
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

The following code demonstrates how to connect to Amazon DocumentDB using Node.js when TLS is enabled.

```

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: `rds-combined-ca-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':'DocDB'}, function(err, result){
      //Print the result to the screen
      console.log(result);

      //Close the connection
      client.close()
    });
  });
}
);

```

PHP

The following code demonstrates how to connect to Amazon DocumentDB using PHP when TLS is enabled.

```
<?php
//Include Composer's autoloader
require 'vendor/autoload.php';

$TLS_DIR = "/home/ubuntu/rds-combined-ca-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

The following code demonstrates how to connect to Amazon DocumentDB using Go when TLS is enabled.

Note

As of version 1.2.1, the MongoDB Go Driver will only use the first CA server certificate found in `sslcertificateauthorityfile`. The example code below addresses this limitation by manually appending all server certificates found in `sslcertificateauthorityfile` to a custom TLS configuration used during client creation.

```
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 = "rds-combined-ca-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

When connecting to a TLS-enabled Amazon DocumentDB cluster from a Java application, your program must use the AWS-provided certificate authority (CA) file to validate the connection. To use the Amazon RDS CA certificate, do the following:

1. Download the Amazon RDS CA file from <https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem>.
2. Create a trust store with the CA certificate contained in the file by performing the following commands. Be sure to change the `<truststorePassword>` to something else. If you are accessing a trust store that contains both the old CA certificate (`rds-ca-2015-root.pem`) and the new CA certificate (`rds-ca-2019-root.pem`), you can import the certificate bundle into the trust store.

The following is a sample shell script that imports the certificate bundle into a trust store on a Linux operating system.

```

mydir=/tmp/certs
truststore=${mydir}/rds-truststore.jks
storepassword=<truststorePassword>

curl -sS "https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem" >
${mydir}/rds-combined-ca-bundle.pem
awk 'split_after == 1 {n++;split_after=0} /----END CERTIFICATE----/
{split_after=1}{print > "rds-ca-" n ".pem"}' < ${mydir}/rds-combined-ca-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}/rds-combined-ca-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

```

The following is a sample shell script that imports the certificate bundle into a trust store on macOS.

```

mydir=/tmp/certs
truststore=${mydir}/rds-truststore.jks
storepassword=<truststorePassword>

curl -sS "https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem" >
${mydir}/rds-combined-ca-bundle.pem
split -p "-----BEGIN CERTIFICATE-----" ${mydir}/rds-combined-ca-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}/rds-combined-ca-bundle.pem

echo "Trust store content is: "

keytool -list -v -keystore "$truststore" -storepass ${storepassword} | grep Alias | 
cut -d " " -f3- | while read alias
do
    expiry=`keytool -list -v -keystore "$truststore" -storepass ${storepassword} - 
alias "${alias}" | grep Valid | perl -ne 'if(/until: (.*)\n/) { print "$1\n"; }'` 
    echo " Certificate ${alias} expires in '$expiry'"
done

```

3. Use the keystore in your program by setting the following system properties in your application before making a connection to the Amazon DocumentDB cluster.

```

javax.net.ssl.trustStore: <truststore>
javax.net.ssl.trustStorePassword: <truststorePassword>

```

4. The following code demonstrates how to connect to Amazon DocumentDB using Java when TLS is enabled.

```

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://:

@/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 = "<truststorePassword>";

        System.setProperty("javax.net.ssl.trustStore", truststore);
        System.setProperty("javax.net.ssl.trustStorePassword", truststorePassword);

        MongoClient mongoClient = MongoClients.create(connectionString);

        MongoDB 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

The following code demonstrates how to connect to Amazon DocumentDB using C# / .NET when TLS is enabled.

```

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 pathToCAFFile = "<PATH/rds-combined-ca-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(pathToCAFFile);
            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);
        }
    }
}

```

mongo shell

The following code demonstrates how to connect to and query Amazon DocumentDB using the mongo shell when TLS is enabled.

1. Connect to Amazon DocumentDB with the mongo shell. If you are on a mongo shell version earlier than 4.2, use the following code to connect.

```
mongo --ssl --host sample-cluster.node.us-east-1.docdb.amazonaws.com:27017 --sslCAFile rds-combined-ca-bundle.pem --username <sample-user> --password <password>
```

If you are using a version equal to or greater than 4.2, use the following code to connect. Retryable writes are not supported in AWS DocumentDB. Exception: If you are using mongo shell, do not include the `retryWrites=false` command in any code string. By default, retryable writes are disabled. Including `retryWrites=false` might cause failure in normal read commands.

```
mongo --tls --host sample-cluster.node.us-east-1.docdb.amazonaws.com:27017 --tlsCAFile rds-combined-ca-bundle.pem --username <sample-user> --password <password>
```

2. Insert a single document.

```
db.myTestCollection.insertOne({'hello':'Amazon DocumentDB'})
```

3. Find the document that was previously inserted.

```
db.myTestCollection.find({'hello':'Amazon DocumentDB'})
```

R

The following code demonstrates how to connect to Amazon DocumentDB with R using mongolite (<https://jeroen.github.io/mongolite/>) when TLS is enabled.

```
#Include the mongolite library.
library(mongolite)

mongouri <- 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 = mongouri, options = ssl_options(weak_cert_validation = F, ca = "<PATH/rds-combined-ca-bundle.pem>"))

#Insert a single document
str <- c('{"hello" : "Amazon DocumentDB"}')
client$insert(str)

#Find the document that was previously written
client$find()
```

Ruby

The following code demonstrates how to connect to Amazon DocumentDB with Ruby when TLS is enabled.

```
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/rds-combined-ca-bundle.pem'>',
    retryWrites: 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

```

Connecting with TLS Disabled

To view a code example for programmatically connecting to a TLS-disabled Amazon DocumentDB cluster, choose the tab for language that you want to use.

Python

The following code demonstrates how to connect to Amazon DocumentDB using Python when TLS is disabled.

```

## 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

The following code demonstrates how to connect to Amazon DocumentDB using Node.js when TLS is disabled.

```
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

The following code demonstrates how to connect to Amazon DocumentDB using PHP when TLS is disabled.

```
<?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

The following code demonstrates how to connect to Amazon DocumentDB using Go when TLS is disabled.

```
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://:/:@/:/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

The following code demonstrates how to connect to Amazon DocumentDB using Java when TLS is disabled.

```

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

The following code demonstrates how to connect to Amazon DocumentDB using C# / .NET when TLS is disabled.

```

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://{}:{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);
}
```

mongo shell

The following code demonstrates how to connect to and query Amazon DocumentDB using the mongo shell when TLS is disabled.

1. Connect to Amazon DocumentDB with the mongo shell.

```
mongo --host mycluster.node.us-east-1.docdb.amazonaws.com:27017 --username <sample-user> --password <password>
```

2. Insert a single document.

```
db.myTestCollection.insertOne({'hello':'Amazon DocumentDB'})
```

3. Find the document that was previously inserted.

```
db.myTestCollection.find({'hello':'Amazon DocumentDB'})
```

R

The following code demonstrates how to connect to Amazon DocumentDB with R using mongolite (<https://jeroen.github.io/mongolite/>) when TLS is disabled.

```
#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

The following code demonstrates how to connect to Amazon DocumentDB with Ruby when TLS is disabled.

```
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
```

Using Change Streams with Amazon DocumentDB

The change streams feature in Amazon DocumentDB (with MongoDB compatibility) provides a time-ordered sequence of change events that occur within your cluster's collections. You can read events from a change stream to implement many different use cases, including the following:

- Change notification
- Full-text search with Amazon OpenSearch Service (OpenSearch Service)
- Analytics with Amazon Redshift

Applications can use change streams to subscribe to data changes on individual collections. Change streams events are ordered as they occur on the cluster and are stored for 3 hours (by default) after the event has been recorded. The retention period can be extended up to 7 days using the `change_stream_log_retention_duration` parameter. To modify the change stream retention period, please see [Modifying the Change Stream Log Retention Duration](#).

Topics

- [Supported Operations \(p. 487\)](#)
- [Billing \(p. 487\)](#)
- [Limitations \(p. 487\)](#)
- [Enabling Change Streams \(p. 487\)](#)
- [Example: Using Change Streams with Python \(p. 489\)](#)
- [Full Document Lookup \(p. 491\)](#)

- [Resuming a Change Stream \(p. 491\)](#)
- [Resuming a Change Stream with startAtOperationTime \(p. 492\)](#)
- [Transactions in change streams \(p. 494\)](#)
- [Modifying the Change Stream Log Retention Duration \(p. 494\)](#)

Supported Operations

Amazon DocumentDB supports the following operations for change streams:

- All change events supported in the MongoDB `db.collection.watch()`, `db.watch()` and `client.watch()` API.
- Full document lookup for updates.
- Aggregation stages: `$match`, `$project`, `$redact`, and `$addFields` and `$replaceRoot`.
- Resuming a change stream from a resume token
- Resuming a change stream from a timestamp using `startAtOperation` (applicable to Amazon DocumentDB v4.0+)

Billing

The Amazon DocumentDB change streams feature is disabled by default and does not incur any additional charges until the feature is enabled. Using change streams in a cluster incurs additional read and write IOs and storage costs. You can use the `modifyChangeStreams` API operation to enable this feature for your cluster. For more information on pricing, see [Amazon DocumentDB pricing](#).

Limitations

Change streams have the following limitations in Amazon DocumentDB:

- Change streams can only be opened from a connection to the primary instance of an Amazon DocumentDB cluster. Reading from change streams on a replica instance is not currently supported. When invoking the `watch()` API operation, you must specify a **primary** read preference to ensure that all reads are directed to the primary instance (see the [Example \(p. 489\)](#) section).
- Events written to a change stream for a collection are available for up to 7 days (the default is 3 hours). Change streams data is deleted after the log retention duration window, even if no new changes have occurred.
- A long-running write operation on a collection like `updateMany` or `deleteMany` can temporarily stall the writing of change streams events until the long running write operation is complete.
- Amazon DocumentDB does not support the MongoDB operations log (oplog).
- With Amazon DocumentDB, you must explicitly enable change streams on a given collection.
- If the total size of a change streams event (including the change data and full document, if requested) is greater than 16 MB, the client will experience a read failure on the change streams.
- The Ruby driver is currently not supported when using `db.watch()` and `client.watch()` with Amazon DocumentDB v3.6.

Enabling Change Streams

You can enable Amazon DocumentDB change streams for all collections within a given database, or only for selected collections. The following are examples of how to enable change streams for different

use cases using the mongo shell. Empty strings are treated as wildcards when specifying database and collection names.

```
//Enable change streams for the collection "foo" in database "bar"
db.adminCommand({modifyChangeStreams: 1,
  database: "bar",
  collection: "foo",
  enable: true});
```

```
//Disable change streams on collection "foo" in database "bar"
db.adminCommand({modifyChangeStreams: 1,
  database: "bar",
  collection: "foo",
  enable: false});
```

```
//Enable change streams for all collections in database "bar"
db.adminCommand({modifyChangeStreams: 1,
  database: "bar",
  collection: "",
  enable: true});
```

```
//Enable change streams for all collections in all databases in a cluster
db.adminCommand({modifyChangeStreams: 1,
  database: "",
  collection: "",
  enable: true});
```

Change streams will be enabled for a collection if any of the following are true:

- Both the database and collection are explicitly enabled.
- The database containing the collection is enabled.
- All databases are enabled.

Dropping a collection from a database does not disable change streams for that collection if the parent database also has change streams enabled, or if all databases in the cluster are enabled. If a new collection is created with the same name as the deleted collection, change streams will be enabled for that collection.

You can list all of your cluster's enabled change streams by using the `$listChangeStreams` aggregation pipeline stage. All aggregation stages supported by Amazon DocumentDB can be used in the pipeline for additional processing. If a previously enabled collection has been disabled, it will not appear in the `$listChangeStreams` output.

```
//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:{}));
```

Example: Using Change Streams with Python

The following is an example of using an Amazon DocumentDB change stream with Python at the collection level.

```
import os
import sys
from pymongo import MongoClient, ReadPreference

username = "DocumentDBUsername"
password = <Insert your password>

clusterendpoint = "DocumentDBClusterEndpoint"
client = MongoClient(clusterendpoint, username=username, password=password, ssl='true',
                      ssl_ca_certs='rds-combined-ca-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}}}
"""

```

The following is an example of using an Amazon DocumentDB change stream with Python at the database level.

```

import os
import sys
from pymongo import MongoClient

username = "DocumentDBusername"
password = <Insert your password>
clusterendpoint = "DocumentDBClusterEndpoint"
client = MongoClient(clusterendpoint, username=username, password=password, ssl='true',
                      ssl_ca_certs='rds-combined-ca-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': '015daf94f60000002010000000200009025'},
'clusterTime': Timestamp(1571788022, 2),
'documentKey': {'_id': ObjectId('5daf94f6ea258751778163d6')},
'fullDocument': {'_id': ObjectId('5daf94f6ea258751778163d6'), 'x': 1},
'ns': {'coll': 'foo1', 'db': 'bar'},
'operationType': 'insert'}
"""

```

Full Document Lookup

The update change event does not include the full document; it includes only the change that was made. If your use case requires the complete document affected by an update, you can enable full document lookup when opening the stream.

The fullDocument document for an update change streams event represents the most current version of the updated document at the time of document lookup. If changes occurred between the update operation and the fullDocument lookup, the fullDocument document might not represent the document state at update time.

```

#Create a stream object with update lookup enabled
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:
{'_id': {'_data': '015daf9b7c0000001010000000100009025'},
'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}}}

```

Resuming a Change Stream

You can resume a change stream later by using a resume token, which is equal to the `_id` field of the last retrieved change event document.

```

import os
import sys
from pymongo import MongoClient

username = "DocumentDBusername"
password = <Insert your password>
clusterendpoint = "DocumentDBCclusterEndpoint"
client = MongoClient(clusterendpoint, username=username, password=password, ssl='true',
                      ssl_ca_certs='rds-combined-ca-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': '015daf9c5b00000001010000000100009025'}
"""

#Python provides a nice shortcut for getting a stream's resume token
print(stream.resume_token)

"""
Output
{'_data': '015daf9c5b00000001010000000100009025'}
"""

#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': '015f7e8f0c0000006010000006000fe038'},
'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': '015f7e8f0c0000007010000007000fe038'},
'operationType': 'insert',
'clusterTime': Timestamp(1602129676, 7),
'ns': {'db': 'bar', 'coll': 'foo'},
'documentKey': {'_id': ObjectId('5f7e8f0cbf8c233ed577eb94')},
'fullDocument': {'_id': ObjectId('5f7e8f0cbf8c233ed577eb94'), 'y': 5}}
"""

```

Resuming a Change Stream with `startAtOperationTime`

You can resume a change stream later from a particular time stamp by using `startAtOperationTime`.

Note

The ability to use `startAtOperationTime` is available in Amazon DocumentDB 4.0+. When using `startAtOperationTime`, the change stream cursor will only return changes that

occurred at or after the specified `Timestamp`. The `startAtOperationTime` and `resumeAfter` commands are mutually exclusive and thus cannot be used together.

```

import os
import sys
from pymongo import MongoClient

username = "DocumentDBusername"
password = <Insert your password>
clusterendpoint = "DocumentDBCclusterEndpoint"
client = MongoClient(clusterendpoint, username=username, password=password, ssl='true',
    ssl_ca_certs='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': '015f7e941a000000030100000003000fe038'},
'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}}
```

Transactions in change streams

Change stream events will not contain events from uncommitted and/or aborted transactions. For example, if you start a transaction with one INSERT operation and one UPDATE operation and. If your INSERT operation succeeds, but the UPDATE operation fails, the transaction will be rolled back. Since this transaction was rolled back, your change stream will not contain any events for this transaction.

Modifying the Change Stream Log Retention Duration

You can modify the change stream log retention duration to be between 1 hour and 7 days using the AWS Management Console or the AWS CLI.

Using the AWS Management Console

To modify the change stream log retention duration

1. Sign in to the AWS Management Console, and open the Amazon DocumentDB console at <https://console.aws.amazon.com/docdb>.
2. In the navigation pane, choose **Parameter groups** .
Tip
If you don't see the navigation pane on the left side of your screen, choose the menu icon (≡) in the upper-left corner of the page.
3. In the **Parameter groups** pane, choose the cluster parameter group that is associated with your cluster. To identify the cluster parameter group that is associated with your cluster, see [Determining an Amazon DocumentDB Cluster's Parameter Group \(p. 358\)](#).
4. The resulting page shows the parameters and their corresponding details for your cluster parameter group. Select the parameter `change_stream_log_retention_duration`.
5. On the top right of the page, choose **Edit** to change the value of the parameter. The `change_stream_log_retention_duration` parameter can be modified to be between 1 and 7 days.
6. Make your change, and then choose **Modify cluster parameter** to save the changes. To discard your changes, choose **Cancel**.

Using the AWS CLI

To modify your cluster parameter group's `change_stream_log_retention_duration` parameter, use the `modify-db-cluster-parameter-group` operation with the following parameters:

- **--db-cluster-parameter-group-name** — Required. The name of the cluster parameter group that you are modifying. To identify the cluster parameter group that is associated with your cluster, see [Determining an Amazon DocumentDB Cluster's Parameter Group \(p. 358\)](#).
- **--parameters** — Required. The parameter that you are modifying. Each parameter entry must include the following:
 - **ParameterName** — The name of the parameter that you are modifying. In this case, it is `change_stream_log_retention_duration`

- **ParameterValue** — The new value for this parameter.
- **ApplyMethod** — How you want changes to this parameter applied. Permitted values are immediate and pending-reboot.

Note

Parameters with the ApplyType of static must have an ApplyMethod of pending-reboot.

1. To change the values of the parameter `change_stream_log_retention_duration`, run the following command and replace `parameter-value` with the value you want to modify the parameter to.

For Linux, macOS, or 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"
```

For 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 from this operation looks something like the following (JSON format).

```
{  
    "DBClusterParameterGroupName": "sample-parameter-group"  
}
```

2. Wait at least 5 minutes.
3. List the parameter values of `sample-parameter-group` to ensure that your changes have been made.

For Linux, macOS, or Unix:

```
aws docdb describe-db-cluster-parameters \
    --db-cluster-parameter-group-name sample-parameter-group
```

For Windows:

```
aws docdb describe-db-cluster-parameters ^
    --db-cluster-parameter-group-name sample-parameter-group
```

Output from this operation looks something like the following (JSON format).

```
{
```

```
"Parameters": [
    {
        "ParameterName": "audit_logs",
        "ParameterValue": "disabled",
        "Description": "Enables auditing on cluster.",
        "Source": "system",
        "ApplyType": "dynamic",
        "DataType": "string",
        "AllowedValues": "enabled,disabled",
        "IsModifiable": true,
        "ApplyMethod": "pending-reboot"
    },
    {
        "ParameterName": "change_stream_log_retention_duration",
        "ParameterValue": "12345",
        "Description": "Duration of time in seconds that the change stream log is retained and can be consumed.",
        "Source": "user",
        "ApplyType": "dynamic",
        "DataType": "integer",
        "AllowedValues": "3600-86400",
        "IsModifiable": true,
        "ApplyMethod": "immediate"
    }
]
```

Connecting to Amazon DocumentDB as a Replica Set

When you're developing against Amazon DocumentDB (with MongoDB compatibility), we recommend that you connect to your cluster as a replica set and distribute reads to replica instances using the built-in read preference capabilities of your driver. This section goes deeper into what that means and describes how you can connect to your Amazon DocumentDB cluster as a replica set using the SDK for Python as an example.

Amazon DocumentDB has three endpoints that you can use to connect to your cluster:

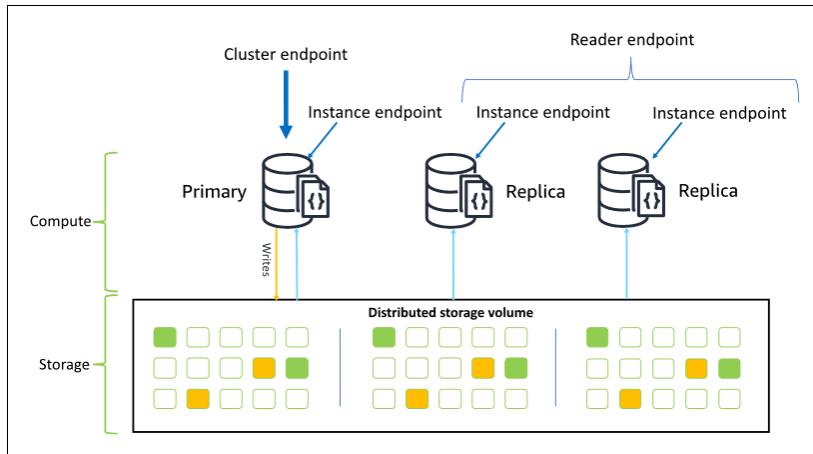
- Cluster endpoint
- Reader endpoint
- Instance endpoints

In most cases when you connect to Amazon DocumentDB, we recommend that you use the cluster endpoint. This is a CNAME that points to the primary instance in your cluster, as shown in the following diagram.

When using an SSH tunnel, we recommend that you connect to your cluster using the cluster endpoint and do not attempt to connect in replica set mode (i.e., specifying `replicaSet=rs0` in your connection string) as it will result in an error.

Note

For more information about Amazon DocumentDB endpoints, see [Amazon DocumentDB Endpoints \(p. 8\)](#).



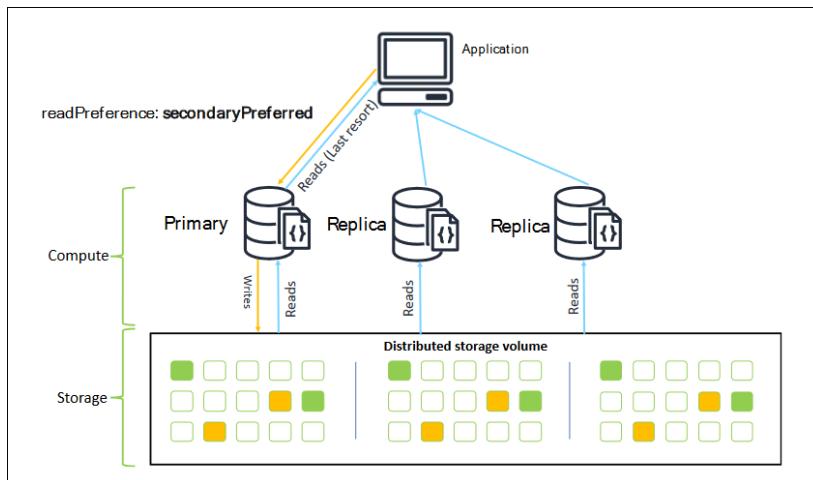
Using the cluster endpoint, you can connect to your cluster in replica set mode. You can then use the built-in read preference driver capabilities. In the following example, specifying `?replicaSet=rs0` signifies to the SDK that you want to connect as a replica set. If you omit `?replicaSet=rs0`, the client routes all requests to the cluster endpoint, that is, your primary instance.

```
## Create a MongoDB client, open a connection to Amazon DocumentDB as a
## replica set and specify the read preference as secondary preferred
client = pymongo.MongoClient('mongodb://<user-name>:<password>@mycluster.node.us-
east-1.docdb.amazonaws.com:27017/?replicaSet=rs0')
```

The advantage of connecting as a replica set is that it enables your SDK to discover the cluster topology automatically, including when instances are added or removed from the cluster. You can then use your cluster more efficiently by routing read requests to your replica instances.

When you connect as a replica set, you can specify the `readPreference` for the connection. If you specify a read preference of `secondaryPreferred`, the client routes read queries to your replicas and write queries to your primary instance (as in the following diagram). This is a better use of your cluster resources. For more information, see [Read Preference Options \(p. 12\)](#).

```
## 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')
```



Reads from Amazon DocumentDB replicas are eventually consistent. They return the data in the same order as it was written on the primary, and there is often less than a 50 ms replication lag. You can monitor the replica lag for your cluster using the Amazon CloudWatch metrics DBInstanceReplicaLag and DBClusterReplicaLagMaximum. For more information, see [Monitoring Amazon DocumentDB with CloudWatch \(p. 414\)](#).

Unlike traditional monolithic database architecture, Amazon DocumentDB separates storage and compute. Given this modern architecture, we encourage you to read scale on replica instances. Reads on replica instances don't block writes being replicated from the primary instance. You can add up to 15 read replica instances in a cluster and scale out to millions of reads per second.

The key benefit of connecting as a replica set and distributing reads to replicas is that it increases the overall resources in your cluster that are available to do work for your application. We recommend connecting as a replica set as a best practice. Further, we recommend it most commonly in the following scenarios:

- You're using nearly 100 percent CPU on your primary.
- The buffer cache hit ratio is near zero.
- You reach the connection or cursor limits for an individual instance.

Scaling up a cluster instance size is an option, and in some cases, that can be the best way to scale the cluster. But you should also consider how to better use the replicas that you already have in your cluster. This lets you increase scale without the increased cost of using a larger instance type. We also recommend that you monitor and alert on these limits (that is CPUUtilization, DatabaseConnections, and BufferCacheHitRatio) using CloudWatch alarms so that you know when a resource is being heavily used.

For more information, see the following topics:

- [Best Practices for Amazon DocumentDB \(p. 84\)](#)
- [Amazon DocumentDB Quotas and Limits \(p. 535\)](#)

Using Cluster Connections

Consider the scenario of using all the connections in your cluster. For example, an `r5.2xlarge` instance has a limit of 4,500 connections (and 450 open cursors). If you create a three-instance Amazon DocumentDB cluster and connect only to the primary instance using the cluster endpoint, your cluster limits for open connections and cursors are 4,500 and 450 respectively. You might reach these limits if you're building applications that use many workers that get spun up in containers. The containers open up a number of connections all at once and saturate the cluster.

Instead, you could connect to the Amazon DocumentDB cluster as a replica set and distribute your reads to the replica instances. You could then effectively triple the number of available connections and cursors available in the cluster to 13,500 and 1,350 respectively. Adding more instances to the cluster only increases the number of connections and cursors for read workloads. If you need to increase the number of connections for writes to your cluster, we recommend increasing the instance size.

Note

The number of connections for `large`, `xlarge`, and `2xlarge` instances increases with the instance size up to 4,500. The maximum number of connections per instance for `4xlarge` instances or greater is 4,500. For more information on limits by instance types, see [Instance Limits \(p. 539\)](#).

Typically we don't recommend that you connect to your cluster using the read preference of `secondary`. This is because if there are no replica instances in your cluster, the reads fail. For example, suppose that you have a two-instance Amazon DocumentDB cluster with one primary and one replica. If the replica

has an issue, read requests from a connection pool that is set as secondary fail. The advantage of `secondaryPreferred` is that if the client can't find a suitable replica instance to connect to, it falls back to the primary for reads.

Multiple Connection Pools

In some scenarios, reads in an application need to have read-after-write consistency, which can be served only from the primary instance in Amazon DocumentDB. In these scenarios, you might create two client connection pools: one for writes and one for reads that need read-after-write consistency. To do that, your code would look something like the following.

```
## 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')
```

Another option is to create a single connection pool and overwrite the read preference for a given collection.

```
##Specify the collection and set the read preference level for that collection
col = db.review.with_options(read_preference=ReadPreference.SECONDARY_PREFERRED)
```

Summary

To better use the resources in your cluster, we recommend that you connect to your cluster using the replica set mode. If it's suitable for your application, you can read scale your application by distributing your reads to the replica instances.

Connecting to an Amazon DocumentDB Cluster from Outside an Amazon VPC

Amazon DocumentDB (with MongoDB compatibility) clusters are deployed within an Amazon Virtual Private Cloud (Amazon VPC). They can be accessed directly by Amazon EC2 instances or other AWS services that are deployed in the same Amazon VPC. Additionally, Amazon DocumentDB can be accessed by EC2 instances or other AWS services in different VPCs in the same AWS Region or other Regions via VPC peering.

However, suppose that your use case requires that you (or your application) access your Amazon DocumentDB resources from outside the cluster's VPC. In that case, you can use SSH tunneling (also known as *port forwarding*) to access your Amazon DocumentDB resources.

It is beyond the scope of this topic to discuss SSH tunneling in depth. For more information about SSH tunneling, see the following:

- [SSH Tunnel](#)
- [SSH Port Forwarding Example](#), specifically the [Local Forwarding](#) section

To create an SSH tunnel, you need an Amazon EC2 instance running in the same Amazon VPC as your Amazon DocumentDB cluster. You can either use an existing EC2 instance in the same VPC as your cluster or create one. For more information, see the topic that is appropriate for your operating system:

- [Getting Started with Amazon EC2 Linux Instances](#)
- [Getting Started with Amazon EC2 Windows Instances](#)

You might typically connect to an EC2 instance using the following command.

```
ssh -i "ec2Access.pem" ubuntu@ec2-34-229-221-164.compute-1.amazonaws.com
```

If so, you can set up an SSH tunnel to the Amazon DocumentDB cluster `sample-cluster.node.us-east-1.docdb.amazonaws.com` by running the following command on your local computer. The `-L` flag is used for forwarding a local port. When using an SSH tunnel, we recommend that you connect to your cluster using the cluster endpoint and do not attempt to connect in replica set mode (i.e., specifying `replicaSet=rs0` in your connection string) as it will result in an error.

```
ssh -i "ec2Access.pem" -L 27017:sample-cluster.node.us-east-1.docdb.amazonaws.com:27017  
ubuntu@ec2-34-229-221-164.compute-1.amazonaws.com -N
```

After the SSH tunnel is created, any commands that you issue to `localhost:27017` are forwarded to the Amazon DocumentDB cluster `sample-cluster` running in the Amazon VPC. If Transport Layer Security (TLS) is enabled on your Amazon DocumentDB cluster, you need to download the public key for Amazon DocumentDB from <https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem>. The following operation downloads this file:

```
wget https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem
```

Note

TLS is enabled by default for new Amazon DocumentDB clusters. However, you can disable it. For more information, see [Managing Amazon DocumentDB Cluster TLS Settings \(p. 147\)](#).

To connect to your Amazon DocumentDB cluster from outside the Amazon VPC, use the following command.

```
mongo --sslAllowInvalidHostnames --ssl --sslCAFile rds-combined-ca-bundle.pem --username  
<yourUsername> --password <yourPassword>
```

Connecting to an Amazon DocumentDB Cluster from Robo 3T

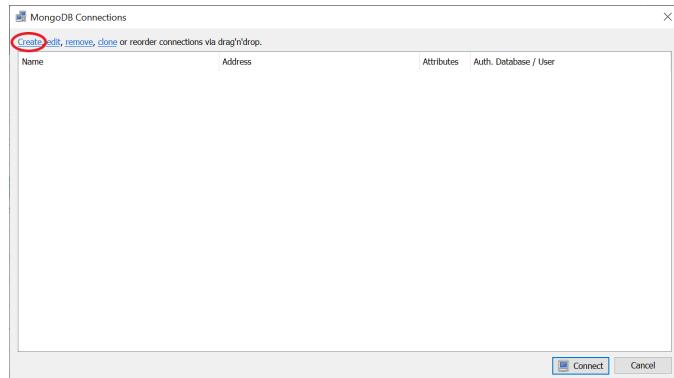
Robo 3T is a lightweight, open-source, shell centric, cross-platform graphical user interface tool for managing MongoDB workloads. Robo 3T gives you the ability to create databases, collections, add users, documents, execute one-time queries with auto-completion, and visualize results from a GUI interface.

Prerequisites

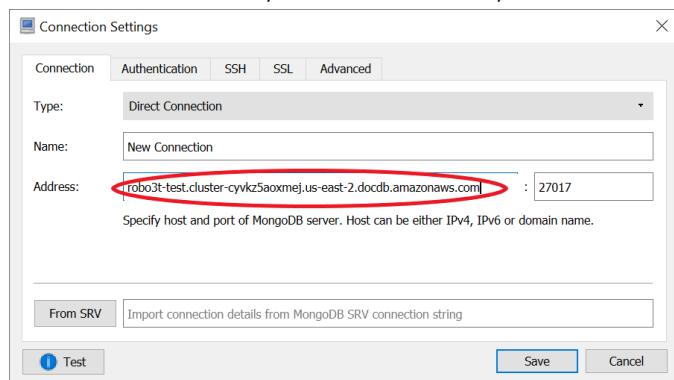
- If you don't already have an Amazon DocumentDB cluster running on Amazon EC2, follow the instructions on how to [Connect with Amazon EC2](#).
- If you don't have Robo 3T, [download and install it](#).

Connect with Robo 3T

1. Open Robo 3T and choose **Create**.



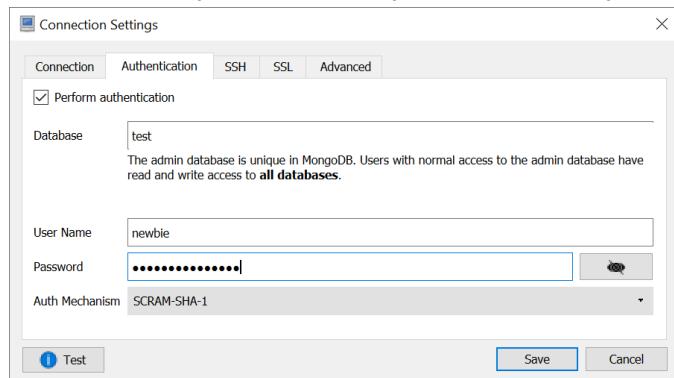
2. On the **Connection** tab, in the **Address** field, enter the cluster endpoint information.



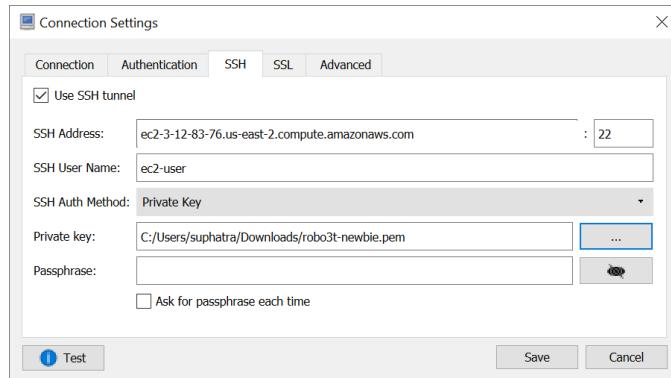
Note

Can't find your cluster endpoint? Just follow the steps [here](#).

3. On the **Authentication** tab, check the box for **Perform Authentication**. Now enter the authentication information for your cluster. Make sure to use a custom database name like test. Using admin (default setting) does not work for Amazon DocumentDB for clusters with no databases. Once you have created your first database you can modify your connection to use admin.



4. On the **SSH** tab, check the box for **Use SSH tunnel**, and add the SSH address, username, and private key/password of your EC2 instance. The SSH address is the public DNS of your EC2 instance.



5. In **SSH Auth Method**, choose one of the authentication methods.

- If you chose **Private Key**, then select the “...” button to open up the file finder and select the .pem file for your EC2 instance.
- If you chose **Password**, you must enter the SSH address, username and private key for your AWS EC2 instance. You can find this on the AWS EC2 console.

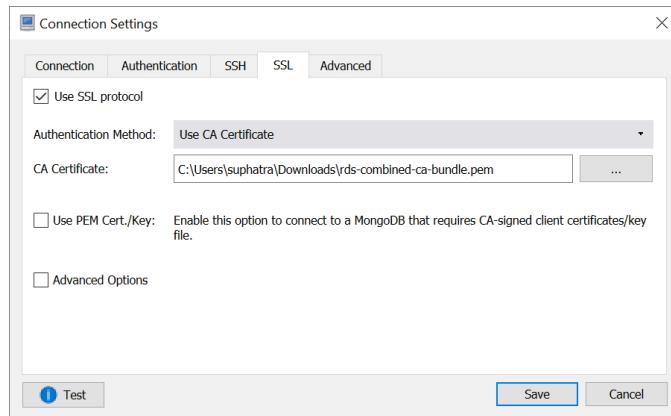
Note

Tip: If you are on Linux/macOS client machine, you might have to change the permissions of your private key using the following command:
 > chmod 400 /fullPathToYourPemFile/<yourKey>.pem

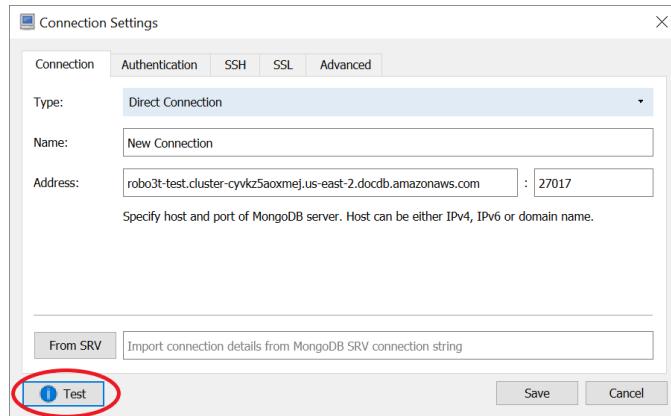
6. Now choose the **SSL** tab and click the drop down menu for **Authentication Method**. Choose **Use CA Certificate**. Select **Advanced Options** and for the **Invalid Hostnames** menu, select **Allowed**.

Note

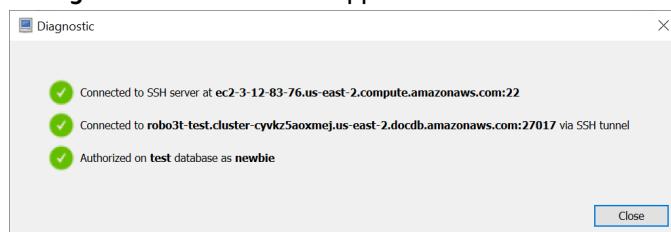
If you are on macOS Catalina or above, choose **Self-signed Certificate** as the **Authentication Method** because the macOS does not accept certificates with validity greater than 825 days.



7. Test the connection by choosing the **Test** button.



8. A **Diagnostic** window should appear with the test results. If everything is green, then close the box.



9. Now choose **Save**.
10. Now select your cluster and choose **Connect**.

Connecting to an Amazon DocumentDB Cluster from Studio 3T

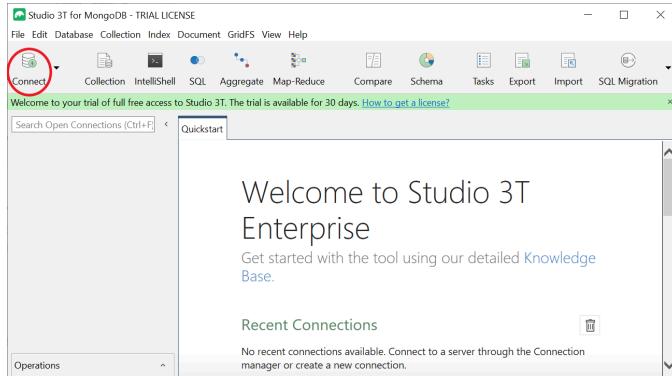
Studio 3T is a popular GUI and IDE for developers and data engineers who work with MongoDB. It offers several powerful capabilities Tree, Table and JSON views of your data, easy import/export in CSV, JSON, SQL and BSON/mongodump, flexible querying option, a visual drag-and-drop UI, a built-in mongo shell with auto-completion, an aggregation pipeline editor, and SQL query support.

Prerequisites

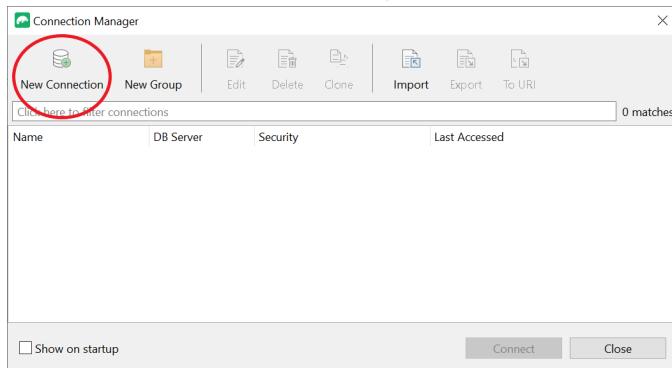
- If you don't already have an Amazon DocumentDB cluster running on Amazon EC2, follow the instructions on how to [Connect with Amazon EC2](#).
- If you don't have Studio 3T, [download and install it](#).

Connect with Studio 3T

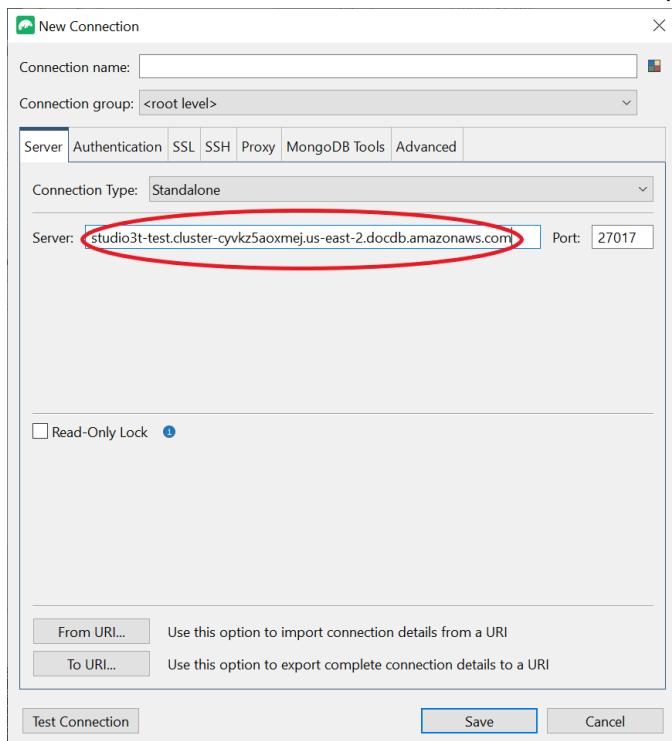
1. Choose **Connect** in the top left corner of the toolbar.



2. Choose New Connection in the top left corner of the toolbar.



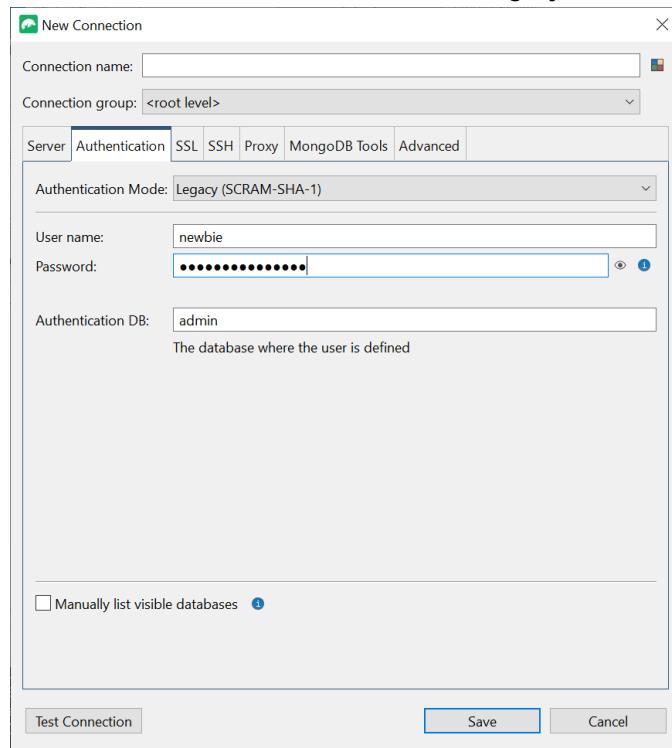
3. On the Server tab, in the Server field, enter the cluster endpoint information.



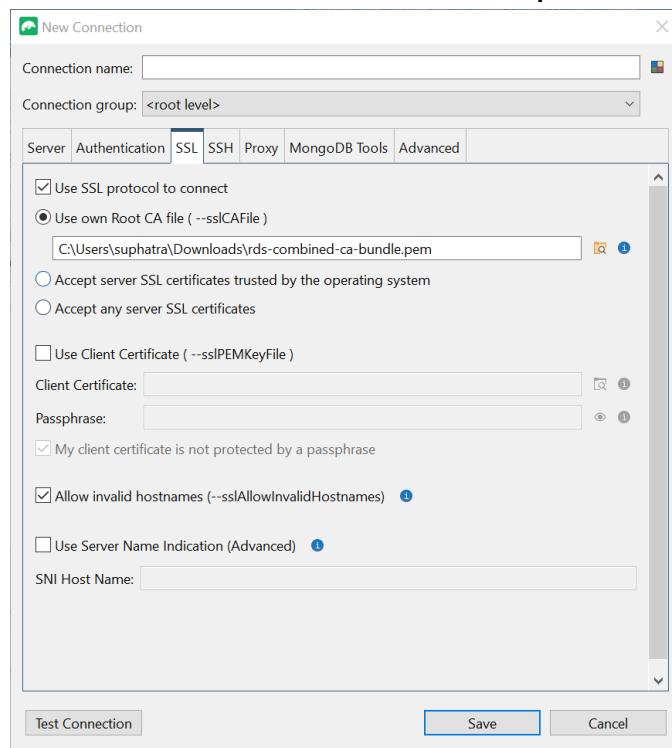
Note

Can't find your cluster endpoint? Just follow the steps [here](#).

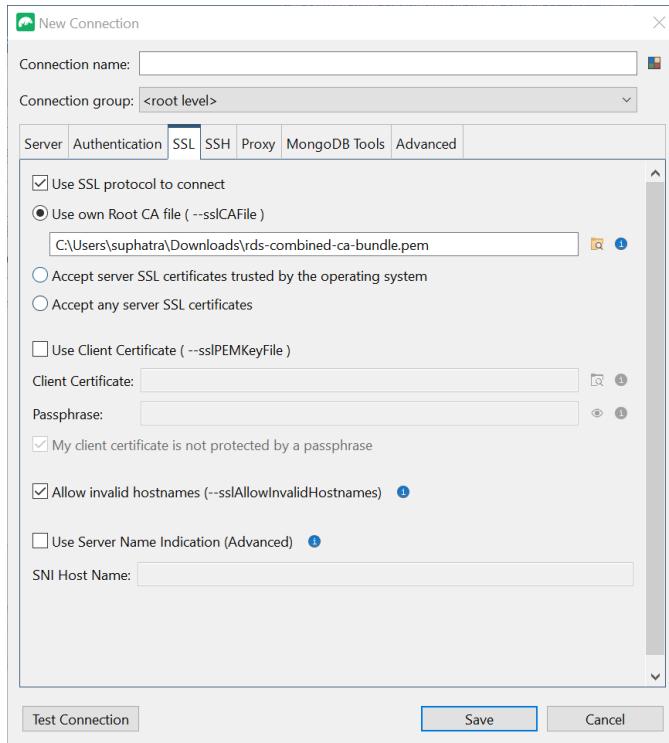
4. Choose the **Authentication** tab and select **Legacy** in the drop down menu for **Authentication Mode**.



5. Input your username and credentials in the **User name** and **Password** fields.
6. Choose the **SSL** tab and check the box **Use SSL protocol to connect**.



7. Choose **Use own Root CA file**. Then add the Amazon DocumentDB certificate (you can skip this step if SSL is disabled on your DocumentDB cluster). Check the box to allow **invalid hostnames**.

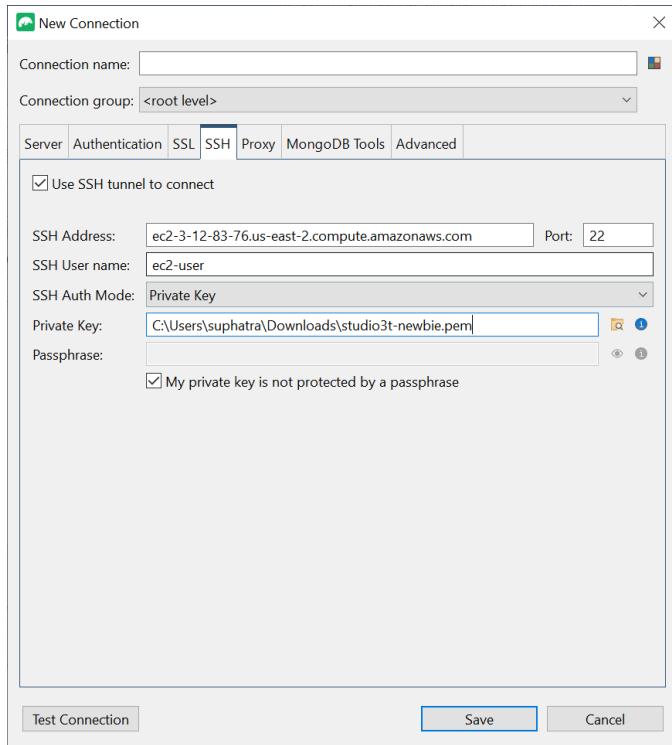


Note

Don't have the certificate? You can download it with the following command:
`wget https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem`

8. If you are connecting from a client machine outside the Amazon VPC, you need to create a SSH tunnel. You will do this in the **SSH** tab.
 - a. Check the box for **Use SSH tunnel** and input the SSH address in the **SSH Address** field. This is your instance Public DNS (IPV4). You can get this URL from your [Amazon EC2 Management Console](#).
 - b. Enter your username. This is the username of your Amazon EC2 instance
 - c. For **SSH Auth Mode**, select **Private Key**. In the **Private Key** field, choose the file finder icon to locate and choose the Private key of your Amazon EC2 instance. This is the .pem file (key pair) that you saved while creating your instance in Amazon EC2 Console.
 - d. If you are on Linux/macOS client machine, you might have to change the permissions of your private key using the following command:

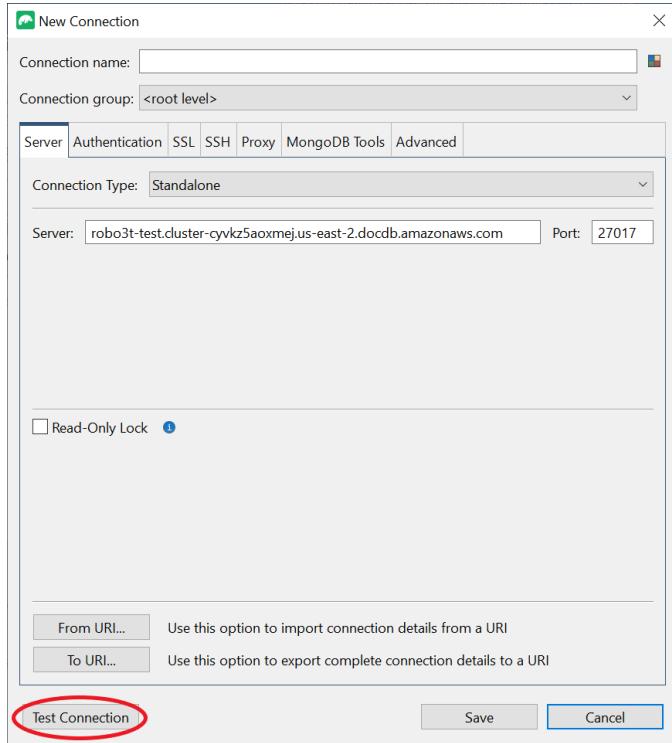
```
chmod 400 /fullPathToYourPemFile/<yourKey>.pem
```



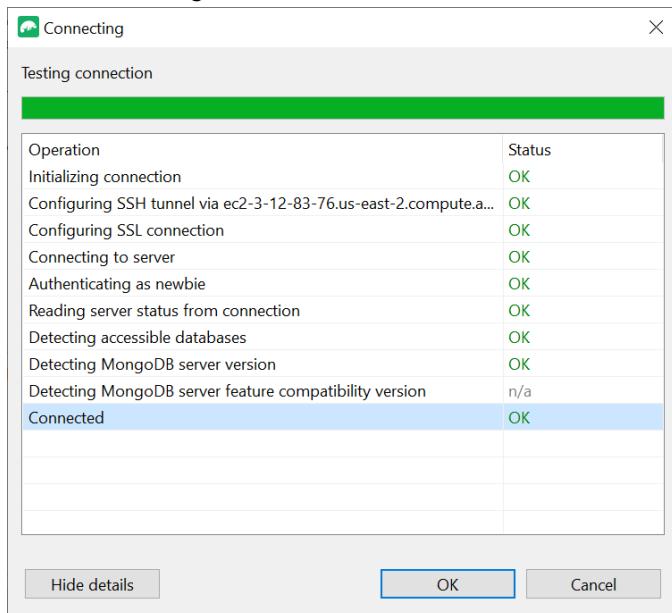
Note

This Amazon EC2 instance should be in the same Amazon VPC and security group as your DocumentDB cluster. You can get the SSH address, username and private key from your [Amazon EC2 Management Console](#).

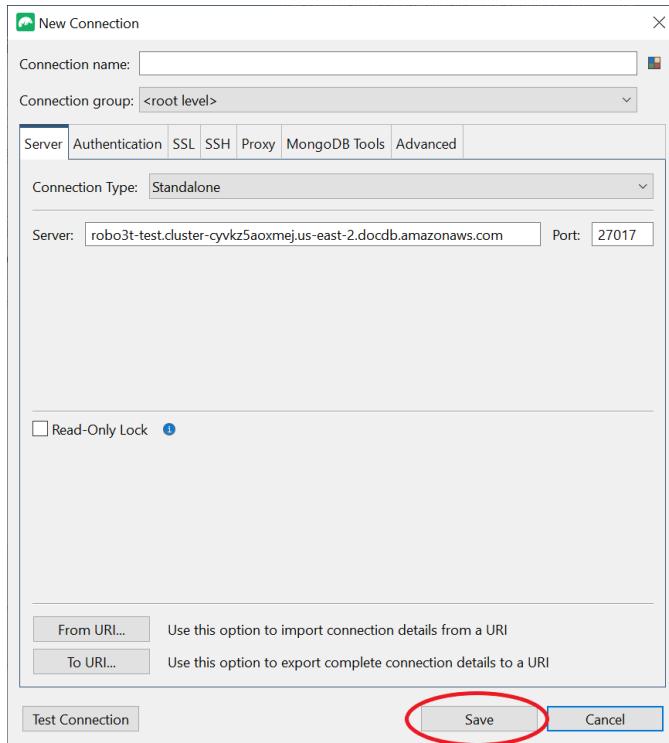
9. Now test your configuration by choosing the **Test connection** button.



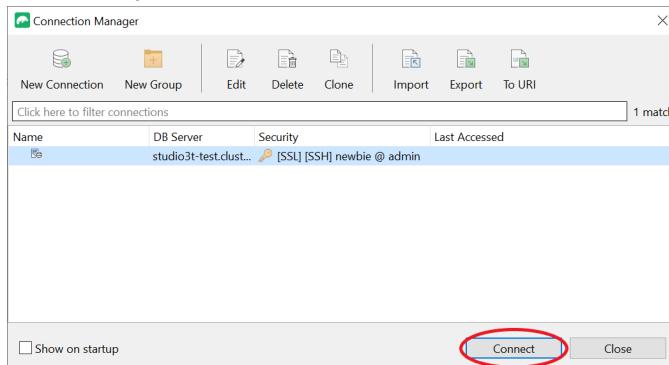
10. A diagnostic window should load a green bar to indicate the test was successful. Now choose **OK** to close out the diagnostic window.



11. Choose **Save** to save your connection for future use.



12. Now select your cluster and choose **Connect.**



Congratulations! You are now successfully connected to your Amazon DocumentDB cluster through Studio 3T.

Connect Using Amazon EC2

This section describes how to launch an Amazon DocumentDB (with MongoDB compatibility) cluster using Amazon EC2 and interact with it, using the mongo shell. The video below demonstrates the steps in this guide.

Prerequisites

Before you create your first Amazon DocumentDB cluster, you must do the following:

Create an Amazon Web Services (AWS) account

Before you can begin using Amazon DocumentDB, you must have an Amazon Web Services (AWS) account. The AWS account is free. You pay only for the services and resources that you use.

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open <https://portal.aws.amazon.com/billing/signup>.
2. Follow the online instructions.

Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, [assign administrative access to an administrative user](#), and use only the root user to perform [tasks that require root user access](#).

Set up the needed AWS Identity and Access Management (IAM) permissions.

Access to manage Amazon DocumentDB resources such as clusters, instances, and cluster parameter groups requires credentials that AWS can use to authenticate your requests. For more information, see [Identity and Access Management in Amazon DocumentDB \(p. 153\)](#).

1. In the search bar of the AWS Management Console, type in IAM and select **IAM** in the drop down menu that appears.
2. Once you're in the IAM console, select **Users** from the navigation pane.
3. Select your username.
4. Click the button **Add permissions**.
5. Select **Attach existing policies directly**.
6. Type **AmazonDocDBFullAccess** in the search bar and select it once it appears in the search results.
7. Click the blue button at the bottom that says **Next: Review**.
8. Click the blue button at the bottom that says **Add permissions**.

Create an Amazon Virtual Private Cloud (Amazon VPC)

Depending on which AWS Region you are in, you may or may not have a default VPC already created. If you don't have a default VPC, complete step 1 of the [Getting Started with Amazon VPC](#) in the [Amazon VPC User Guide](#). This will take less than five minutes.

Step 1: Create an Amazon EC2 Instance

In this step, you will create an Amazon EC2 instance in the same Region and Amazon VPC that you will later use to provision your Amazon DocumentDB cluster.

1. On the Amazon EC2 console, choose **Launch instance**.

Amazon DocumentDB Developer Guide

Step 1: Create an Amazon EC2 Instance

The screenshot shows the AWS EC2 console interface. On the left, there's a navigation sidebar with sections like Instances, Images, and Elastic Block Store. The main area has a heading 'Welcome to the new EC2 console' with a note about the redesign. Below it is a 'Resources' summary with counts for Running instances, Snapshots, Key pairs, Elastic IPs, Volumes, and Security groups. A callout box suggests using the AWS Launch Wizard for Microsoft SQL Server. The central part of the screen is titled 'Launch instance' with a sub-instruction: 'To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.' Below this is a large orange 'Launch instance' button, which is circled in red. To the right, there are tabs for 'Service health' and 'Zone status'. At the bottom, it says 'Region: US East (N. Virginia)'.

2. Locate Amazon Linux 2 AMI and choose Select.

This screenshot shows the 'Choose an Amazon Machine Image (AMI)' step. It lists several AMI options under 'Quick Start'. The 'Amazon Linux 2 AMI (HVM, SSD Volume Type)' is selected, and its details are displayed below: 'ami-007a07144b0f1920b (64-bit Arm)'. It's described as 'Amazon Linux 2 is a 2-year supported AMI based on the latest Amazon Linux 2 release. It is built for optimal performance on Amazon EC2, system 219, G2G, T3, Gts, 2.26, Rds, 2.29.1, and the latest software packages through extras.' There are two 'Select' buttons at the bottom right, both highlighted with red circles.

3. Choose the t3.micro instance type.

This screenshot shows the 'Step 2: Choose an Instance Type' step. It displays a table of instance types: t3a.2xlarge, t3.nano, t3.micro (which is selected and highlighted with a red circle), and t3.small. The table includes columns for instance type, memory, and vCPUs.

	General purpose	t3a.2xlarge	8	32
<input type="checkbox"/>	General purpose	t3.nano	2	0.5
<input checked="" type="checkbox"/>	General purpose	t3.micro	2	1
<input type="checkbox"/>	General purpose	t3.small	2	2

4. Choose Review and Launch, which will allow you to skip to the console's Step 7: Review Instance Launch page.

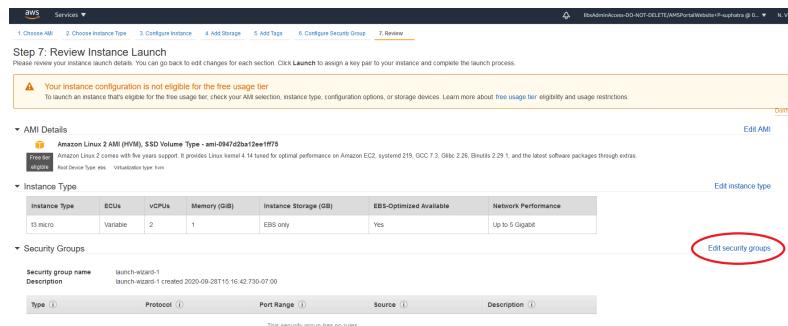
This screenshot shows the 'Review and Launch' step. It contains a table of storage options and their details. At the bottom, there are 'Cancel', 'Previous', 'Review and Launch' (which is circled in red), and 'Next: Configure Instance Details' buttons.

EBS only	Yes	Up to 5 Gigabit	Yes
EBS only	Yes	Up to 5 Gigabit	Yes
EBS only	Yes	Up to 5 Gigabit	Yes
1 x 75 (SSD)	Yes	Up to 10 Gigabit	Yes
1 x 150 (SSD)	Yes	Up to 10 Gigabit	Yes

5. Under Security Groups, choose Edit security groups.

Amazon DocumentDB Developer Guide

Step 1: Create an Amazon EC2 Instance



Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

AMI Details

Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-0947d2ba12ee1ff75

Instance Type

Instance Type	ECUs	vCPUs	Memory (GB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t1.micro	Variable	2	1	EBS only	Yes	Up to 5 Gigabit

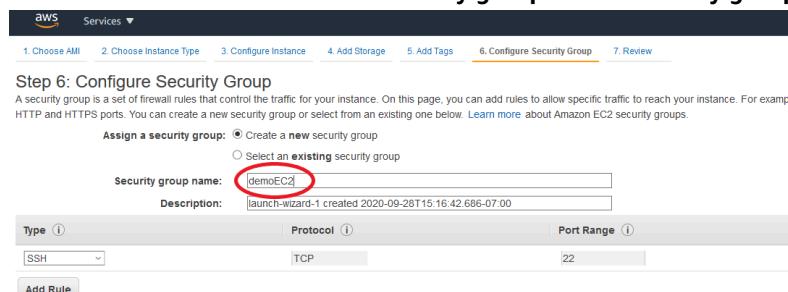
Security Groups

Security group name: Launch-wizard-1
Description: launch-wizard-1 created 2020-09-28T15:16:42.730-07:00

Type	Protocol	Port Range	Source	Description
This security group has no rules.				

[Edit security groups](#)

6. This will default to **Create a new security group**. In the **Security group name** field, write demoEC2.



Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: Create a new security group
 Select an existing security group

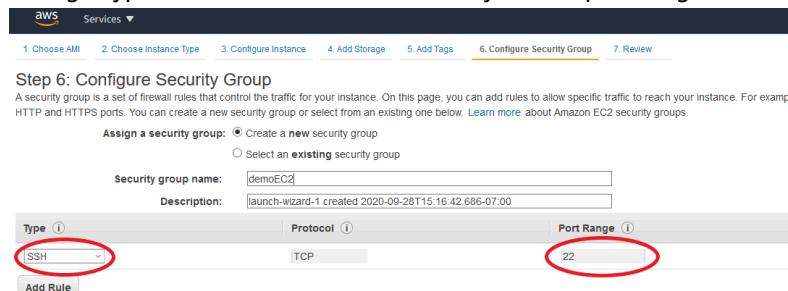
Security group name: (demoEC2)

Description: launch-wizard-1 created 2020-09-28T15:16:42.686-07:00

Type	Protocol	Port Range
SSH	TCP	22

[Add Rule](#)

7. Change Type to SSH. This will automatically set the port range to 22.



Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: Create a new security group
 Select an existing security group

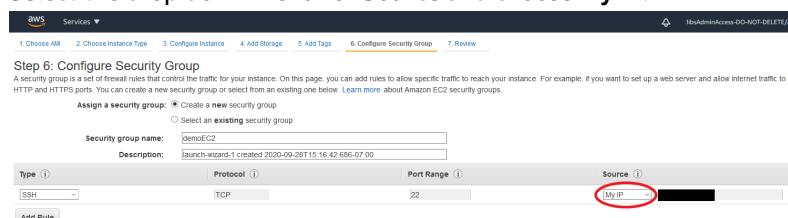
Security group name: (demoEC2)

Description: launch-wizard-1 created 2020-09-28T15:16:42.686-07:00

Type	Protocol	Port Range
SSH	TCP	22

[Add Rule](#)

8. Select the drop down menu for **Source** and choose **My IP**.



Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow internet traffic to it, you can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: Create a new security group
 Select an existing security group

Security group name: (demoEC2)

Description: launch-wizard-1 created 2020-09-28T15:16:42.686-07:00

Type	Protocol	Port Range	Source
SSH	TCP	22	My IP

[Add Rule](#)

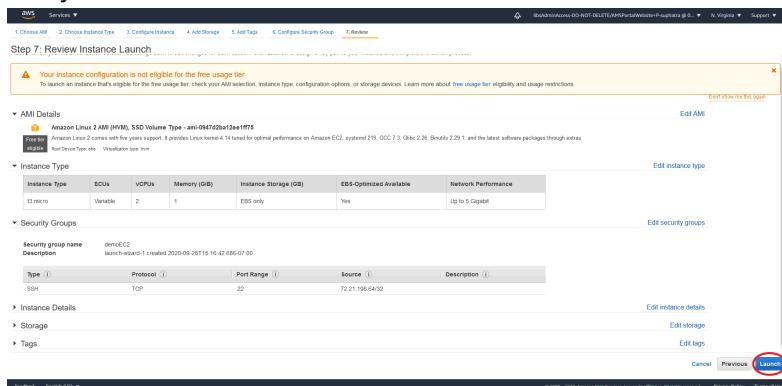
Note

You can only access the demoEC2 security group from your current IP address. If your IP address changes, you must update the security group.

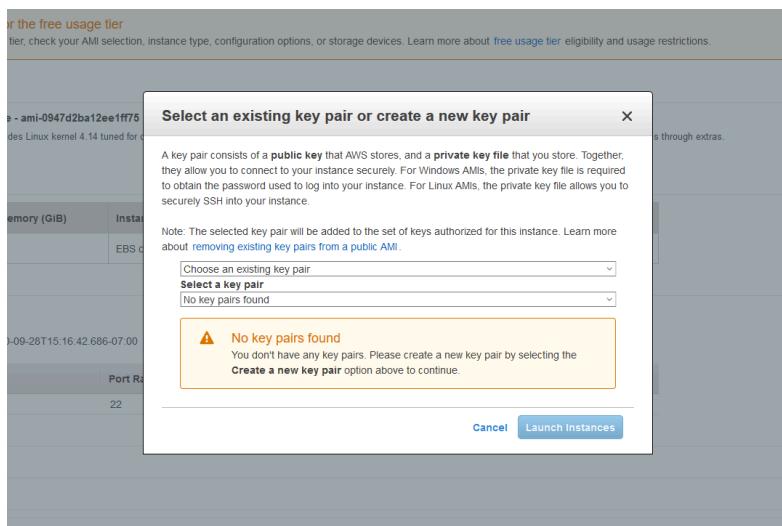
9. Now choose **Review and Launch**. You should now see the demoEC2 security group in the **Security Groups** section.



10. Verify the information and choose **Launch**.



11. A window will pop up titled **Select an existing key pair or create a new key pair**. It will look like this:



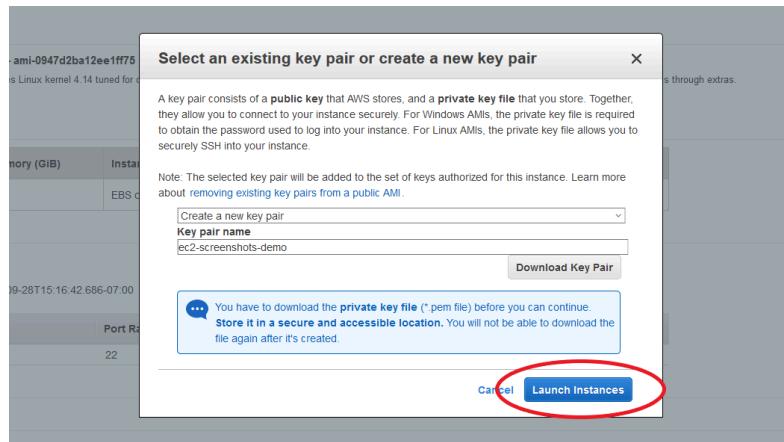
You must provide an Amazon EC2 key pair. If you do have an Amazon EC2 key pair:

- Select a key pair, choose your key pair from the list.
- You must already have the private key file (.pem file) available to log in to your Amazon EC2 instance.

If you do not have an Amazon EC2 key pair:

- a. Choose **Create a new key pair**.
- b. Write a name for the key bar in the field **Key pair name**.
- c. Download the private key file (.pem file). You need this file later when you log in to your Amazon EC2 instance.

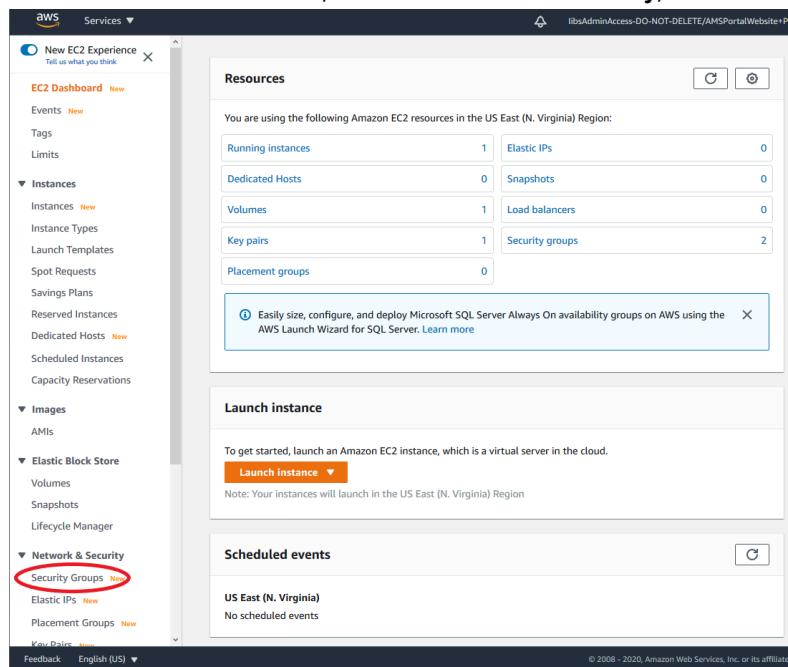
12. Choose **Launch Instances.**



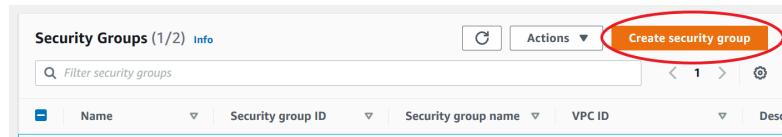
Step 2: Create a security group

You will now create a new security group in your default Amazon VPC. The security group `demoDocDB` enables you to connect to your Amazon DocumentDB cluster on port 27017 (the default port for Amazon DocumentDB) from your Amazon EC2 instance.

1. On the Amazon EC2 console, under **Network and Security, choose **Security groups**.**



2. Choose **Create security group.**



3. For **Security group name**, enter demoDocDB.

Create security group Info

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, provide a name and optional description.

Basic details

Security group name Info
demoDocDB

Name cannot be edited after creation.

Description Info
Allows SSH access to developers

4. For **Description**, enter a description.

Create security group Info

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, provide a name and optional description.

Basic details

Security group name Info
demoDocDB

Name cannot be edited after creation.

Description Info
Connecting to DocDB with EC2

VPC Info
vpc-3626e24b

5. For **VPC**, accept the usage of your default VPC.

6. In the **Inbound rules** section, choose **Add rule**.

Inbound rules Info

This security group has no inbound rules.

Add rule

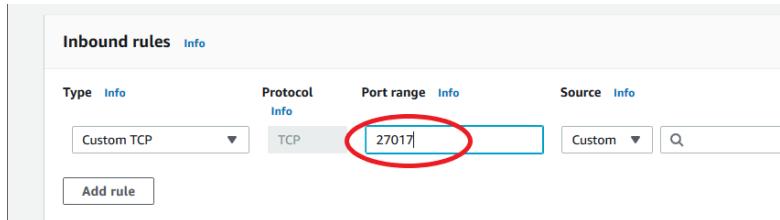
7. For **Type**, choose **Custom TCP Rule**.

Inbound rules Info

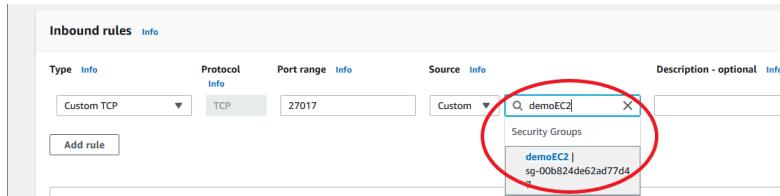
Type <small>Info</small>	Protocol <small>Info</small>	Port range <small>Info</small>	Source <small>Info</small>
Custom TCP	TCP	0	Custom

Add rule

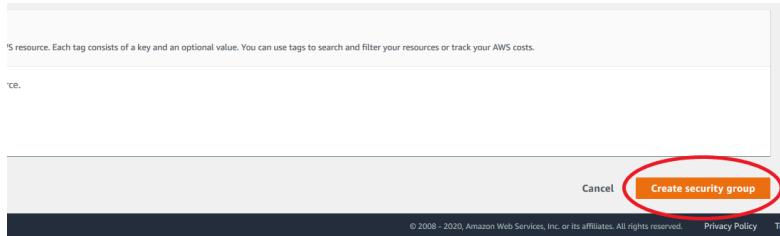
8. For **Port range**, enter 27017.



9. For Destination, choose **Custom**. In the field next to it, search for the security group you just made called demoEC2. You may need to refresh your browser for the Amazon EC2 console to auto-populate the demoEC2 source name.



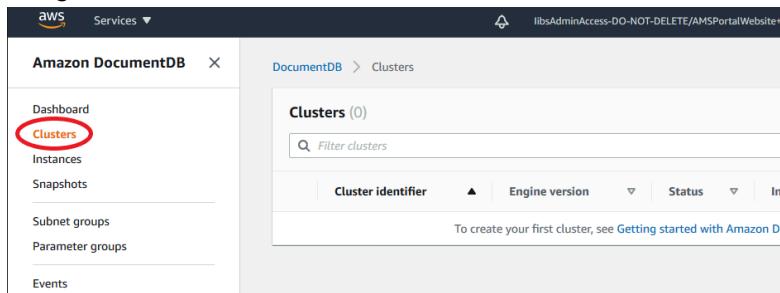
10. Accept all other defaults and choose **Create security group**.



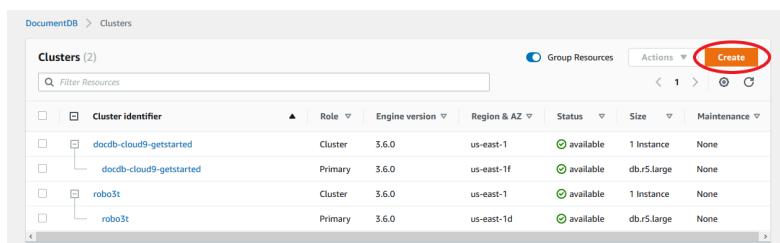
Step 3: Create an Amazon DocumentDB Cluster

While the Amazon EC2 instance is being provisioned, you will create your Amazon DocumentDB cluster.

1. Navigate to the Amazon DocumentDB console and choose **Clusters** from the navigation pane.



2. Choose **Create**.



3. For **Number of instances**, choose **1**. This will minimize cost. Leave other settings at their default.

Create Amazon DocumentDB cluster

Configuration

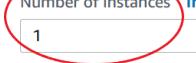
Cluster identifier [Info](#)
Specify a unique cluster identifier.

Engine version

Instance class [Info](#)

2 vCPUs 16GiB RAM

Number of instances [Info](#)



4. For **Authentication**, enter a username and password. Important: You will need this username and password to authenticate your cluster in a later step.

Authentication

Master username [Info](#)
Specify an alphanumeric string that defines the login ID for the master user.

Master username must start with a letter and contain 1 to 63 characters

Master password [Info](#) Confirm master password [Info](#)

Master password must be at least eight characters long and cannot contain a / (slash), " (double quote) or @ (at symbol).

5. Turn on **Show advanced settings**.

Amazon DocumentDB requires permissions to manage AWS resources on your behalf. By clicking Create cluster, you grant permission for Amazon DocumentDB to create a service-linked role in AWS IAM that contains the required permissions.

Show advanced settings Hide advanced settings

[Cancel](#) [Create cluster](#)

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#)

6. In the **Network settings** section, for **Amazon VPC security groups**, choose **demoDocDB**.

Network settings

Virtual Private Cloud (VPC) [Info](#)
VPC defines the virtual networking environment for this cluster.

Only VPCs with a corresponding subnet group are listed. Once a cluster is created, the VPC cannot be changed.

Select VPC security groups
 demoEC2 (VPC)
 demoDocDB (VPC)
 default (VPC)

[Select VPC security groups](#)

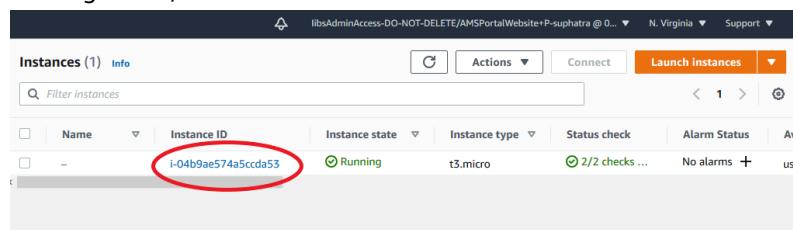
7. Choose Create cluster.



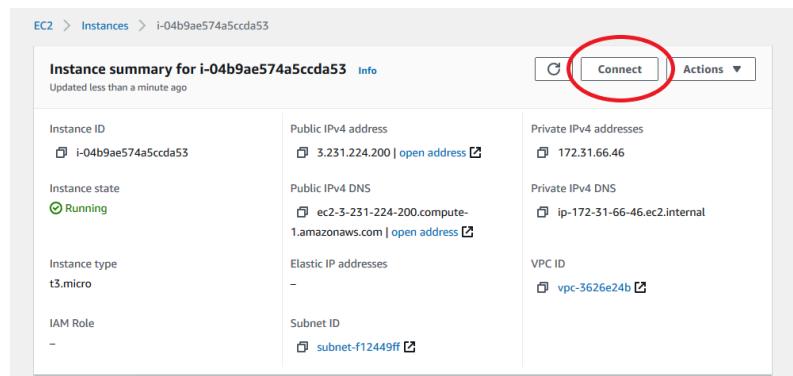
Step 4: Connect to your Amazon EC2 instance

To install the mongo shell, you must first connect to your Amazon EC2 instance. Installing the mongo shell enables you to connect to and query your Amazon DocumentDB cluster. Complete the following steps:

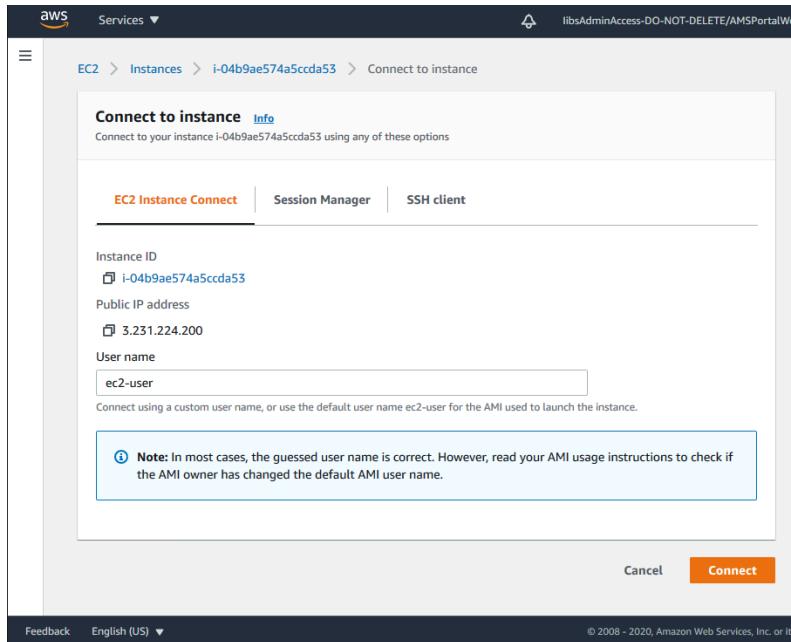
1. On the Amazon EC2 console, navigate to your instances and see if the instance you just created is running. If it is, **select the instance**.



2. Choose **Connect**.



3. You will see three options for your connection method: Amazon EC2 Instance Connect, Session Manager, or SSH client. You must choose one and follow its instructions.



Note

If your IP address changed after you started this walkthrough, or you are coming back to your environment at a later time, you must update your demoEC2 security group inbound rule to enable inbound traffic from your new API address.

Step 5: Install the mongo shell

You can now install the mongo shell, which is a command-line utility that you use to connect and query your Amazon DocumentDB cluster. Follow the instructions below to install the mongo shell for your operating system.

On Amazon Linux

To install the mongo shell on Amazon Linux

1. Create the repository file. At the command line of your EC2 instance, execute the follow command:

```
echo -e "[mongodb-org-4.0] \nname=MongoDB Repository\nbaseurl=https://repo.mongodb.org/yum/amazon/2013.03/mongodb-org/4.0/x86_64/\npgpcheck=1 \nenabled=1 \npgpkey=https://www.mongodb.org/static/pgp/server-4.0.asc" | sudo tee /etc/yum.repos.d/mongodb-org-4.0.repo
```

2. When it is complete, install the mongo shell by executing the following command:

```
sudo yum install -y mongodb-org-shell
```

On Ubuntu 18.04

To install the mongo shell on Ubuntu 18.04

1. Import the public key that will be used by the package management system.

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv  
2930ADAE8CAF5059EE73BB4B58712A2291FA4AD5
```

2. Create the list file /etc/apt/sources.list.d/mongodb-org-3.6.list for MongoDB using the command appropriate for your version of Ubuntu.

Ubuntu 18.04

```
echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/3.6 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.6.list
```

Note

The command above will install the mongo 3.6 shell for both Bionic and Xenial.

3. Reload the local package database using the following command:

```
sudo apt-get update
```

4. Install the MongoDB shell.

```
sudo apt-get install -y mongodb-org-shell
```

For information about installing earlier versions of MongoDB on your Ubuntu system, see [Install MongoDB Community Edition on Ubuntu](#).

On other operating systems

To install the mongo shell on other operating systems, see [Install MongoDB Community Edition](#) in the MongoDB documentation.

Step 6: Manage Amazon DocumentDB TLS

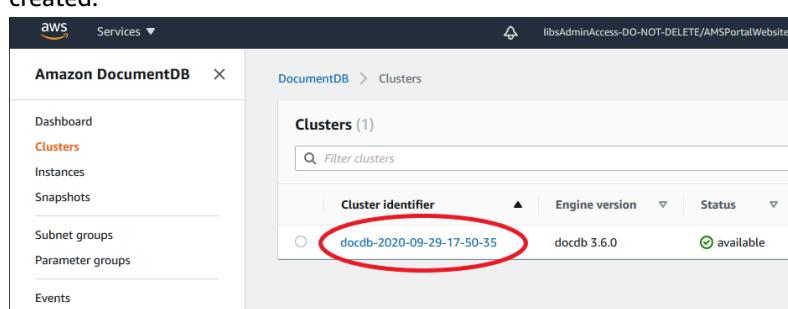
Download the CA certificate for Amazon DocumentDB with the following code: wget <https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem>

Note

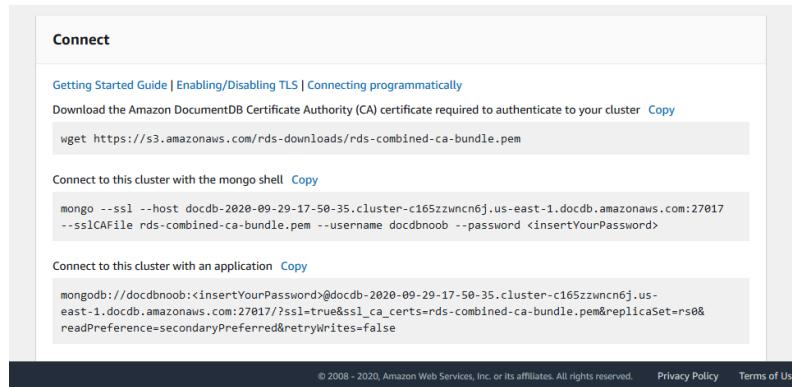
Transport Layer Security (TLS) is enabled by default for any new Amazon DocumentDB clusters. For more information, see [Managing Amazon DocumentDB Cluster TLS Settings](#).

Step 7: Connect to your Amazon DocumentDB cluster

1. On the Amazon Document DB console, under Clusters, locate your cluster. Choose the cluster you created.

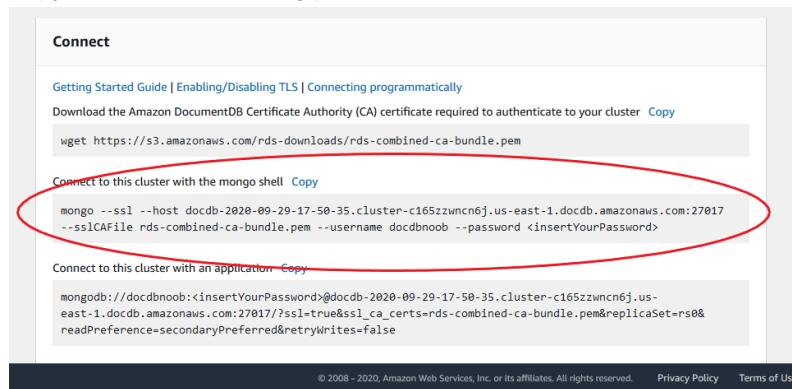


2. Navigate to the **Connection** box. It will look like this.



The screenshot shows the 'Connect' section of the Amazon DocumentDB developer guide. It includes three main sections: 1) 'Getting Started Guide | Enabling/Disabling TLS | Connecting programmatically'. 2) 'Download the Amazon DocumentDB Certificate Authority (CA) certificate required to authenticate to your cluster' with a 'Copy' button and a wget command. 3) 'Connect to this cluster with the mongo shell' with a 'Copy' button and a mongo command. Below these are instructions for connecting with an application. At the bottom, there's a note about the certificate, a 'Privacy Policy' link, and a 'Terms of Use' link.

Copy the connection string provided.



This screenshot is identical to the one above, showing the 'Connect' section. However, the mongo shell connection string (the second section) is circled with a red oval for emphasis.

After you paste it into your terminal and make the following changes to it: first, make sure you have the correct username in the string. Then, omit <insertYourPassword> so that you are prompted for the password by the mongo shell when you connect. Your connection string should look similar to the following:

```
mongo --ssl host docdb-2020-02-08-14-15-11.
cluster.region.docdb.amazonaws.com:27107 --sslCAFile rds-combined-ca-
bundle.pem --username demoUser --password
```

3. Now press enter in your terminal. You will now be prompted for your password. Enter your password.
4. When you enter your password and can see the rs0:PRIMARY> prompt, you are successfully connected to your Amazon DocumentDB cluster.

Having problems connecting? See [Troubleshooting Amazon DocumentDB](#).

Step 8: Insert and query data

Now that you are connected to your cluster, you can run a few queries to get familiar with using a document database.

1. To insert a single document, enter the following:

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

2. You get the following output:

```
WriteResult({ "nInserted" : 1 })
```

3. You can read the document that you wrote with the `findOne()` command (because it only returns a single document). Input the following:

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

4. You get the following output:

```
{ "_id" : ObjectId("5e401fe56056fda7321fdb67"), "hello" : "DocumentDB" }
```

5. To perform a few more queries, consider a gaming profiles use case. First, insert a few entries into a collection titled `profiles`. Input the following:

```
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}  
])
```

6. You get the following output:

```
{ "acknowledged" : true, "insertedIds" : [ 1, 2, 3, 4 ] }
```

7. Use the `find()` command to return all the documents in the `profiles` collection. Input the following:

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

8. You will get an output that will match the data you typed in Step 5.

9. Use a query for a single document using a filter. Input the following:

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

10. You should get back this output:

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

11. Now let's try to find a profile and modify it using the `findAndModify` command. We'll give the user Matt an extra ten points with the following code:

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

12. You get the following output (note that his score hasn't increased yet):

```
{  
    "_id" : 1,  
    "name" : "Matt",  
    "status" : "active",  
    "level" : 12,  
    "score" : 202  
}
```

13. You can verify that his score has changed with the following query:

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

14. You get the following output:

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

Step 9: Explore

Congratulations! You have successfully completed the Quick Start Guide to Amazon DocumentDB.

What's next? Learn how to fully leverage this powerful database with some of its popular features:

- [Managing Amazon DocumentDB](#)
- [Scaling](#)
- [Backing up and restoring](#)

Note

To save on cost, you can either stop your Amazon DocumentDB cluster to reduce costs or delete the cluster. By default, after 30 minutes of inactivity, your AWS Cloud9 environment will stop the underlying Amazon EC2 instance.

Connect Using Amazon DocumentDB JDBC Driver

The JDBC driver for Amazon DocumentDB provides an SQL-relational interface for developers and enables connectivity from BI tools such as Tableau and DbVisualizer.

For more detailed information, refer to the [Amazon DocumentDB JDBC Driver documentation on GitHub](#).

Topics

- [Getting Started \(p. 523\)](#)
- [Connect to Amazon DocumentDB from Tableau Desktop \(p. 524\)](#)
- [Connect to Amazon DocumentDB from DbVisualizer \(p. 527\)](#)
- [Automatic schema generation \(p. 528\)](#)
- [SQL Support and Limitations \(p. 534\)](#)
- [Troubleshooting \(p. 534\)](#)

Getting Started

Step 1. Create Amazon DocumentDB Cluster

If you do not have an Amazon DocumentDB cluster created, then create one using the instructions in the [Getting Started](#) section of the Amazon DocumentDB Developer Guide.

Note

DocumentDB is a Virtual Private Cloud (VPC) only service. If you are connecting from a local machine, outside the cluster's VPC, you will need to create an SSH connection to an Amazon EC2 instance. In this case, launch your cluster using the instructions in [Connect with EC2](#).

See [Using an SSH Tunnel to Connect to Amazon DocumentDB](#) for more information on SSH tunneling and when you might need it.

Step 2. JRE or JDK Installation

Depending on your BI application, you may need to ensure a 64-bit JRE or JDK installation version 8 or later is installed on your computer. You can download the Java SE Runtime Environment 8 [here](#).

Step 3. Download the DocumentDB JDBC Driver

Download the DocumentDB JDBC driver from [here](#). The driver is packaged as a single JAR file (e.g. documentdb-jdbc-1.0.0-all.jar).

Step 4. Using an SSH Tunnel to Connect to Amazon DocumentDB

Amazon DocumentDB (with MongoDB compatibility) clusters are deployed within an Amazon Virtual Private Cloud (Amazon VPC). They can be accessed directly by Amazon EC2 instances or other AWS services that are deployed in the same Amazon VPC. Additionally, Amazon DocumentDB can be accessed by EC2a instances or other AWS services in different VPCs in the same AWS Region or other Regions via VPC peering.

You can use SSH tunneling (also known as port forwarding) to access your Amazon DocumentDB resources, from outside the cluster's VPC. This will be the case for most users not running their application on a VM in the same VPC as the DocumentDB cluster.

To create an SSH tunnel, you need an Amazon EC2 instance running in the same Amazon VPC as your Amazon DocumentDB cluster. You can either use an existing EC2 instance in the same VPC as your cluster or create one. You can set up an SSH tunnel to the Amazon DocumentDB cluster sample-cluster.node.us-east-1.docdb.amazonaws.com by running the following command on your local computer.

```
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
```

The -L flag is used for forwarding a local port. This is a prerequisite for connecting to any BI tool running on a client outside your VPC. Once you run the step above you can move on to the next steps for the BI tool of your choice.

For further information on SSH tunneling , please refer to the documentation on [Using an SSH tunnel to connect to Amazon DocumentDB](#).

Connect to Amazon DocumentDB from Tableau Desktop

Topics

- [Adding the Amazon DocumentDB JDBC Driver \(p. 524\)](#)
- [Connecting to Amazon DocumentDB Using Tableau - SSH Tunnel \(p. 525\)](#)

Adding the Amazon DocumentDB JDBC Driver

To connect to Amazon DocumentDB from Tableau Desktop you must download and install the DocumentDB JDBC driver and the DocumentDB Tableau connector.

1. Download the DocumentDB JDBC driver JAR file and copy it to one of these directories according to your operating system:

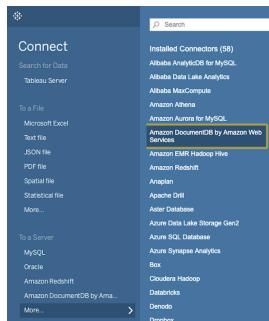
- Windows - C:\Program Files\Tableau\Drivers
 - MacOS - ~/Library/Tableau/Drivers
2. Download the DocumentDB Tableau connector (a TACO file) and copy it to your *My Tableau Repository/Connectors* directory.
 - Windows - C:\Users\[user]\Documents\My Tableau Repository\Connectors
 - MacOS - /Users/[user]/Documents/My Tableau Repository/Connectors

For additional information, refer to the [Tableau documentation](#).

Connecting to Amazon DocumentDB Using Tableau - SSH Tunnel

To connect to Tableau from a client machine outside of the VPC of your DocumentDB cluster, you must setup an SSH tunnel before following the steps below:

1. Launch the Tableau Desktop application.
2. Navigate to **Connect > To A Server > More**.
3. Choose **Amazon DocumentDB by Amazon Web Services** under **Installed Connectors**.

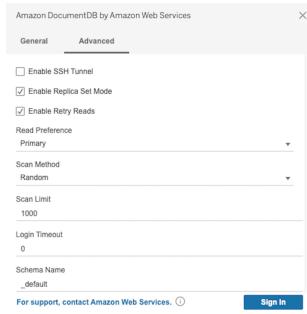


Connecting to Amazon DocumentDB Using Tableau - External SSH Tunnel

1. Enter the required connection parameters **Hostname**, **Port**, **Database**, **Username** and **Password**. The connection parameters in the example below are equivalent to the JDBC connection string :

```
jdbc:documentdb://localhost:27019/test?
tls=true&tlsAllowInvalidHostnames=true&scanMethod=random&scanLimit=1000&loginTimeoutSe
with the username and password parameters passed separately in a properties collection. For more
information on connection string parameters, refer to the Amazon DocumentDB JDBC Driver github
documentation.
```

2. (Optional) More advanced options can be found on the **Advanced** tab.



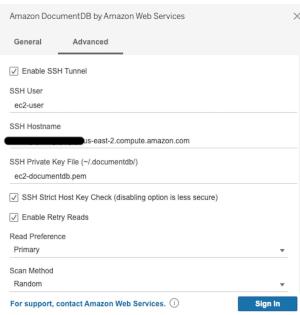
3. Choose **Sign in**.

Connecting to Amazon DocumentDB Using Tableau - Internal SSH Tunnel

Note

If you prefer to not setup the SSH tunnel using a terminal, you can use the Tableau GUI to specify your EC2 instance details which the JDBC driver will inherently use to create a SSH tunnel.

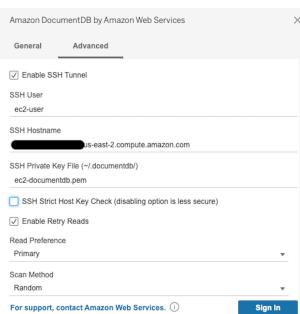
1. On the Advanced tab, choose the **Enable SSH Tunnel option** to review further properties.



2. Enter the **SSH User**, **SSH Hostname**, and **SSH Private Key File**.
3. (Optional) You can disable the **SSH Strict Host Key Check** option which bypasses the host key check against a known hosts file.

Note

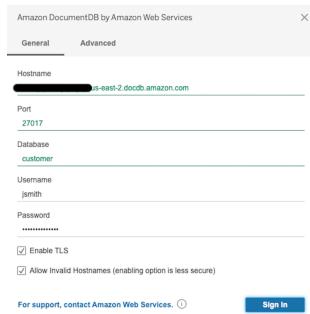
Disabling this option is less secure as it can lead to a [man-in-the-middle](#) attack.



4. Enter the required parameters; **Hostname**, **Port**, **Database**, **Username** and **Password**.

Note

Make sure you use the DocumentDB cluster endpoint and not localhost when using the internal SSH tunnel option.



5. Choose **Sign In**.

Connect to Amazon DocumentDB from DbVisualizer

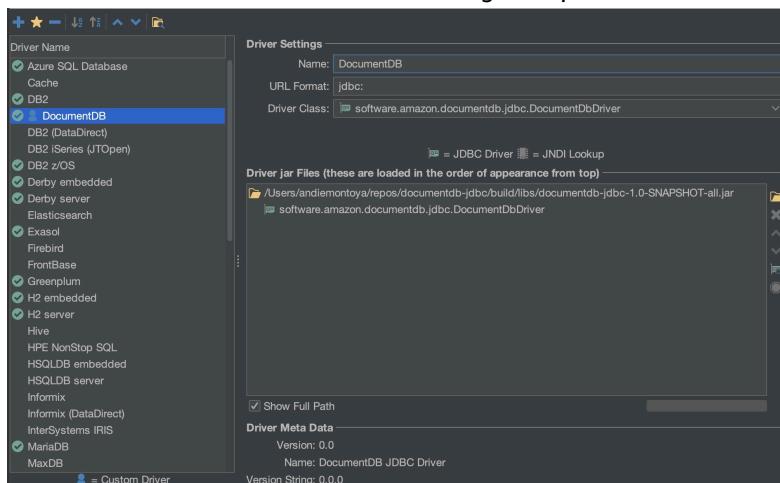
Topics

- [Adding the Amazon DocumentDB JDBC Driver \(p. 527\)](#)
- [Connecting to Amazon DocumentDB Using DbVisualizer \(p. 528\)](#)

Adding the Amazon DocumentDB JDBC Driver

To connect to Amazon DocumentDB from DbVisualizer you must first import the Amazon DocumentDB JDBC Driver

1. Start the DbVisualizer application and navigate to the menu path: **Tools > Driver Manager...**
2. Choose **+** (or in the menu, select **Driver > Create Driver**).
3. Set **Name** to DocumentDB.
4. Set **URL Format** to `jdbc:documentdb://<host>[:port]/<database>[?option=value[&option=value[...]]]`
5. Choose the **folder** button and then select the Amazon DocumentDB JDBC driver JAR file and choose the **Open** button.
6. Verify that the **Driver Class** field is set to `software.amazon.documentdb.jdbc.DocumentDbDriver`. Your Driver Manager settings for **DocumentDB** should look like the following example.



7. Close the dialog. The Amazon DocumentDB JDBC driver will be setup and ready to use.

Connecting to Amazon DocumentDB Using DbVisualizer

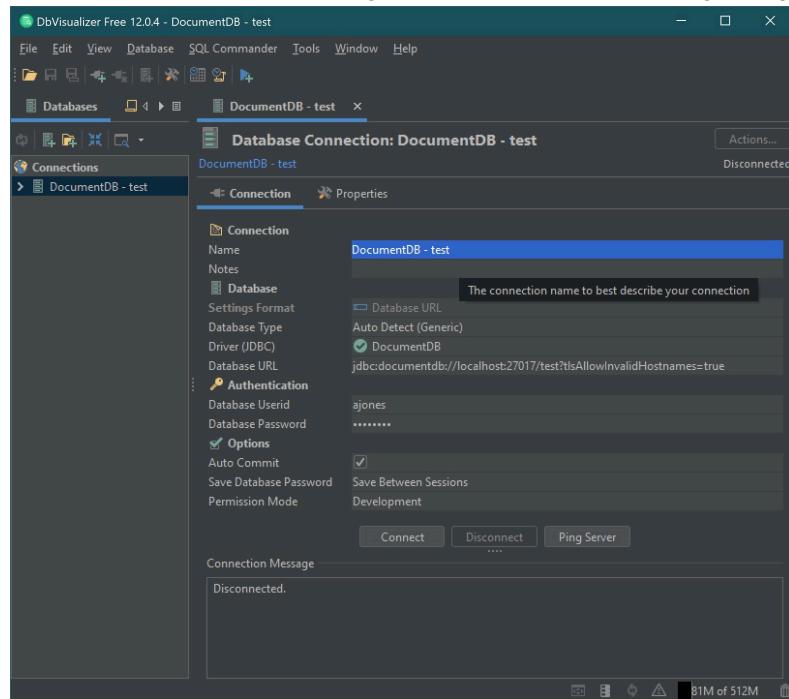
Connect to Amazon DocumentDB Using DbVisualizer

1. If you are connecting from outside the Amazon DocumentDB cluster's VPC, ensure you have setup an SSH tunnel.
2. Choose **Database > Create Database Connection** from the top level menu.
3. Enter a descriptive name for the **Name** field.
4. Set **Driver (JDBC)** to the DocumentDB driver you created in the previous section.
5. Set **Database URL** to your JDBC connection string.

For example: `jdbc:documentdb://localhost:27017/database?tlsAllowInvalidHostnames=true`

6. Set **Database Userid** to your Amazon DocumentDB user ID.
7. Set **Database Password** to the corresponding password for the user ID.

Your Database Connection dialog should look like the following dialog:



8. Choose **Connect**.

Automatic schema generation

Amazon DocumentDB is a document database and therefore does not have the concept of tables and schema. However, BI tools such as Tableau will expect the database it connects to present a schema. Specifically, when the JDBC driver connection needs to get the schema for the collection in the database, it will poll for all the collections in the database. The driver will determine if a cached version of the schema for that collection already exists. If a cached version does not exist, it will sample the collection for documents and create a schema based on the following behavior.

Topics

- [Schema Generation Limitations \(p. 529\)](#)

- [Scanning Method Options \(p. 529\)](#)
- [DocumentDB Data Types \(p. 529\)](#)
- [Mapping Scalar Document Fields \(p. 530\)](#)
- [Object and Array Data Type Handling \(p. 532\)](#)

Schema Generation Limitations

The DocumentDB JDBC driver imposes a limit on the length of identifiers at 128 characters. The schema generator may truncate the length of generated identifiers (table names and column names) to ensure they fit that limit.

Scanning Method Options

The sampling behavior can be modified using connection string or data source options.

- `scanMethod=<option>`
 - `random` - (default) - *The sample documents are returned in random order.*
 - `idForward` - The sample documents are returned in order of id.
 - `idReverse` - The sample documents are returned in reverse order of id.
 - `all` - Sample all the documents in the collection.
- `scanLimit=<n>` - The number of documents to sample. The value must be a positive integer. The default value is 1000. If `scanMethod` is set to `all`, this option is ignored.

DocumentDB Data Types

The DocumentDB server supports a number of MongoDB data types. Listed below are the supported data types, and their associated JDBC data types.

MongoDB Data Type	Supported in DocumentDB	JDBC Data Type
Binary Data	Yes	VARBINARY
Boolean	Yes	BOOLEAN
Double	Yes	DOUBLE
32-bit Integer	Yes	INTEGER
64-bit Integer	Yes	BIGINT
String	Yes	VARCHAR
ObjectId	Yes	VARCHAR
Date	Yes	TIMESTAMP
Null	Yes	VARCHAR
Regular Expression	Yes	VARCHAR
Timestamp	Yes	VARCHAR
MinKey	Yes	VARCHAR

MongoDB Data Type	Supported in DocumentDB	JDBC Data Type
MaxKey	Yes	VARCHAR
Object	Yes	virtual table
Array	Yes	virtual table
Decimal128	No	DECIMAL
JavaScript	No	VARCHAR
JavaScript (with scope)	No	VARCHAR
Undefined	No	VARCHAR
Symbol	No	VARCHAR
DBPointer (4.0+)	No	VARCHAR

Mapping Scalar Document Fields

When scanning a sample of documents from a collection, the JDBC driver will create one or more schema to represent the samples in the collection. In general, a scalar field in the document maps to a column in the table schema. For example, in a collection named team, and a single document { "_id" : "112233", "name" : "Alastair", "age": 25 }, this would map to schema:

Table Name	Column Name	Data Type	Key
team	<i>team id</i>	VARCHAR	PK
team	name	VARCHAR	
team	age	INTEGER	

Data Type Conflict Promotion

When scanning the sampled documents, it is possible that the data types for a field are not consistent from document to document. In this case, the JDBC driver will promote the JDBC data type to a common data type that will suit all data types from the sampled documents.

For Example:

```
{
  "_id" : "112233",
  "name" : "Alastair", "age" : 25
}

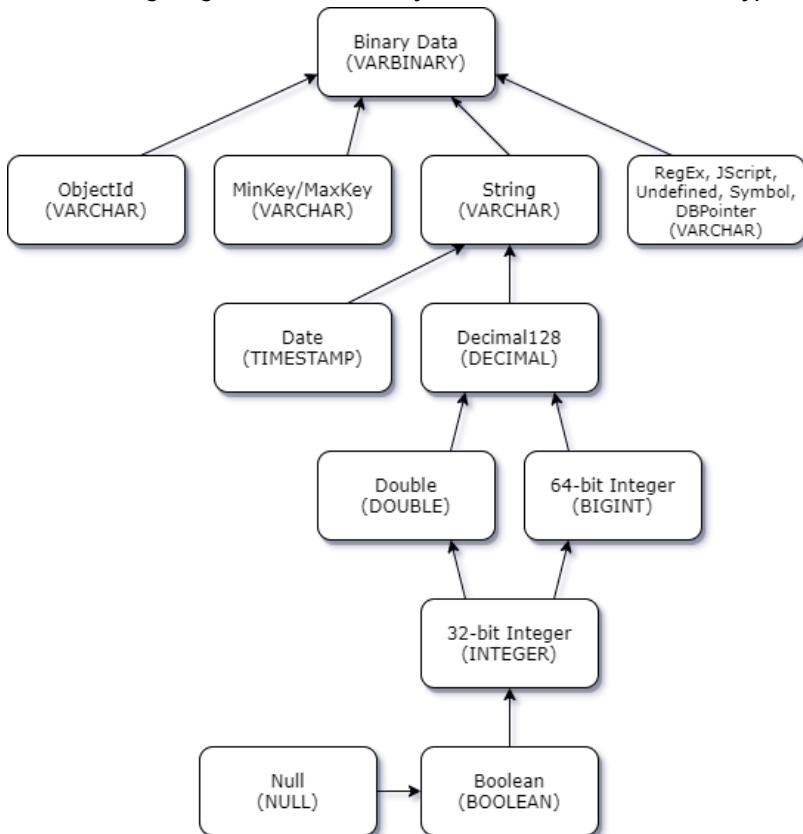
{
  "_id" : "112244",
  "name" : "Benjamin",
  "age" : "32"
}
```

The *age* field is of type 32-bit integer in the first document but string in the second document. Here the JDBC driver will promote the JDBC data type to VARCHAR to handle either data type when encountered.

Table Name	Column Name	Data Type	Key
team	<i>team id</i>	VARCHAR	PK
team	name	VARCHAR	
team	age	VARCHAR	

Scalar-Scalar Conflict Promotion

The following diagram shows the way in which scalar-scalar data type conflicts are resolved.



Scalar-Complex Type Conflict Promotion

Like the scalar-scalar type conflicts, the same field in different documents can have conflicting data types between complex (array and object) and scalar (integer, boolean, etc.). All of these conflicts are resolved (promoted) to VARCHAR for those fields. In this case, array and object data is returned as the JSON representation.

Embedded Array - String Field Conflict Example:

```
{
  "_id": "112233",
  "name": "George Jackson",
  "subscriptions": [
    "Vogue",
    "People",
    "USA Today"
}
```

```

        ]
    }
{
    "_id":"112244",
    "name":"Joan Starr",
    "subscriptions":1
}

```

The above example maps to schema for the `customer2` table:

Table Name	Column Name	Data Type	Key
customer2	<i>customer2 id</i>	VARCHAR	PK
customer2	name	VARCHAR	
customer2	subscription	VARCHAR	

and the `customer1_subscriptions` virtual table:

Table Name	Column Name	Data Type	Key
customer1_subscriptions	<i>customer1 id</i>	VARCHAR	PK/FK
customer1_subscriptions	subscriptions_index_lvl0	BIGINT	PK
customer1_subscriptions	value	VARCHAR	
customer_address	city	VARCHAR	
customer_address	region	VARCHAR	
customer_address	country	VARCHAR	
customer_address	code	VARCHAR	

Object and Array Data Type Handling

So far, we've only described how scalar data types are mapped. Object and Array data types are (currently) mapped to virtual tables. The JDBC driver will create a virtual table to represent either object or array fields in a document. The name of the mapped virtual table will concatenate the original collection's name followed by the field's name separated by an underscore character ("_").

The base table's primary key ("`_id`") takes on a new name in the new virtual table and is provided as a foreign key to the associated base table.

For embedded array type fields, index columns are generated to represent the index into the array at each level of the array.

Embedded Object Field Example

For object fields in a document, a mapping to a virtual table is created by the JDBC driver.

```

{
    "Collection: customer",
    "_id":"112233",
    "name":"George Jackson",
}

```

```
{
    "address": {
        "address1": "123 Avenue Way",
        "address2": "Apt. 5",
        "city": "Hollywood",
        "region": "California",
        "country": "USA",
        "code": "90210"
    }
}
```

The above example maps to schema for customer table:

Table Name	Column Name	Data Type	Key
customer	<i>customer id</i>	VARCHAR	PK
customer	name	VARCHAR	

and the customer_address virtual table:

Table Name	Column Name	Data Type	Key
customer_address	<i>customer id</i>	VARCHAR	PK/FK
customer_address	address1	VARCHAR	
customer_address	address2	VARCHAR	
customer_address	city	VARCHAR	
customer_address	region	VARCHAR	
customer_address	country	VARCHAR	
customer_address	code	VARCHAR	

Embedded Array Field Example

For array fields in a document, a mapping to a virtual table is also created by the JDBC driver.

```
{
    "Collection: customer1",
    "_id": "112233",
    "name": "George Jackson",
    "subscriptions": [
        "Vogue",
        "People",
        "USA Today"
    ]
}
```

The above example maps to schema for customer1 table:

Table Name	Column Name	Data Type	Key
customer1	<i>customer1 id</i>	VARCHAR	PK

Table Name	Column Name	Data Type	Key
customer1	name	VARCHAR	

and the customer1_subscriptions virtual table:

Table Name	Column Name	Data Type	Key
customer1_subscriptions	<i>customer1 id</i>	VARCHAR	PK/FK
customer1_subscriptions	subscriptions_index_lvl0	BIGINT	PK
customer1_subscriptions	value	VARCHAR	
customer_address	city	VARCHAR	
customer_address	region	VARCHAR	
customer_address	country	VARCHAR	
customer_address	code	VARCHAR	

SQL Support and Limitations

The Amazon DocumentDB JDBC driver is a read-only driver that supports a subset of SQL-92 and some common extensions. Refer to the [SQL limitations documentation](#) and [JDBC limitations documentation](#) for more information.

Troubleshooting

If you are having problems using the Amazon DocumentDB JDBC driver, refer to the [Troubleshooting Guide](#).

Amazon DocumentDB Quotas and Limits

This topic describes the resource quotas, limits, and naming constraints for Amazon DocumentDB (with MongoDB compatibility).

For certain management features, Amazon DocumentDB uses operational technology that is shared with Amazon Relational Database Service (Amazon RDS) and Amazon Neptune.

Topics

- [Supported Instance Types \(p. 535\)](#)
- [Supported Regions \(p. 536\)](#)
- [Regional Quotas \(p. 536\)](#)
- [Aggregation Limits \(p. 538\)](#)
- [Cluster Limits \(p. 538\)](#)
- [Instance Limits \(p. 539\)](#)
- [Naming Constraints \(p. 541\)](#)
- [TTL Constraints \(p. 542\)](#)

Supported Instance Types

Amazon DocumentDB supports on-demand instances and the following instance types:

- Memory Optimized:
 - **R6G instance types:** db.r6g.large, db.r6g.2xlarge, db.r6g.4xlarge, db.r6g.8xlarge, db.r6g.12xlarge, db.r6g.16xlarge.
 - **R5 instance types:** db.r5.large, db.r5.2xlarge, db.r5.4xlarge, db.r5.8xlarge, db.r5.12xlarge, db.r5.16xlarge db.r5.24xlarge.
 - **R4 instance types:** db.r4.large, db.r4.2xlarge, db.r4.4xlarge, db.r4.8xlarge, db.r4.16xlarge.
- Burstable Performance:
 - **T4G instance types:** db.t4g.medium.
 - **T3 instance types:** db.t3.medium.

For more information on the supported instance types and their specifications, see [Instance Class Specifications \(p. 317\)](#).

Supported Regions

Amazon DocumentDB is available in the following AWS regions:

Region Name	Region	Availability Zones (compute)
US East (Ohio)	us-east-2	3
US East (N. Virginia)	us-east-1	6
US West (Oregon)	us-west-2	4
South America (São Paulo)	sa-east-1	3
Asia Pacific (Mumbai)	ap-south-1	3
Asia Pacific (Seoul)	ap-northeast-2	4
Asia Pacific (Singapore)	ap-southeast-1	3
Asia Pacific (Sydney)	ap-southeast-2	3
Asia Pacific (Tokyo)	ap-northeast-1	3
Canada (Central)	ca-central-1	3
China (Beijing) Region	cn-north-1	3
China (Ningxia)	cn-northwest-1	3
Europe (Frankfurt)	eu-central-1	3
Europe (Ireland)	eu-west-1	3
Europe (London)	eu-west-2	3
Europe (Milan)	eu-south-1	3
Europe (Paris)	eu-west-3	3
AWS GovCloud (US)	us-gov-west-1	3

Regional Quotas

For certain management features, Amazon DocumentDB uses operational technology that is shared with Amazon Relational Database Service (Amazon RDS) and Amazon Neptune. The following table contains regional limits that are shared among Amazon DocumentDB, Amazon RDS, and Neptune.

The following limits are per AWS account per region.

Resource	AWS Default Limit
Clusters	40
Cluster parameter groups	50

Resource	AWS Default Limit
Event subscriptions	20
Instances	40
Manual cluster snapshots	100
Read replicas per cluster	15
Subnet groups	50
Subnets per subnet group	20
Tags per resource	50
VPC security groups per instance	5

You can use Service Quotas to request an increase for a quota, if the quota is adjustable. Some requests are automatically resolved, while others are submitted to AWS Support. You can track the status of a quota increase request that is submitted to AWS Support. Requests to increase service quotas do not receive priority support. If you have an urgent request, please contact [AWS Support](#). For more information on Service Quotas, see [What Is Service Quotas?](#)

To request a quota increase for Amazon DocumentDB:

1. Open the Service Quotas console at <https://console.aws.amazon.com/servicequotas> and, if necessary, sign in.
2. In the navigation pane, choose **AWS services**.
3. Select Amazon DocumentDB from the list, or type Amazon DocumentDB in the search field.
4. If the quota is adjustable, you can select its radio button or its name, and then choose **Request quota increase** from the top right of the page.
5. For **Change quota value**, enter the new value. The new value must be greater than the current value.
6. Choose **Request**. After the request is resolved, the **Applied quota value** for the quota is set to the new value.
7. To view any pending or recently resolved requests, choose **Dashboard** from the navigation pane. For pending requests, choose the status of the request to open the request receipt. The initial status of a request is Pending. After the status changes to Quota requested, you'll see the case number with AWS Support. Choose the case number to open the ticket for your request.

Aggregation Limits

The following table describes aggregation limits in Amazon DocumentDB.

Resource	Limit
Maximum number of supported stages	500

Cluster Limits

The following table describes Amazon DocumentDB cluster limits.

Resource	Limit
Cluster size (sum of all collections and indexes)	64 TiB
Collection size (sum of all collections can't exceed cluster limit) – does not include the index size	32 TB
Collections per cluster	100,000
Databases per cluster	100,000
Database size (sum of all databases can't exceed cluster limit)	64 TiB
Document nesting depth	100 levels
Document size	16 MB
Index key size	2,048 bytes
Indexes per collection	64
Keys in a compound index	32
Maximum number of writes in a single batch command	100,000
Number of users per cluster	1000

Instance Limits

The following table describes Amazon DocumentDB limits per instance.

Instance Type	Instance Memory (GiB)	Connection Limit	Cursor Limit (per minute)	Open Transaction Limit (per minute)
T3.medium	4	500	30	50
T4G.medium	4	500	30	50
R4.large	15.25	1700	450	N/A
R4.xlarge	30.5	3400	450	N/A
R4.2xlarge	61	6800	450	N/A
R4.4xlarge	122	13600	725	N/A
R4.8xlarge	288	27200	1450	N/A
R4.16xlarge	488	30000	2900	N/A
R5.large	16	1700	450	200
R5.xlarge	32	3500	450	400
R5.2xlarge	64	7100	450	800
R5.4xlarge	128	14200	760	1600
R5.8xlarge	256	28400	1520	3200
R5.12xlarge	383	30000	2280	4800
R5.16xlarge	512	30000	3040	6400
R5.24xlarge	768	30000	4560	9600
R6G.large	16	1700	450	200
R6G.xlarge	32	3500	450	400
R6G.2xlarge	64	7100	450	800
R6G.4xlarge	128	14200	760	1600
R6G.8xlarge	256	28400	1520	3200
R6G.12xlarge	383	30000	2280	4800
R6G.16xlarge	512	30000	3040	6400

You can monitor and alarm on the per instance limits using the following CloudWatch metrics. For more on Amazon DocumentDB CloudWatch metrics, see [Monitoring Amazon DocumentDB with CloudWatch \(p. 414\)](#).

Limit	CloudWatch Metrics
Instance Memory	FreeableMemory
Connections	DatabaseConnectionsMax
Cursors	DatabaseCursorsMax
Transactions	TransactionsOpenMax

Naming Constraints

The following table describes naming constraints in Amazon DocumentDB.

Resource	Default Limit
Cluster identifier	<ul style="list-style-type: none"> Length is [1–63] letters, numbers, or hyphens. First character must be a letter. Cannot end with a hyphen or contain two consecutive hyphens. Must be unique for all clusters (across Amazon RDS, Amazon Neptune, and Amazon DocumentDB) per AWS account, per Region.
Collection name: <col>	Length is [1–57] characters.
Database name: <db>	Length is [1–63] characters.
Fully qualified collection name: <db>. <col>	Length is [3–120] characters.
Fully qualified index name: <db>. <col>. \$<index>	Length is [6–127] characters.
Index name: <col>\$<index>	Length is [3–63] characters.
Instance identifier	<ul style="list-style-type: none"> Length is [1–63] letters, numbers, or hyphens First character must be a letter Cannot end with a hyphen or contain two consecutive hyphens Must be unique for all instances (across Amazon RDS, Amazon Neptune, and Amazon DocumentDB) per AWS account, per Region.
Master password	<ul style="list-style-type: none"> Length is [8–100] printable ASCII characters. Can use any printable ASCII characters except for the following: <ul style="list-style-type: none"> / (forward slash) " (double quotation mark) @ (at symbol)
Master user name	<ul style="list-style-type: none"> Length is [1–63] alphanumeric characters. First character must be a letter. Cannot be a word reserved by the database engine.
Parameter group name	Length is [1–255] alphanumeric characters.

Resource	Default Limit
	<ul style="list-style-type: none">• First character must be a letter.• Cannot end with a hyphen or contain two consecutive hyphens.

TTL Constraints

Deletes from a TTL index are not guaranteed within a specific timeframe and are best effort. Factors like instance resource utilization, document size, and overall throughput can affect the timing of a TTL delete.

Querying

This section explains all aspects of querying with Amazon DocumentDB.

Topics

- [Querying Documents \(p. 543\)](#)
- [Querying Geospatial data with Amazon DocumentDB \(p. 545\)](#)
- [Query Plan \(p. 548\)](#)
- [Explain Results \(p. 550\)](#)

Querying Documents

At times, you might need to look up your online store's inventory so that customers can see and purchase what you're selling. Querying a collection is relatively easy, whether you want all documents in the collection or only those documents that satisfy a particular criterion.

To query for documents, use the `find()` operation. The `find()` command has a single document parameter that defines the criteria to use in choosing the documents to return. The output from `find()` is a document formatted as a single line of text with no line breaks. To format the output document for easier reading, use `find().pretty()`. All the examples in this topic use `.pretty()` to format the output.

The following code samples use the four documents you inserted into the example collection in the preceding two exercises — `insertOne()` and `insertMany()` that are located in the Adding Documents section of [Working with Documents](#).

Topics

- [Retrieving All Documents in a Collection \(p. 543\)](#)
- [Retrieving Documents That Match a Field Value \(p. 544\)](#)
- [Retrieving Documents That Match an Embedded Document \(p. 544\)](#)
- [Retrieving Documents That Match a Field Value in an Embedded Document \(p. 544\)](#)
- [Retrieving Documents That Match an Array \(p. 544\)](#)
- [Retrieving Documents That Match a Value in an Array \(p. 545\)](#)
- [Retrieving Documents Using Operators \(p. 545\)](#)

Retrieving All Documents in a Collection

To retrieve all the documents in your collection, use the `find()` operation with an empty query document.

The following query returns all documents in the example collection.

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

Retrieving Documents That Match a Field Value

To retrieve all documents that match a field and value, use the `find()` operation with a query document that identifies the fields and values to match.

Using the preceding documents, this query returns all documents where the "Item" field equals "Pen".

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

Retrieving Documents That Match an Embedded Document

To find all the documents that match an embedded document, use the `find()` operation with a query document that specifies the embedded document name and all the fields and values for that embedded document.

When matching an embedded document, the document's embedded document must have the same name as in the query. In addition, the fields and values in the embedded document must match the query.

The following query returns only the "Poster Paint" document. This is because the "Pen" has different values for "OnHand" and "MinOnHand", and "Spray Paint" has one more field (`OrderQty`) than the query document.

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

Retrieving Documents That Match a Field Value in an Embedded Document

To find all the documents that match an embedded document, use the `find()` operation with a query document that specifies the embedded document name and all the fields and values for that embedded document.

Given the preceding documents, the following query uses "dot notation" to specify the embedded document and fields of interest. Any document that matches these are returned, regardless of what other fields might be present in the embedded document. The query returns "Poster Paint" and "Spray Paint" because they both match the specified fields and values.

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

Retrieving Documents That Match an Array

To find all documents that match an array, use the `find()` operation with the array name that you are interested in and all the values in that array. The query returns all documents that have an array with that name in which the array values are identical to and in the same order as in the query.

The following query returns only the "Pen" because the "Poster Paint" has an additional color (White), and "Spray Paint" has the colors in a different order.

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

Retrieving Documents That Match a Value in an Array

To find all the documents that have a particular array value, use the `find()` operation with the array name and the value that you're interested in.

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

The preceding operation returns all three documents because each of them has an array named `Colors` and the value "Red" somewhere in the array. If you specify the value "White," the query would only return "Poster Paint."

Retrieving Documents Using Operators

The following query returns all documents where the "Inventory.OnHand" value is less than 50.

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

For a listing of supported query operators, see [Query and Projection Operators \(p. 106\)](#).

Querying Geospatial data with Amazon DocumentDB

This section covers how you can query Geospatial data with Amazon DocumentDB. After you read this section, you will be able to answer how do store, query and index Geospatial data in Amazon DocumentDB.

Topics

- [Overview \(p. 1\)](#)
- [Indexing and Storing Geospatial Data \(p. 545\)](#)
- [Querying Geospatial Data \(p. 547\)](#)
- [Limitations \(p. 548\)](#)

Overview

Common use cases for Geospatial involve proximity analysis from your data. For example, "finding all airports within 50 miles of Seattle", or "find the closest restaurants from a given location". Amazon DocumentDB uses the [GeoJSON specification](#) to represent geospatial data. GeoJSON is an open-source specification for the JSON-formatting of shapes in a coordinate space. GeoJSON coordinates captures both longitude and latitude, representing positions on an earth-like sphere.

Indexing and Storing Geospatial Data

Amazon DocumentDB uses the 'Point' GeoJSON type to store geospatial data. Each GeoJSON document (or subdocument) is generally composed of two fields:

- **type** - the shape being represented, which informs Amazon DocumentDB how to interpret the "coordinates" field. At this moment, Amazon DocumentDB only supports points
- **coordinates** – a latitude and longitude pair represented as an object in an array – [longitude, latitude]

Amazon DocumentDB also uses 2dsphere indexes to index Geospatial data. Amazon DocumentDB supports indexing points. Amazon DocumentDB supports proximity querying with 2dsphere indexing.

Let's consider a scenario where you are building an application for food delivery service. You want to store various restaurant's latitudes and longitude pair in Amazon DocumentDB. To do so, first we recommend that you create an index on the Geospatial field that holds the latitude and longitude pair.

```
use restaurantsdb
db.usarestaurants.createIndex({location:"2dsphere"})
```

The output of this command would look something like this:

```
{
  "createdCollectionAutomatically" : true,
  "numIndexesBefore" : 1,
  "numIndexesAfter" : 2,
  "ok" : 1
}
```

Once you have created an index, you can start inserting data into your Amazon DocumentDB collection.

```
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
    ]
  }
});
```

Querying Geospatial Data

Amazon DocumentDB supports proximity, inclusion and intersection querying of Geospatial data. A good example of a proximity query is finding all points (all airports) that are less than a certain distance and more than a distance from another point (city). A good example of inclusion querying is to find all points (all airports) that located in a specified area/polygon (state of New York). A good example of intersection query is finding a polygon (state) which intersects with a point (city). You can use the following Geospatial operators to gain insights from your data.

- **\$nearSphere** - \$nearSphere is a find operator that supports finding points from nearest to farthest from a GeoJSON point.
- **\$geoNear** - \$geoNear is an aggregation operator that supports calculating the distance in meters from a GeoJSON point.
- **\$minDistance** - \$minDistance is a find operator that is used in conjunction with \$nearSphere or \$geoNear to filter documents that are at least at the specified minimum distance from the center point.
- **\$maxDistance** - \$maxDistance is a find operator that is used in conjunction with \$nearSphere or \$geoNear to filter documents that are at most at the specified maximum distance from the center point.
- **\$geoWithin** - \$geoWithin is a find operator that supports finding documents with geospatial data that exists entirely within a specified shape such as a polygon.
- **\$geoIntersects** - \$geoIntersects is a find operator that supports finding documents whose geospatial data intersects with a specified GeoJSON object.

Note

\$geoNear and \$nearSphere require a 2dsphere index on the GeoJSON field that you use in your proximity query.

Example 1

In this example, you will learn how to find all restaurants (points) sorted by closest distance from an address (point).

To perform such a query, you can use \$geoNear to calculate distance of set of points from another point. You can also add the distanceMultiplier to measure the distance in kilometers.

```
db.usarestaurants.aggregate([
  {
    "$geoNear": {
      "near": {
        "type": "Point",
        "coordinates": [
          -122.3516,
          47.6156
        ]
      },
      "spherical": true,
      "distanceField": "DistanceKilometers",
      "distanceMultiplier": 0.001
    }
  }
])
```

The command above would return restaurants sorted by distance (closest to furthest) from the point specified. The output of this command would look something like this

```
{ "_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 }
```

Example 2

In this example, you will learn how to find all restaurants (points) within 2 kilometers of a specific address (point). To perform such a query, you can use `$nearSphere` within a minimum `$minDistance` and maximum `$maxDistance` from a GeoJSON Point

```
db.usarestaurants.find({  
    "location":{  
        "$nearSphere":{  
            "$geometry":{  
                "$type":"Point",  
                "coordinates": [  
                    -122.3516,  
                    47.6156  
                ]  
            },  
            "$minDistance":1,  
            "$maxDistance":2000  
        }  
    }  
,  
{  
    "name":1  
})
```

The command above would return restaurants at a maximum distance of 2 kilometers from the point specified. The output of this command would look something like this

```
{ "_id" : ObjectId("611f3da985009a81ad38e74b"), "name" : "Noodle House" }
```

Limitations

Amazon DocumentDB does not support querying or indexing of Polygons, LineString, MultiPoint, MultiPolygon, MultiLineString, and GeometryCollection.

Query Plan

How Can I See the executionStats for a Query Plan?

When determining why a query is executing slower than expected, it can be useful to understand what the `executionStats` are for the query plan. The `executionStats` provide the number of documents returned from a particular stage (`nReturned`), the amount of execution time spent at each stage (`executionTimeMillisEstimate`), and the amount of time it takes to generate a query plan

(`planningTimeMillis`). You can determine the most time-intensive stages of your query to help focus your optimization efforts from the output of `executionStats`, as shown in the query examples below. The `executionStats` parameter does not currently support update and delete commands.

Note

Amazon DocumentDB emulates the MongoDB 3.6 API on a purpose-built database engine that utilizes a distributed, fault-tolerant, self-healing storage system. As a result, query plans and the output of `explain()` may differ between Amazon DocumentDB and MongoDB. Customers who want control over their query plan can use the `$hint` operator to enforce selection of a preferred index.

Run the query that you want to improve under the `explain()` command as follows.

```
db.runCommand({explain: {query document}}).  
explain("executionStats").executionStats;
```

The following is an example operation.

```
db.fish.find({}).limit(2).explain("executionStats");
```

Output from this operation looks something like the following.

```
{  
    "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
```

```
}
```

If you are interested in seeing only the `executionStats` from the query above, you can use the following command. For small collections, the Amazon DocumentDB query processor can choose to not use an index if the performance gains are negligible.

```
db.fish.find({}).limit(2).explain("executionStats").executionStats;
```

Query Plan Cache

In order to optimize performance and reduce planning duration, Amazon DocumentDB internally caches query plans. This enables queries with the same shape to be executed directly using a cached plan.

However, this caching may sometimes cause a random delay for the same query; for example, a query that typically takes one second to run may occasionally take ten seconds. This is because over time, the reader instance cached various shapes of the query, thus consuming memory. If you experience this random slowness, there is no action needed you need to do to release the memory--the system will manage the memory usage for you and once the memory reaches certain threshold, it will be automatically released.

Explain Results

If you want to return information on query plans, Amazon DocumentDB supports verbosity mode `queryPlanner`. The `explain` results return the selected query plan chosen by the optimizer in a format similar to the following:

```
{
  "queryPlanner" : {
    "plannerVersion" : <int>,
    "namespace" : <string>,
    "winningPlan" : {
      "stage" : <STAGE1>,
      ...
      "inputStage" : {
        "stage" : <STAGE2>,
        ...
        "inputStage" : {
          ...
        }
      }
    }
  }
}
```

The following sections will define common explain results.

Topics

- [Scan and Filter Stage \(p. 551\)](#)
- [Index Intersection \(p. 551\)](#)
- [Index Union \(p. 552\)](#)
- [Multiple Index Intersection/Union \(p. 552\)](#)
- [Compound Index \(p. 553\)](#)

- [Sort Stage \(p. 553\)](#)
- [Group Stage \(p. 553\)](#)

Scan and Filter Stage

The optimizer may choose one of the following scans:

COLLSCAN

This stage is a sequential collection scan.

```
{  
    "stage" : "COLLSCAN"  
}
```

IXSCAN

This stage scans the index keys. The optimizer may retrieve the document within this stage and this may result in a FETCH stage appended later.

```
db.foo.find({"a": 1})  
{  
    "stage" : "IXSCAN",  
    "direction" : "forward",  
    "indexName" : <idx_name>  
}
```

FETCH

If the optimizer retrieved documents in a stage other than IXSCAN, the result will include a FETCH stage. For example, the IXSCAN query above may result a combination of FETCH and IXSCAN stages:

```
db.foo.find({"a": 1})  
{  
    "stage" : "FETCH",  
    "inputStage" : {  
        "stage" : "IXSCAN",  
        "indexName" : <idx_name>  
    }  
}
```

IXONLYSCAN scans only the index key. Create compound indexes won't avoid FETCH.

Index Intersection

IXAND

Amazon DocumentDB may include an IXAND stage with an inputStages array of IXSCAN if it can utilize index intersection. For example, we may see output like:

```
{  
    "stage" : "FETCH",  
    "inputStage" : {
```

```
    "stage" : "IXAND",
    "inputStages" : [
        {
            "stage" : "IXSCAN",
            "indexName" : "a_1"
        },
        {
            "stage" : "IXSCAN",
            "indexName" : "b_1"
        }
    ]
}
```

Index Union

IXOR

Similar to index intersection, Amazon DocumentDB may include IXOR stage with an inputStages array for the \$or operator.

```
db.foo.find({"$or": [{"a": {"$gt": 2}}, {"b": {"$lt": 2}}]})
```

For the above query, the explain output may look like this:

```
{
    "stage" : "FETCH",
    "inputStage" : {
        "stage" : "IXOR",
        "inputStages" : [
            {
                "stage" : "IXSCAN",
                "indexName" : "a_1"
            },
            {
                "stage" : "IXSCAN",
                "indexName" : "b_1"
            }
        ]
    }
}
```

Multiple Index Intersection/Union

Amazon DocumentDB can combine multiple index intersection or union stages together then fetch the result. For example:

```
{
    "stage" : "FETCH",
    "inputStage" : {
        "stage" : "IXOR",
        "inputStages" : [
            {
                "stage" : "IXSCAN",
                ...
            },
            {

```

```
    "stage" : "IXAND",
    "inputStages" : [
        {
            "stage" : "IXSCAN",
            ...
        },
        {
            "stage" : "IXSCAN",
            ...
        }
    ]
}
```

The usage of index intersection or union stages are not impacted by the index type (sparse, compound, etc).

Compound Index

Amazon DocumentDB compound index usage is not limited in the beginning subsets of indexed fields; it can use index with the suffix part but it may not be very efficient.

For example, the compound index of { a: 1, b: -1 } can support all three queries below:

```
db.orders.find( { a: 1 } )  
db.orders.find( { b: 1 } )  
db.orders.find( { a: 1, b: 1 } )
```

Sort Stage

If there is an index on the requested sort key(s), DocumentDB can use the index to obtain the order. In that case, the result will not include a SORT stage, but rather an IXSCAN stage. If the optimizer favors a plain sort, it will include a stage like this:

```
{
    "stage" : "SORT",
    "sortPattern" : {
        "a" : 1,
        "b" : -1
    }
}
```

Group Stage

Amazon DocumentDB supports two different group strategies:

- SORT_AGGREGATE: On disk sort aggregate.
- HASH_AGGREGATE: In memory hash aggregate.

Troubleshooting Amazon DocumentDB

The following sections provide information about how to troubleshoot problems that you might encounter when using Amazon DocumentDB (with MongoDB compatibility).

Topics

- [Connection Issues \(p. 554\)](#)
- [Index Creation \(p. 557\)](#)
- [Performance and Resource Utilization \(p. 558\)](#)

Connection Issues

Having trouble connecting? Here are some common scenarios and how to resolve them.

Cannot Connect to an Amazon DocumentDB Endpoint

When you try to connect to Amazon DocumentDB, the following is one of the most common error messages that you might receive.

```
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
```

What this error message typically means is that your client (the mongo shell in this example) cannot access the Amazon DocumentDB endpoint. This might be the case for several reasons:

Topics

- [Connecting from Public Endpoints \(p. 554\)](#)
- [Cross Region Connections \(p. 555\)](#)
- [Connecting from Different Amazon VPCs \(p. 555\)](#)
- [Security Group Blocks Inbound Connections \(p. 556\)](#)

Connecting from Public Endpoints

You are trying to connect to an Amazon DocumentDB cluster directly from your laptop or local development machine.

Trying to connect to an Amazon DocumentDB cluster directly from a public endpoint, such as your laptop or local development machine, will fail. Amazon DocumentDB is virtual private cloud (VPC)-only and does not currently support public endpoints. Thus, you can't connect directly to your Amazon DocumentDB cluster from your laptop or local development environment outside of your VPC.

To connect to an Amazon DocumentDB cluster from outside an Amazon VPC, you can use an SSH tunnel. For more information, see [Connecting to an Amazon DocumentDB Cluster from Outside an Amazon VPC \(p. 499\)](#). Additionally, if your development environment is in a different Amazon VPC, you can also use VPC Peering and connect to your Amazon DocumentDB cluster from another Amazon VPC in the same region or a different region.

Cross Region Connections

You are trying to connect to an Amazon DocumentDB cluster in another region.

If you try to connect to an Amazon DocumentDB cluster from an Amazon EC2 instance in a Region other than the cluster's Region—for example, trying to connect to a cluster in US East (N. Virginia) Region (us-east-1) from US West (Oregon) Region (us-west-2)—the connection will fail.

To verify the Region of your Amazon DocumentDB cluster, run the following command. The Region is in the endpoint.

```
aws docdb describe-db-clusters \
--db-cluster-identifier sample-cluster \
--query 'DBClusters[*].Endpoint'
```

Output from this operation looks something like the following.

```
[ "sample-cluster.node.us-east-1.docdb.amazonaws.com" ]
```

To verify the Region of your EC2 instance, run the following command.

```
aws ec2 describe-instances \
--query 'Reservations[*].Instances[*].Placement.AvailabilityZone'
```

Output from this operation looks something like the following.

```
[ [ "us-east-1a" ] ]
```

Connecting from Different Amazon VPCs

You are trying to connect to an Amazon DocumentDB cluster from a VPC that is different than the Amazon VPC your cluster is deployed to.

If both your Amazon DocumentDB cluster and Amazon EC2 instance are in the same AWS Region, but not in the same Amazon VPC, you cannot connect directly to your Amazon DocumentDB cluster unless VPC Peering is enabled between the two Amazon VPCs.

To verify the Amazon VPC of your Amazon DocumentDB instance, run the following command.

```
aws docdb describe-db-instances \
```

```
--db-instance-identifier sample-instance \
--query 'DBInstances[*].DBSubnetGroup.VpcId'
```

To verify the Amazon VPC of your Amazon EC2 instance, run the following command.

```
aws ec2 describe-instances \
--query 'Reservations[*].Instances[*].VpcId'
```

Security Group Blocks Inbound Connections

You are trying to connect to an Amazon DocumentDB cluster, and the cluster's security group does not allow inbound connections on the cluster's port (default port: 27017).

Suppose that your Amazon DocumentDB cluster and Amazon EC2 instance are both in the same Region and Amazon VPC and use the same Amazon VPC security group. If you can't connect to your Amazon DocumentDB cluster, the likely cause is that your security group (that is, firewall) for your cluster doesn't allow inbound connections on the port you chose for your Amazon DocumentDB cluster (default port is 27017).

To verify the port for your Amazon DocumentDB cluster, run the following command.

```
aws docdb describe-db-clusters \
--db-cluster-identifier sample-cluster \
--query 'DBClusters[*].[DBClusterIdentifier,Port]'
```

To get your Amazon DocumentDB security group for your cluster, run the following command.

```
aws docdb describe-db-clusters \
--db-cluster-identifier sample-cluster \
--query 'DBClusters[*].[VpcSecurityGroups[*],VpcSecurityGroupId]'
```

To check the inbound rules for your security group, see the following topics in the Amazon EC2 documentation:

- [Authorizing Inbound Traffic for Your Linux Instances](#)
- [Authorizing Inbound Traffic for Your Windows Instances](#)

Testing a Connection to an Amazon DocumentDB Instance

You can test your connection to a cluster using common Linux or Windows tools.

From a Linux or Unix terminal, test the connection by entering the following (replace `cluster-endpoint` with the endpoint, and replace `port` with the port of your instance):

```
nc -zv cluster-endpoint port
```

The following is an example of a sample operation and the return value:

```
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!
```

Connecting to an Invalid Endpoint

When connecting to an Amazon DocumentDB cluster and you use a cluster endpoint that is not valid, an error similar to the following appears.

```
mongo --ssl \
--host sample-cluster.node.us-east-1.docdb.amazonaws.com:27017 \
--sslCAFile rds-combined-ca-bundle.pem \
--username <user-name> \
--password <password>
```

The output looks like this:

```
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
```

To get the valid endpoint for a cluster, run the following command:

```
aws docdb describe-db-clusters \
--db-cluster-identifier sample-cluster \
--query 'DBClusters[*].[Endpoint,Port]'
```

To get the valid endpoint for an instance, run the following command:

```
aws docdb describe-db-instances \
--db-instance-identifier sample-instance \
--query 'DBInstances[*].[Endpoint.Address,Endpoint.Port]'
```

For more information, see [Understanding Amazon DocumentDB Endpoints \(p. 378\)](#).

Index Creation

The following topics address what to do if your index or background index build fails.

Topics

- [Index Build Fails \(p. 557\)](#)
- [Background Index Build Latency Issues and Fails \(p. 558\)](#)

Index Build Fails

Amazon DocumentDB utilizes local storage on an instance as part of the index creation process. You can monitor this disk usage using the **FreeLocalStorage** CloudWatch metric (CloudWatch -> Metrics -> DocDB -> Instance Metrics). When an index build consumes all of the local disk and fails, you will receive an error. When migrating data to Amazon DocumentDB, we encourage you to create indexes first and then insert the data. For more information on migration strategies and creating indexes, see

[Migrating to Amazon DocumentDB \(p. 121\)](#) in the Amazon DocumentDB documentation and the blog: [Migrate from MongoDB to Amazon DocumentDB using the offline method](#).

When creating indexes on an existing cluster, if the index build is taking longer than expected or is failing, we recommend that you scale up the instance to create the index then, after the index is created, scale back down. Amazon DocumentDB enables you to quickly scale instance sizes in minutes using the AWS Management Console or the AWS CLI. For more information, see [Managing Instance Classes \(p. 313\)](#). With per-second instance pricing, you only pay for the resource you use up to the second.

Background Index Build Latency Issues and Fails

Background index builds in Amazon DocumentDB do not start until all queries on the primary instance that started before the index build was initiated complete executing. If there is a long running query, background index builds will block until the query finishes and thus can take longer than expected to complete. This is true even if collections are empty.

Foreground index builds do not exhibit the same blocking behavior. Instead, foreground index builds take an exclusive lock on the collection until the index build is completed. Thus, to create indexes on empty collection and to avoid blocking on any long running queries, we suggest using foreground index builds.

Note

Amazon DocumentDB allows only one background index build to occur on a collection at any given time. If DDL (Data Definition Language) operations such as `createIndex()` or `dropIndex()` occur on the same collection during a background index build, the background index build fails.

Performance and Resource Utilization

This section provides questions and solutions for common diagnostics issues in Amazon DocumentDB deployments. The examples provided use the *mongo shell* and are scoped to an individual instance. To find an instance endpoint, see [Understanding Amazon DocumentDB Endpoints \(p. 378\)](#).

Topics

- [How Do I Find and Terminate Long Running or Blocked Queries? \(p. 558\)](#)
- [How Can I See a Query Plan and Optimize a Query? \(p. 560\)](#)
- [How Do I List All Running Operations on an Instance? \(p. 561\)](#)
- [How Do I Know When a Query Is Making Progress? \(p. 562\)](#)
- [How Do I Determine Why a System Suddenly Runs Slowly? \(p. 564\)](#)
- [How Do I Determine the Cause of High CPU Utilization on One or More Cluster Instances? \(p. 565\)](#)
- [How Do I Determine the Open Cursors on an Instance? \(p. 566\)](#)
- [How do I Determine the Current Amazon DocumentDB Engine Version? \(p. 566\)](#)
- [How Do I Identify Unused Indexes? \(p. 567\)](#)
- [How Do I Identify Missing Indexes? \(p. 567\)](#)
- [Summary of Useful Queries \(p. 568\)](#)

How Do I Find and Terminate Long Running or Blocked Queries?

User queries can run slowly because of a suboptimal query plan or can be blocked due to resource contention.

To find long running queries that slow down due to a suboptimal query plan, or queries that are blocked due to resource contention, use the `currentOp` command. You can filter the command to help narrow down the list of relevant queries to terminate. You must have opid associated with the long running query to be able to terminate the query.

The following query uses the `currentOp` command to list all queries that are either blocked or running for more than 10 seconds.

```
db.adminCommand({
  aggregate: 1,
  pipeline: [
    {$currentOp: {}},
    {$match:
      {$or: [
        {secs_running: {$gt: 10}},
        {WaitState: {$exists: true}}]}},
      {$project: {_id:0, opid: 1, secs_running: 1}}],
    cursor: {}
});
```

Next, you can narrow down the query to find the opid of a query running for more than 10 seconds and terminate it.

To find and terminate a query running for more than 10 seconds

1. Find the opid of the query.

```
db.adminCommand({
  aggregate: 1,
  pipeline: [
    {$currentOp: {}},
    {$match:
      {$or:
        [{secs_running: {$gt: 10}},
         {WaitState: {$exists: true}}]}},
      cursor: {}
});
```

Output from this operation looks something like the following (JSON format).

```
{
  "waitedMS" : NumberLong(0),
  "cursor" : {
    "firstBatch" : [
      {
        "opid" : 24646,
        "secs_running" : 12
      }
    ],
    "id" : NumberLong(0),
    "ns" : "admin.$cmd"
  },
  "ok" : 1
}
```

2. Terminate the query using the `killOp` operation.

```
db.adminCommand({killOp: 1, op: 24646});
```

How Can I See a Query Plan and Optimize a Query?

If a query runs slow, it could be because the query execution requires a full scan of the collection to choose the relevant documents. Sometimes creating appropriate indexes enables the query to run faster. To detect this scenario and decide the fields on which to create the indexes, use the `explain` command.

Note

Amazon DocumentDB emulates the MongoDB 3.6 API on a purpose-built database engine that utilizes a distributed, fault-tolerant, self-healing storage system. As a result, query plans and the output of `explain()` may differ between Amazon DocumentDB and MongoDB. Customers who want control over their query plan can use the `$hint` operator to enforce selection of a preferred index.

Run the query that you want to improve under the `explain` command as follows.

```
db.runCommand({explain: {<query document>}})
```

The following is an example operation.

```
db.runCommand({explain:{  
    aggregate: "sample-document",  
    pipeline: [{$match: {x: {$eq: 1}}}],  
    cursor: {batchSize: 1}  
}});
```

Output from this operation looks something like the following (JSON format).

```
{  
  "queryPlanner" : {  
    "plannerVersion" : 1,  
    "namespace" : "db.test",  
    "winningPlan" : {  
      "stage" : "COLLSCAN"  
    }  
  },  
  "serverInfo" : {  
    "host" : "...",  
    "port" : ...,  
    "version" : "..."  
  },  
  "ok" : 1  
}
```

The preceding output indicates that the `$match` stage requires scanning the whole collection and checking if the field "x" in each document is equal to 1. If there are many documents in the collection, the collection scan (and therefore the overall query performance) is very slow. Thus the presence of the "COLLSCAN" in the output of the `explain` command indicates that the query performance can be improved by creating appropriate indexes.

In this example, the query checks whether the field "x" equals 1 in all documents. So creating an index on field "x" enables the query to avoid the complete collection scan and use the index to return the relevant documents sooner.

After creating an index on field "x", the `explain` output is as follows.

```
{  
  "queryPlanner" : {  
    "plannerVersion" : 1,  
    "namespace" : "db.test",  
  }
```

```

        "winningPlan" : {
            "stage" : "IXSCAN",
            "indexName" : "x_1",
            "direction" : "forward"
        }
    },
    "serverInfo" : {
        "host" : "...",
        "port" : ...,
        "version" : "..."
    },
    "ok" : 1
}

```

Creating an index on field "x" enables the \$match stage to use an index scan to reduce the number of documents on which the predicate "x = 1" must be evaluated.

For small collections, the Amazon DocumentDB query processor can choose not to use an index if the performance gains are negligible.

How Do I List All Running Operations on an Instance?

As a user or master user, you often want to list all the current operations running on an instance for diagnostics and troubleshooting purposes. (For information about managing users, see [Managing Amazon DocumentDB Users \(p. 166\)](#).)

With the mongo shell, you can use the following query to list all the running operations on an Amazon DocumentDB instance.

```
db.adminCommand({currentOp: 1, $all: 1});
```

The query returns the complete list of all user queries and internal system tasks currently operating on the instance.

Output from this operation looks something like the following (JSON format).

```
{
    "inprog" : [
        {
            "desc" : "INTERNAL"
        },
        {
            "desc" : "TTLMonitor",
            "active" : false
        },
        {
            "desc" : "GARBAGE_COLLECTION"
        },
        {
            "client" : ....,
            "desc" : "Conn",
            "active" : true,
            "killPending" : false,
            "opid" : 195,
            "ns" : "admin.$cmd",
            "command" : {
                "currentOp" : 1,
                "$all" : 1
            },
            "op" : "command",
            "$db" : "admin",
            "secs_running" : 0,
            "secs_since_last_io" : 0,
            "millis_running" : 0,
            "millis_since_last_io" : 0
        }
    ]
}
```

```

    "microsecs_running" : NumberLong(68),
    "clientMetaData" : {
        "application" : {
            "name" : "MongoDB Shell"
        },
        "driver" : {
            ...
        },
        "os" : {
            ...
        }
    },
    "ok" : 1
}

```

The following are valid values for the "desc" field:

- **INTERNAL** — Internal system tasks like the cursor cleanup or stale user cleanup tasks.
- **TTLMonitor** — The Time to Live (TTL) monitor thread. Its running status is reflected in the "active" field.
- **GARBAGE_COLLECTION** — The internal garbage collector thread. There can be a maximum of three garbage collector threads running concurrently in the system.
- **CONN** — The user query.
- **CURSOR** — The operation is an idle cursor waiting on the user to call the "getMore" command to get the next batch of results. In this state, the cursor is consuming memory, but is not consuming any compute.

The preceding output also lists all user queries running in the system. Each user query runs in the context of a database and collection, and the union of these two is called a *namespace*. The namespace of each user query is available in the "ns" field.

Sometimes you need to list all user queries that are running in a particular namespace. So the previous output must be filtered on the "ns" field. The following is an example query to achieve the output to filter. The query lists all user queries that are currently running in the database "db" and collection "test" (that is, the "db.test" namespace).

```

db.adminCommand({aggregate: 1,
  pipeline: [{$currentOp: {allUsers: true, idleConnections: true}},
    {$match: {ns: {$eq: "db.test"}}}],
  cursor: {}});

```

As the master user of the system, you can see queries of all users and also all internal system tasks. All other users can see only their respective queries.

If the total number of queries and internal system tasks exceeds the default batch cursor size, the mongo shell automatically generates an iterator object 'it' to view the rest of the results. Keep executing the 'it' command until all results have been exhausted.

How Do I Know When a Query Is Making Progress?

User queries can run slowly due to a suboptimal query plan, or they can be blocked due to resource contention. Debugging such queries is a multi-step process that can require executing the same step multiple times.

The first step of debugging is to list all queries that are long running or blocked. The following query lists all user queries that have been running for more than 10 seconds or that are waiting for resources.

```
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: {}
});
```

Repeat the preceding query periodically to determine whether the list of queries changes and to identify the long running or blocked queries.

If the output document for the query of interest has a `WaitState` field, it indicates that resource contention is why the query is running slow or is blocked. The resource contention could either be due to I/O, internal system tasks, or other user queries.

Output from this operation looks something like the following (JSON format).

```
{
    "waitedMS" : NumberLong(0),
    "cursor" : {
        "firstBatch" : [
            {
                "opid" : 201,
                "command" : {
                    "aggregate" : ...
                },
                "secs_running" : 208,
                "WaitState" : "IO"
            }
        ],
        "id" : NumberLong(0),
        "ns" : "admin.$cmd"
    },
    "ok" : 1
}
```

I/O can be a bottleneck if many queries across different collections are running concurrently on the same instance, or if the instance size is too small for the dataset that the query is running on. If the queries are read-only queries, you can mitigate the former situation by separating the queries for each collection across separate replicas. For concurrent updates across different collections, or when the instance size is too small for the dataset, you can mitigate by scaling up the instance.

If the resource contention is due to other user queries, the `"blockedOn"` field in the output document will have the `"opid"` of the query that is affecting this query. Using the `"opid"` follows the chain of `"WaitState"` and `"blockedOn"` fields of all the queries to find the query at the head of the chain.

If the task at the head of the chain is an internal task, the only mitigation in this case would be to terminate the query and rerun it later.

The following is sample output in which the find query is blocked on a collection lock that is owned by another task.

```
{
    "inprog" : [
        {
            "client" : "...",
            "lock" : {
                "opid" : 1000000000000000000,
                "secs_running" : 1000000000000000000,
                "WaitState" : "LOCK"
            }
        }
    ]
}
```

```

    "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
}

```

If the "WaitState" has values "Latch", "SystemLock", "BufferLock", "BackgroundActivity", or "Other", the source of resource contention is internal system tasks. If the situation continues for a long time, the only mitigation would be to terminate the query and rerun it later.

How Do I Determine Why a System Suddenly Runs Slowly?

The following are some common reasons for a system slowing down:

- Excessive resource contention between concurrent queries
- The number of active concurrent queries increasing over time
- Internal system tasks such as "GARBAGE_COLLECTION"

To monitor the system usage over time, run the following "currentOp" query periodically and output the results to an external store. The query counts the number of queries and operations in each

namespace in the system. You can then analyze the system usage results to understand the load on the system and take appropriate action.

```
db.adminCommand({aggregate: 1,
                  pipeline: [{$currentOp: {allUsers: true, idleConnections: true}},
                             {$group: {_id: {desc: "$desc", ns: "$ns", WaitState: "$WaitState"}, count: {$sum: 1}}}],
                  cursor: {}});
```

This query returns an aggregate of all queries running in each namespace, all the internal system tasks, and the unique number of wait states (if any) per namespace.

Output from this operation looks something like the following (JSON format).

```
{
  "waitedMS" : NumberLong(0),
  "cursor" : {
    "firstBatch" : [
      {
        "_id" : {
          "desc" : "Conn",
          "ns" : "db.test",
          "WaitState" : "CollectionLock"
        },
        "count" : 2
      },
      {
        "_id" : {
          "desc" : "Conn",
          "ns" : "admin.$cmd"
        },
        "count" : 1
      },
      {
        "_id" : {
          "desc" : "TTLMonitor"
        },
        "count" : 1
      }
    ],
    "id" : NumberLong(0),
    "ns" : "admin.$cmd"
  },
  "ok" : 1
}
```

In the preceding output, two user queries in namespace "db.test" are blocked on collection lock: one query in the namespace "admin.\$cmd", and one internal "TTLMonitor" task.

If the output indicates many queries with blocking wait states, see [How Do I Find and Terminate Long Running or Blocked Queries? \(p. 558\)](#)

How Do I Determine the Cause of High CPU Utilization on One or More Cluster Instances?

The following sections might help you identify the cause of high instance CPU utilization. Your results can vary depending on the workload.

- To determine why an instance is suddenly running slowly, see [How Do I Determine Why a System Suddenly Runs Slowly? \(p. 564\)](#)
- To identify and terminate long running queries on a particular instance, see [How Do I Find and Terminate Long Running or Blocked Queries? \(p. 558\)](#)
- To understand whether a query is progressing, see [How Do I Know When a Query Is Making Progress? \(p. 562\)](#)
- To determine why a query takes a long time to run, see [How Can I See a Query Plan and Optimize a Query? \(p. 560\)](#)
- To track long-running queries over time, see [Profiling Amazon DocumentDB Operations \(p. 427\)](#).

Depending on the reason for your high instance CPU utilization, doing one or more of the following can help.

- If the primary instance exhibits high CPU utilization, but the replica instances don't, consider distributing read traffic across replicas via client read preference settings (for example, `secondaryPreferred`). For more information, see [Connecting to Amazon DocumentDB as a Replica Set \(p. 496\)](#).

Using replicas for reads can make better use of the cluster's resources by allowing the primary instance to process more write traffic. Reads from replicas are eventually consistent.

- If the high CPU utilization is a result of your write workload, changing the size of the cluster's instances to a larger instance type increases the number of CPU cores available to service the workload. For more information, see [Instances \(p. 2\)](#) and [Instance Class Specifications \(p. 317\)](#).
- If all cluster instances exhibit high CPU utilization, and the workload is using replicas for reads, adding more replicas to the cluster increases the resources available for read traffic. For more information, see [Adding an Amazon DocumentDB Instance to a Cluster \(p. 319\)](#).

How Do I Determine the Open Cursors on an Instance?

When connected to a Amazon DocumentDB instance, you can use the command `db.runCommand("listCursors")` to list the open cursors on that instance. There is a limit of up to 4,560 active cursors open at any given time on a given Amazon DocumentDB instance, depending on the instance type. It is generally advised to close cursors that are no longer in use because cursors utilize resources on an instance and have an upper limit. See [Amazon DocumentDB Quotas and Limits \(p. 535\)](#) for specific limits.

```
db.runCommand("listCursors")
```

How do I Determine the Current Amazon DocumentDB Engine Version?

To determine your current Amazon DocumentDB engine version, run the following command.

```
db.runCommand({getEngineVersion: 1})
```

Output from this operation looks something like the following (JSON format).

```
{ "engineVersion" : "2.x.x", "ok" : 1 }
```

Note

The engine version for Amazon DocumentDB 3.6 is 1.x.x and the engine version for Amazon DocumentDB 4.0 is 2.x.x.

How Do I Identify Unused Indexes?

It is a best practice to regularly identify and remove unused indexes in order to improve performance and reduce cost, as it eliminates unnecessary compute, storage, and IOs used to maintain the indexes. To identify the indexes for a given collection, run the following command:

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

To identify whether or not an index has been utilized, run the following command. Output from the command describes the following:

```
db.collection.aggregate([{$indexStats:{}}]).pretty()
```

- **ops** —The number of operations that used the index. If your workload has been running for a sufficiently long time and you are confident that your workload is in a steady state, an ops value of zero would indicate that the index is not used at all.
- **since** —The time since Amazon DocumentDB started collecting stats on index usage, which is typically the value since the last database restart or maintenance action.

To determine the overall index size for a collection, run the following command:

```
db.collection.stats()
```

To drop an unused index, run the following command:

```
db.collection.dropIndex("indexName")
```

How Do I Identify Missing Indexes?

You can use the [Amazon DocumentDB profiler to log slow queries](#). A query that appears repeatedly in the slow query log may indicate that an additional index is required to improve that query's performance.

You can identify opportunities for helpful indexes by looking for long running queries that have one or more stages that perform at least one COLLSCAN stage, meaning that they query stage has to read every document in the collection in order to provide a response to the query.

The following example shows a query on a collection of taxi rides that ran on a large collection.

```
db.rides.count({"fare.totalAmount":{$gt:10.0}})
```

In order to execute this example, the query had to perform a collection scan (i.e. read every single document in the collection) since there is no index on the fare.totalAmount field. Output from the Amazon DocumentDB profiler for this query looks something like the following:

```
{  
  ...  
  "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"  
}
```

To speed up the query in this example, you want to create an index on `fare.totalAmount`, as shown below.

```
db.rides.createIndex( {"fare.totalAmount": 1}, {background: true} )
```

Note

Indexes created in the foreground (meaning if the `{background: true}` option was not supplied when creating the index) take an exclusive write lock, which prevents applications from writing data to the collection until the index build completes. Be aware of this potential impact when creating indexes on production clusters. When creating indexes, we recommend setting `{background: true}`.

In general, you want to create indexes on fields that have high cardinality (for example, a large number of unique values). Creating an index on a field with low cardinality can result in a large index that is not used. The Amazon DocumentDB query optimizer considers the overall size of the collection and selectivity of the indexes when creating a query plan. There are times where you will see the query processor select a COLLSCAN even when an index is present. This happens when the query processor estimates that utilizing the index will not yield a performance advantage over scanning the entire collection. If you want to force the query processor to utilize a particular index, you can use the `hint()` operator as shown below.

```
db.collection.find().hint("indexName")
```

Summary of Useful Queries

The following queries can be useful for monitoring performance and resource utilization in Amazon DocumentDB.

- Use the following query to list all activity.

```
db.adminCommand({currentOp: 1, $all: 1});
```

- The following code lists all long running or blocked queries.

```
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,  
            client: 1,  
            appName: 1,  
            user: 1}],  
    cursor: 1});
```

```
    command: 1}]],  
    cursor: {}  
});
```

- The following code terminates a query.

```
db.adminCommand({killOp: 1, op: <opid of running or blocked query>});
```

- Use the following code to get an aggregated view of the system state.

```
db.adminCommand({aggregate: 1,  
    pipeline: [{$currentOp: {allUsers: true, idleConnections: true}},  
        {$group: {_id: {desc: "$desc", ns: "$ns", WaitState:  
            "$WaitState"}, count: {$sum: 1}}}],  
    cursor: {}  
});
```

Amazon DocumentDB Cluster, Instance, and Resource Management API Reference

This section describes the cluster, instance, and resource management operations for Amazon DocumentDB (with MongoDB compatibility) that are accessible via HTTP, the AWS Command Line Interface (AWS CLI), or the AWS SDK. You can use these APIs to create, delete, and modify clusters and instances.

Important

These APIs are used only for managing clusters, instances, and related resources. For information about how to connect to a running Amazon DocumentDB cluster, see [Get Started Guide \(p. 31\)](#).

Topics

- [Actions \(p. 570\)](#)
- [Data Types \(p. 703\)](#)
- [Common Errors \(p. 755\)](#)
- [Common Parameters \(p. 757\)](#)

Actions

The following actions are supported:

- [AddSourceIdentifierToSubscription \(p. 572\)](#)
- [AddTagsToResource \(p. 574\)](#)
- [ApplyPendingMaintenanceAction \(p. 576\)](#)
- [CopyDBClusterParameterGroup \(p. 578\)](#)
- [CopyDBClusterSnapshot \(p. 580\)](#)
- [CreateDBCluster \(p. 584\)](#)
- [CreateDBClusterParameterGroup \(p. 590\)](#)
- [CreateDBClusterSnapshot \(p. 592\)](#)
- [CreateDBInstance \(p. 594\)](#)
- [CreateDBSubnetGroup \(p. 599\)](#)
- [CreateEventSubscription \(p. 601\)](#)
- [CreateGlobalCluster \(p. 604\)](#)
- [DeleteDBCluster \(p. 607\)](#)
- [DeleteDBClusterParameterGroup \(p. 609\)](#)
- [DeleteDBClusterSnapshot \(p. 611\)](#)
- [DeleteDBInstance \(p. 613\)](#)
- [DeleteDBSubnetGroup \(p. 615\)](#)
- [DeleteEventSubscription \(p. 617\)](#)

- [DeleteGlobalCluster \(p. 619\)](#)
- [DescribeCertificates \(p. 621\)](#)
- [DescribeDBClusterParameterGroups \(p. 623\)](#)
- [DescribeDBClusterParameters \(p. 625\)](#)
- [DescribeDBClusters \(p. 627\)](#)
- [DescribeDBClusterSnapshotAttributes \(p. 629\)](#)
- [DescribeDBClusterSnapshots \(p. 631\)](#)
- [DescribeDBEngineVersions \(p. 634\)](#)
- [DescribeDBInstances \(p. 637\)](#)
- [DescribeDBSubnetGroups \(p. 639\)](#)
- [DescribeEngineDefaultClusterParameters \(p. 641\)](#)
- [DescribeEventCategories \(p. 643\)](#)
- [DescribeEvents \(p. 645\)](#)
- [DescribeEventSubscriptions \(p. 648\)](#)
- [DescribeGlobalClusters \(p. 650\)](#)
- [DescribeOrderableDBInstanceStateOptions \(p. 652\)](#)
- [DescribePendingMaintenanceActions \(p. 654\)](#)
- [FailoverDBCluster \(p. 656\)](#)
- [ListTagsForResource \(p. 658\)](#)
- [ModifyDBCluster \(p. 660\)](#)
- [ModifyDBClusterParameterGroup \(p. 665\)](#)
- [ModifyDBClusterSnapshotAttribute \(p. 667\)](#)
- [ModifyDBInstance \(p. 669\)](#)
- [ModifyDBSubnetGroup \(p. 674\)](#)
- [ModifyEventSubscription \(p. 676\)](#)
- [ModifyGlobalCluster \(p. 678\)](#)
- [RebootDBInstance \(p. 680\)](#)
- [RemoveFromGlobalCluster \(p. 682\)](#)
- [RemoveSourceIdentifierFromSubscription \(p. 684\)](#)
- [RemoveTagsFromResource \(p. 686\)](#)
- [ResetDBClusterParameterGroup \(p. 688\)](#)
- [RestoreDBClusterFromSnapshot \(p. 690\)](#)
- [RestoreDBClusterToPointInTime \(p. 695\)](#)
- [StartDBCluster \(p. 700\)](#)
- [StopDBCluster \(p. 702\)](#)

AddSourceIdentifierToSubscription

Adds a source identifier to an existing event notification subscription.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

SOURCEIDENTIFIER

The identifier of the event source to be added:

- If the source type is an instance, a DBInstanceIdentifier must be provided.
- If the source type is a security group, a DBSecurityGroupName must be provided.
- If the source type is a parameter group, a DBParameterGroupName must be provided.
- If the source type is a snapshot, a DBSnapshotIdentifier must be provided.

Type: String

Required: Yes

SubscriptionName

The name of the Amazon DocumentDB event notification subscription that you want to add a source identifier to.

Type: String

Required: Yes

Response Elements

The following element is returned by the service.

EventSubscription

Detailed information about an event to which you have subscribed.

Type: [EventSubscription \(p. 735\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

SourceNotFound

The requested source could not be found.

HTTP Status Code: 404

SubscriptionNotFound

The subscription name does not exist.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

AddTagsToResource

Adds metadata tags to an Amazon DocumentDB resource. You can use these tags with cost allocation reporting to track costs that are associated with Amazon DocumentDB resources or in a Condition statement in an AWS Identity and Access Management (IAM) policy for Amazon DocumentDB.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

ResourceName

The Amazon DocumentDB resource that the tags are added to. This value is an Amazon Resource Name .

Type: String

Required: Yes

Tags.Tag.N

The tags to be assigned to the Amazon DocumentDB resource.

Type: Array of [Tag \(p. 753\)](#) objects

Required: Yes

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBClusterNotFoundFault

`DBClusterIdentifier` doesn't refer to an existing cluster.

HTTP Status Code: 404

DBInstanceNotFound

`DBInstanceIdentifier` doesn't refer to an existing instance.

HTTP Status Code: 404

DBSnapshotNotFound

`DBSnapshotIdentifier` doesn't refer to an existing snapshot.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)

- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ApplyPendingMaintenanceAction

Applies a pending maintenance action to a resource (for example, to an Amazon DocumentDB instance).

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

ApplyAction

The pending maintenance action to apply to this resource.

Valid values: system-update, db-upgrade

Type: String

Required: Yes

OptInType

A value that specifies the type of opt-in request or undoes an opt-in request. An opt-in request of type immediate can't be undone.

Valid values:

- immediate - Apply the maintenance action immediately.
- next-maintenance - Apply the maintenance action during the next maintenance window for the resource.
- undo-opt-in - Cancel any existing next-maintenance opt-in requests.

Type: String

Required: Yes

ResourceIdentifier

The Amazon Resource Name (ARN) of the resource that the pending maintenance action applies to.

Type: String

Required: Yes

Response Elements

The following element is returned by the service.

ResourcePendingMaintenanceActions

Represents the output of [ApplyPendingMaintenanceAction \(p. 576\)](#).

Type: [ResourcePendingMaintenanceActions \(p. 751\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

InvalidDBClusterStateFault

The cluster isn't in a valid state.

HTTP Status Code: 400

InvalidDBInstanceState

The specified instance isn't in the *available* state.

HTTP Status Code: 400

ResourceNotFoundFault

The specified resource ID was not found.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

CopyDBClusterParameterGroup

Copies the specified cluster parameter group.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

SourceDBClusterParameterGroupIdentifier

The identifier or Amazon Resource Name (ARN) for the source cluster parameter group.

Constraints:

- Must specify a valid cluster parameter group.
- If the source cluster parameter group is in the same AWS Region as the copy, specify a valid parameter group identifier; for example, my-db-cluster-param-group, or a valid ARN.
- If the source parameter group is in a different AWS Region than the copy, specify a valid cluster parameter group ARN; for example, arn:aws:rds:us-east-1:123456789012:sample-cluster:sample-parameter-group.

Type: String

Required: Yes

TargetDBClusterParameterGroupDescription

A description for the copied cluster parameter group.

Type: String

Required: Yes

TargetDBClusterParameterGroupIdentifier

The identifier for the copied cluster parameter group.

Constraints:

- Cannot be null, empty, or blank.
- Must contain from 1 to 255 letters, numbers, or hyphens.
- The first character must be a letter.
- Cannot end with a hyphen or contain two consecutive hyphens.

Example: my-cluster-param-group1

Type: String

Required: Yes

Tags.Tag.N

The tags that are to be assigned to the parameter group.

Type: Array of [Tag \(p. 753\)](#) objects

Required: No

Response Elements

The following element is returned by the service.

DBClusterParameterGroup

Detailed information about a cluster parameter group.

Type: [DBClusterParameterGroup \(p. 714\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBParameterGroupAlreadyExists

A parameter group with the same name already exists.

HTTP Status Code: 400

DBParameterGroupNotFound

DBParameterGroupName doesn't refer to an existing parameter group.

HTTP Status Code: 404

DBParameterGroupQuotaExceeded

This request would cause you to exceed the allowed number of parameter groups.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

CopyDBClusterSnapshot

Copies a snapshot of a cluster.

To copy a cluster snapshot from a shared manual cluster snapshot, `SourceDBClusterSnapshotIdentifier` must be the Amazon Resource Name (ARN) of the shared cluster snapshot. You can only copy a shared DB cluster snapshot, whether encrypted or not, in the same AWS Region.

To cancel the copy operation after it is in progress, delete the target cluster snapshot identified by `TargetDBClusterSnapshotIdentifier` while that cluster snapshot is in the *copying* status.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

SourceDBClusterSnapshotIdentifier

The identifier of the cluster snapshot to copy. This parameter is not case sensitive.

Constraints:

- Must specify a valid system snapshot in the *available* state.
- If the source snapshot is in the same AWS Region as the copy, specify a valid snapshot identifier.
- If the source snapshot is in a different AWS Region than the copy, specify a valid cluster snapshot ARN.

Example: `my-cluster-snapshot1`

Type: String

Required: Yes

TargetDBClusterSnapshotIdentifier

The identifier of the new cluster snapshot to create from the source cluster snapshot. This parameter is not case sensitive.

Constraints:

- Must contain from 1 to 63 letters, numbers, or hyphens.
- The first character must be a letter.
- Cannot end with a hyphen or contain two consecutive hyphens.

Example: `my-cluster-snapshot2`

Type: String

Required: Yes

CopyTags

Set to `true` to copy all tags from the source cluster snapshot to the target cluster snapshot, and otherwise `false`. The default is `false`.

Type: Boolean

Required: No

KmsKeyId

The AWS KMS key ID for an encrypted cluster snapshot. The AWS KMS key ID is the Amazon Resource Name (ARN), AWS KMS key identifier, or the AWS KMS key alias for the AWS KMS encryption key.

If you copy an encrypted cluster snapshot from your AWS account, you can specify a value for `KmsKeyId` to encrypt the copy with a new AWS KMS encryption key. If you don't specify a value for `KmsKeyId`, then the copy of the cluster snapshot is encrypted with the same AWS KMS key as the source cluster snapshot.

If you copy an encrypted cluster snapshot that is shared from another AWS account, then you must specify a value for `KmsKeyId`.

To copy an encrypted cluster snapshot to another AWS Region, set `KmsKeyId` to the AWS KMS key ID that you want to use to encrypt the copy of the cluster snapshot in the destination Region. AWS KMS encryption keys are specific to the AWS Region that they are created in, and you can't use encryption keys from one AWS Region in another AWS Region.

If you copy an unencrypted cluster snapshot and specify a value for the `KmsKeyId` parameter, an error is returned.

Type: String

Required: No

PreSignedUrl

The URL that contains a Signature Version 4 signed request for the `CopyDBClusterSnapshot` API action in the AWS Region that contains the source cluster snapshot to copy. You must use the `PreSignedUrl` parameter when copying a cluster snapshot from another AWS Region.

If you are using an AWS SDK tool or the AWS CLI, you can specify `SourceRegion` (or `--source-region` for the AWS CLI) instead of specifying `PreSignedUrl` manually. Specifying `SourceRegion` autogenerates a pre-signed URL that is a valid request for the operation that can be executed in the source AWS Region.

The presigned URL must be a valid request for the `CopyDBClusterSnapshot` API action that can be executed in the source AWS Region that contains the cluster snapshot to be copied. The presigned URL request must contain the following parameter values:

- `SourceRegion` - The ID of the region that contains the snapshot to be copied.
- `SourceDBClusterSnapshotIdentifier` - The identifier for the the encrypted cluster snapshot to be copied. This identifier must be in the Amazon Resource Name (ARN) format for the source AWS Region. For example, if you are copying an encrypted cluster snapshot from the `us-east-1` AWS Region, then your `SourceDBClusterSnapshotIdentifier` looks something like the following: `arn:aws:rds:us-east-1:12345678012:sample-cluster:sample-cluster-snapshot`.
- `TargetDBClusterSnapshotIdentifier` - The identifier for the new cluster snapshot to be created. This parameter isn't case sensitive.

Type: String

Required: No

Tags.Tag.N

The tags to be assigned to the cluster snapshot.

Type: Array of [Tag \(p. 753\)](#) objects

Required: No

Response Elements

The following element is returned by the service.

DBClusterSnapshot

Detailed information about a cluster snapshot.

Type: [DBClusterSnapshot \(p. 716\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBClusterSnapshotAlreadyExistsFault

You already have a cluster snapshot with the given identifier.

HTTP Status Code: 400

DBClusterSnapshotNotFoundFault

`DBClusterSnapshotIdentifier` doesn't refer to an existing cluster snapshot.

HTTP Status Code: 404

InvalidDBClusterSnapshotStateFault

The provided value isn't a valid cluster snapshot state.

HTTP Status Code: 400

InvalidDBClusterStateFault

The cluster isn't in a valid state.

HTTP Status Code: 400

KMSKeyNotAccessibleFault

An error occurred when accessing an AWS KMS key.

HTTP Status Code: 400

SnapshotQuotaExceeded

The request would cause you to exceed the allowed number of snapshots.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)

- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

CreateDBCluster

Creates a new Amazon DocumentDB cluster.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBClusterIdentifier

The cluster identifier. This parameter is stored as a lowercase string.

Constraints:

- Must contain from 1 to 63 letters, numbers, or hyphens.
- The first character must be a letter.
- Cannot end with a hyphen or contain two consecutive hyphens.

Example: my-cluster

Type: String

Required: Yes

Engine

The name of the database engine to be used for this cluster.

Valid values: docdb

Type: String

Required: Yes

AvailabilityZones.AvailabilityZone.N

A list of Amazon EC2 Availability Zones that instances in the cluster can be created in.

Type: Array of strings

Required: No

BackupRetentionPeriod

The number of days for which automated backups are retained. You must specify a minimum value of 1.

Default: 1

Constraints:

- Must be a value from 1 to 35.

Type: Integer

Required: No

DBClusterParameterGroupName

The name of the cluster parameter group to associate with this cluster.

Type: String

Required: No

DBSubnetGroupName

A subnet group to associate with this cluster.

Constraints: Must match the name of an existing DBSubnetGroup. Must not be default.

Example: mySubnetgroup

Type: String

Required: No

DeletionProtection

Specifies whether this cluster can be deleted. If DeletionProtection is enabled, the cluster cannot be deleted unless it is modified and DeletionProtection is disabled. DeletionProtection protects clusters from being accidentally deleted.

Type: Boolean

Required: No

EnableCloudwatchLogsExports.member.N

A list of log types that need to be enabled for exporting to Amazon CloudWatch Logs. You can enable audit logs or profiler logs. For more information, see [Auditing Amazon DocumentDB Events](#) and [Profiling Amazon DocumentDB Operations](#).

Type: Array of strings

Required: No

EngineVersion

The version number of the database engine to use. The --engine-version will default to the latest major engine version. For production workloads, we recommend explicitly declaring this parameter with the intended major engine version.

Type: String

Required: No

GlobalClusterIdentifier

The cluster identifier of the new global cluster.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [A-Za-z][0-9A-Za-z-:_]*

Required: No

KmsKeyId

The AWS KMS key identifier for an encrypted cluster.

The AWS KMS key identifier is the Amazon Resource Name (ARN) for the AWS KMS encryption key. If you are creating a cluster using the same AWS account that owns the AWS KMS encryption key that is used to encrypt the new cluster, you can use the AWS KMS key alias instead of the ARN for the AWS KMS encryption key.

If an encryption key is not specified in KmsKeyId:

- If the StorageEncrypted parameter is true, Amazon DocumentDB uses your default encryption key.

AWS KMS creates the default encryption key for your AWS account. Your AWS account has a different default encryption key for each AWS Regions.

Type: String

Required: No

MasterUsername

The name of the master user for the cluster.

Constraints:

- Must be from 1 to 63 letters or numbers.
- The first character must be a letter.
- Cannot be a reserved word for the chosen database engine.

Type: String

Required: No

MasterUserPassword

The password for the master database user. This password can contain any printable ASCII character except forward slash (/), double quote ("), or the "at" symbol (@).

Constraints: Must contain from 8 to 100 characters.

Type: String

Required: No

Port

The port number on which the instances in the cluster accept connections.

Type: Integer

Required: No

PreferredBackupWindow

The daily time range during which automated backups are created if automated backups are enabled using the BackupRetentionPeriod parameter.

The default is a 30-minute window selected at random from an 8-hour block of time for each AWS Region.

Constraints:

- Must be in the format hh24:mi - hh24:mi.
- Must be in Universal Coordinated Time (UTC).
- Must not conflict with the preferred maintenance window.
- Must be at least 30 minutes.

Type: String

Required: No

PreferredMaintenanceWindow

The weekly time range during which system maintenance can occur, in Universal Coordinated Time (UTC).

Format: ddd:hh24:mi-ddd:hh24:mi

The default is a 30-minute window selected at random from an 8-hour block of time for each AWS Region, occurring on a random day of the week.

Valid days: Mon, Tue, Wed, Thu, Fri, Sat, Sun

Constraints: Minimum 30-minute window.

Type: String

Required: No

PreSignedUrl

Not currently supported.

Type: String

Required: No

StorageEncrypted

Specifies whether the cluster is encrypted.

Type: Boolean

Required: No

Tags.Tag.N

The tags to be assigned to the cluster.

Type: Array of [Tag \(p. 753\)](#) objects

Required: No

VpcSecurityGroupIds.VpcSecurityGroupId.N

A list of EC2 VPC security groups to associate with this cluster.

Type: Array of strings

Required: No

Response Elements

The following element is returned by the service.

DBCluster

Detailed information about a cluster.

Type: [DBCluster \(p. 708\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBClusterAlreadyExistsFault

You already have a cluster with the given identifier.

HTTP Status Code: 400

DBClusterNotFoundFault

`DBClusterIdentifier` doesn't refer to an existing cluster.

HTTP Status Code: 404

DBClusterParameterGroupNotFound

`DBClusterParameterGroupName` doesn't refer to an existing cluster parameter group.

HTTP Status Code: 404

DBClusterQuotaExceededFault

The cluster can't be created because you have reached the maximum allowed quota of clusters.

HTTP Status Code: 403

DBInstanceNotFound

`DBInstanceIdentifier` doesn't refer to an existing instance.

HTTP Status Code: 404

DBSubnetGroupDoesNotCoverEnoughAZs

Subnets in the subnet group should cover at least two Availability Zones unless there is only one Availability Zone.

HTTP Status Code: 400

DBSubnetGroupNotFoundFault

`DBSubnetGroupName` doesn't refer to an existing subnet group.

HTTP Status Code: 404

GlobalClusterNotFoundFault

The `GlobalClusterIdentifier` doesn't refer to an existing global cluster.

HTTP Status Code: 404

InsufficientStorageClusterCapacity

There is not enough storage available for the current action. You might be able to resolve this error by updating your subnet group to use different Availability Zones that have more storage available.

HTTP Status Code: 400

InvalidDBClusterStateFault

The cluster isn't in a valid state.

HTTP Status Code: 400

InvalidDBInstanceState

The specified instance isn't in the *available* state.

HTTP Status Code: 400

InvalidDBSubnetGroupStateFault

The subnet group can't be deleted because it's in use.

HTTP Status Code: 400

InvalidGlobalClusterStateFault

The requested operation can't be performed while the cluster is in this state.

HTTP Status Code: 400

InvalidSubnet

The requested subnet is not valid, or multiple subnets were requested that are not all in a common virtual private cloud (VPC).

HTTP Status Code: 400

InvalidVPCNetworkStateFault

The subnet group doesn't cover all Availability Zones after it is created because of changes that were made.

HTTP Status Code: 400

KMSKeyNotAccessibleFault

An error occurred when accessing an AWS KMS key.

HTTP Status Code: 400

StorageQuotaExceeded

The request would cause you to exceed the allowed amount of storage available across all instances.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

CreateDBClusterParameterGroup

Creates a new cluster parameter group.

Parameters in a cluster parameter group apply to all of the instances in a cluster.

A cluster parameter group is initially created with the default parameters for the database engine used by instances in the cluster. In Amazon DocumentDB, you cannot make modifications directly to the default.docdb3.6 cluster parameter group. If your Amazon DocumentDB cluster is using the default cluster parameter group and you want to modify a value in it, you must first [create a new parameter group](#) or [copy an existing parameter group](#), modify it, and then apply the modified parameter group to your cluster. For the new cluster parameter group and associated settings to take effect, you must then reboot the instances in the cluster without failover. For more information, see [Modifying Amazon DocumentDB Cluster Parameter Groups](#).

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBClusterParameterGroupName

The name of the cluster parameter group.

Constraints:

- Must not match the name of an existing DBClusterParameterGroup.

Note

This value is stored as a lowercase string.

Type: String

Required: Yes

DBParameterGroupFamily

The cluster parameter group family name.

Type: String

Required: Yes

Description

The description for the cluster parameter group.

Type: String

Required: Yes

Tags.Tag.N

The tags to be assigned to the cluster parameter group.

Type: Array of [Tag \(p. 753\)](#) objects

Required: No

Response Elements

The following element is returned by the service.

DBClusterParameterGroup

Detailed information about a cluster parameter group.

Type: [DBClusterParameterGroup \(p. 714\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBParameterGroupAlreadyExists

A parameter group with the same name already exists.

HTTP Status Code: 400

DBParameterGroupQuotaExceeded

This request would cause you to exceed the allowed number of parameter groups.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

CreateDBClusterSnapshot

Creates a snapshot of a cluster.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBClusterIdentifier

The identifier of the cluster to create a snapshot for. This parameter is not case sensitive.

Constraints:

- Must match the identifier of an existing DBCluster.

Example: my-cluster

Type: String

Required: Yes

DBClusterSnapshotIdentifier

The identifier of the cluster snapshot. This parameter is stored as a lowercase string.

Constraints:

- Must contain from 1 to 63 letters, numbers, or hyphens.
- The first character must be a letter.
- Cannot end with a hyphen or contain two consecutive hyphens.

Example: my-cluster-snapshot1

Type: String

Required: Yes

Tags.Tag.N

The tags to be assigned to the cluster snapshot.

Type: Array of [Tag \(p. 753\)](#) objects

Required: No

Response Elements

The following element is returned by the service.

DBClusterSnapshot

Detailed information about a cluster snapshot.

Type: [DBClusterSnapshot \(p. 716\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBClusterNotFoundFault

DBClusterIdentifier doesn't refer to an existing cluster.

HTTP Status Code: 404

DBClusterSnapshotAlreadyExistsFault

You already have a cluster snapshot with the given identifier.

HTTP Status Code: 400

InvalidDBClusterSnapshotStateFault

The provided value isn't a valid cluster snapshot state.

HTTP Status Code: 400

InvalidDBClusterStateFault

The cluster isn't in a valid state.

HTTP Status Code: 400

SnapshotQuotaExceeded

The request would cause you to exceed the allowed number of snapshots.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

CreateDBInstance

Creates a new instance.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBClusterIdentifier

The identifier of the cluster that the instance will belong to.

Type: String

Required: Yes

DBInstanceClass

The compute and memory capacity of the instance; for example, db.r5.large.

Type: String

Required: Yes

DBInstanceIdentifier

The instance identifier. This parameter is stored as a lowercase string.

Constraints:

- Must contain from 1 to 63 letters, numbers, or hyphens.
- The first character must be a letter.
- Cannot end with a hyphen or contain two consecutive hyphens.

Example: mydbinstance

Type: String

Required: Yes

Engine

The name of the database engine to be used for this instance.

Valid value: docdb

Type: String

Required: Yes

AutoMinorVersionUpgrade

This parameter does not apply to Amazon DocumentDB. Amazon DocumentDB does not perform minor version upgrades regardless of the value set.

Default: false

Type: Boolean

Required: No

AvailabilityZone

The Amazon EC2 Availability Zone that the instance is created in.

Default: A random, system-chosen Availability Zone in the endpoint's AWS Region.

Example: us-east-1d

Type: String

Required: No

CopyTagsToSnapshot

A value that indicates whether to copy tags from the DB instance to snapshots of the DB instance. By default, tags are not copied.

Type: Boolean

Required: No

EnablePerformanceInsights

A value that indicates whether to enable Performance Insights for the DB Instance. For more information, see [Using Amazon Performance Insights](#).

Type: Boolean

Required: No

PerformanceInsightsKMSKeyId

The AWS KMS key identifier for encryption of Performance Insights data.

The AWS KMS key identifier is the key ARN, key ID, alias ARN, or alias name for the KMS key.

If you do not specify a value for PerformanceInsightsKMSKeyId, then Amazon DocumentDB uses your default KMS key. There is a default KMS key for your Amazon Web Services account. Your Amazon Web Services account has a different default KMS key for each Amazon Web Services region.

Type: String

Required: No

PreferredMaintenanceWindow

The time range each week during which system maintenance can occur, in Universal Coordinated Time (UTC).

Format: ddd:hh24:mi-ddd:hh24:mi

The default is a 30-minute window selected at random from an 8-hour block of time for each AWS Region, occurring on a random day of the week.

Valid days: Mon, Tue, Wed, Thu, Fri, Sat, Sun

Constraints: Minimum 30-minute window.

Type: String

Required: No

PromotionTier

A value that specifies the order in which an Amazon DocumentDB replica is promoted to the primary instance after a failure of the existing primary instance.

Default: 1

Valid values: 0-15

Type: Integer

Required: No

Tags.Tag.N

The tags to be assigned to the instance. You can assign up to 10 tags to an instance.

Type: Array of [Tag \(p. 753\)](#) objects

Required: No

Response Elements

The following element is returned by the service.

DBInstance

Detailed information about an instance.

Type: [DBInstance \(p. 723\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

AuthorizationNotFound

The specified CIDR IP or Amazon EC2 security group isn't authorized for the specified security group.

Amazon DocumentDB also might not be authorized to perform necessary actions on your behalf using IAM.

HTTP Status Code: 404

DBClusterNotFoundFault

DBClusterIdentifier doesn't refer to an existing cluster.

HTTP Status Code: 404

DBInstanceAlreadyExists

You already have a instance with the given identifier.

HTTP Status Code: 400

DBParameterGroupNotFound

DBParameterGroupName doesn't refer to an existing parameter group.

HTTP Status Code: 404

DBSecurityGroupNotFound

DBSecurityGroupName doesn't refer to an existing security group.

HTTP Status Code: 404

DBSubnetGroupDoesNotCoverEnoughAZs

Subnets in the subnet group should cover at least two Availability Zones unless there is only one Availability Zone.

HTTP Status Code: 400

DBSubnetGroupNotFoundFault

DBSubnetGroupName doesn't refer to an existing subnet group.

HTTP Status Code: 404

InstanceQuotaExceeded

The request would cause you to exceed the allowed number of instances.

HTTP Status Code: 400

InsufficientDBInstanceCapacity

The specified instance class isn't available in the specified Availability Zone.

HTTP Status Code: 400

InvalidDBClusterStateFault

The cluster isn't in a valid state.

HTTP Status Code: 400

InvalidSubnet

The requested subnet is not valid, or multiple subnets were requested that are not all in a common virtual private cloud (VPC).

HTTP Status Code: 400

InvalidVPCNetworkStateFault

The subnet group doesn't cover all Availability Zones after it is created because of changes that were made.

HTTP Status Code: 400

KMSKeyNotAccessibleFault

An error occurred when accessing an AWS KMS key.

HTTP Status Code: 400

StorageQuotaExceeded

The request would cause you to exceed the allowed amount of storage available across all instances.

HTTP Status Code: 400

StorageTypeNotSupported

Storage of the specified StorageType can't be associated with the DB instance.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

CreateDBSubnetGroup

Creates a new subnet group. Subnet groups must contain at least one subnet in at least two Availability Zones in the AWS Region.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBSubnetGroupDescription

The description for the subnet group.

Type: String

Required: Yes

DBSubnetGroupName

The name for the subnet group. This value is stored as a lowercase string.

Constraints: Must contain no more than 255 letters, numbers, periods, underscores, spaces, or hyphens. Must not be default.

Example: mySubnetgroup

Type: String

Required: Yes

SubnetIds.SubnetIdentifier.N

The Amazon EC2 subnet IDs for the subnet group.

Type: Array of strings

Required: Yes

Tags.Tag.N

The tags to be assigned to the subnet group.

Type: Array of [Tag \(p. 753\)](#) objects

Required: No

Response Elements

The following element is returned by the service.

DBSubnetGroup

Detailed information about a subnet group.

Type: [DBSubnetGroup \(p. 728\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBSubnetGroupAlreadyExists

DBSubnetGroupName is already being used by an existing subnet group.

HTTP Status Code: 400

DBSubnetGroupDoesNotCoverEnoughAZs

Subnets in the subnet group should cover at least two Availability Zones unless there is only one Availability Zone.

HTTP Status Code: 400

DBSubnetGroupQuotaExceeded

The request would cause you to exceed the allowed number of subnet groups.

HTTP Status Code: 400

DBSubnetQuotaExceededFault

The request would cause you to exceed the allowed number of subnets in a subnet group.

HTTP Status Code: 400

InvalidSubnet

The requested subnet is not valid, or multiple subnets were requested that are not all in a common virtual private cloud (VPC).

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

CreateEventSubscription

Creates an Amazon DocumentDB event notification subscription. This action requires a topic Amazon Resource Name (ARN) created by using the Amazon DocumentDB console, the Amazon SNS console, or the Amazon SNS API. To obtain an ARN with Amazon SNS, you must create a topic in Amazon SNS and subscribe to the topic. The ARN is displayed in the Amazon SNS console.

You can specify the type of source (`SourceType`) that you want to be notified of. You can also provide a list of Amazon DocumentDB sources (`SourceIds`) that trigger the events, and you can provide a list of event categories (`EventCategories`) for events that you want to be notified of. For example, you can specify `SourceType = db-instance`, `SourceIds = mydbinstance1, mydbinstance2` and `EventCategories = Availability, Backup`.

If you specify both the `SourceType` and `SourceIds` (such as `SourceType = db-instance` and `SourceIdentifier = myDBInstance1`), you are notified of all the `db-instance` events for the specified source. If you specify a `SourceType` but do not specify a `SourceIdentifier`, you receive notice of the events for that source type for all your Amazon DocumentDB sources. If you do not specify either the `SourceType` or the `SourceIdentifier`, you are notified of events generated from all Amazon DocumentDB sources belonging to your customer account.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

SnsTopicArn

The Amazon Resource Name (ARN) of the SNS topic created for event notification. Amazon SNS creates the ARN when you create a topic and subscribe to it.

Type: String

Required: Yes

SubscriptionName

The name of the subscription.

Constraints: The name must be fewer than 255 characters.

Type: String

Required: Yes

Enabled

A Boolean value; set to `true` to activate the subscription, set to `false` to create the subscription but not active it.

Type: Boolean

Required: No

EventCategories.EventCategory.N

A list of event categories for a `SourceType` that you want to subscribe to.

Type: Array of strings

Required: No

SourceIds.SourceId.N

The list of identifiers of the event sources for which events are returned. If not specified, then all sources are included in the response. An identifier must begin with a letter and must contain only ASCII letters, digits, and hyphens; it can't end with a hyphen or contain two consecutive hyphens.

Constraints:

- If SourceIds are provided, SourceType must also be provided.
- If the source type is an instance, a DBInstanceIdentifier must be provided.
- If the source type is a security group, a DBSecurityGroupName must be provided.
- If the source type is a parameter group, a DBParameterGroupName must be provided.
- If the source type is a snapshot, a DBSnapshotIdentifier must be provided.

Type: Array of strings

Required: No

SourceType

The type of source that is generating the events. For example, if you want to be notified of events generated by an instance, you would set this parameter to db-instance. If this value is not specified, all events are returned.

Valid values: db-instance, db-cluster, db-parameter-group, db-security-group, db-cluster-snapshot

Type: String

Required: No

Tags.Tag.N

The tags to be assigned to the event subscription.

Type: Array of [Tag \(p. 753\)](#) objects

Required: No

Response Elements

The following element is returned by the service.

EventSubscription

Detailed information about an event to which you have subscribed.

Type: [EventSubscription \(p. 735\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

EventSubscriptionQuotaExceeded

You have reached the maximum number of event subscriptions.

HTTP Status Code: 400

SNSInvalidTopic

Amazon SNS has responded that there is a problem with the specified topic.

HTTP Status Code: 400

SNSNoAuthorization

You do not have permission to publish to the SNS topic Amazon Resource Name (ARN).

HTTP Status Code: 400

SNSTopicArnNotFound

The SNS topic Amazon Resource Name (ARN) does not exist.

HTTP Status Code: 404

SourceNotFound

The requested source could not be found.

HTTP Status Code: 404

SubscriptionAlreadyExist

The provided subscription name already exists.

HTTP Status Code: 400

SubscriptionCategoryNotFound

The provided category does not exist.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

CreateGlobalCluster

Creates an Amazon DocumentDB global cluster that can span multiple multiple AWS Regions. The global cluster contains one primary cluster with read-write capability, and up-to give read-only secondary clusters. Global clusters uses storage-based fast replication across regions with latencies less than one second, using dedicated infrastructure with no impact to your workload's performance.

You can create a global cluster that is initially empty, and then add a primary and a secondary to it. Or you can specify an existing cluster during the create operation, and this cluster becomes the primary of the global cluster.

Note

This action only applies to Amazon DocumentDB clusters.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

GlobalClusterIdentifier

The cluster identifier of the new global cluster.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [A-Za-z][0-9A-Za-z-:_]*

Required: Yes

DatabaseName

The name for your database of up to 64 alpha-numeric characters. If you do not provide a name, Amazon DocumentDB will not create a database in the global cluster you are creating.

Type: String

Required: No

DeletionProtection

The deletion protection setting for the new global cluster. The global cluster can't be deleted when deletion protection is enabled.

Type: Boolean

Required: No

Engine

The name of the database engine to be used for this cluster.

Type: String

Required: No

EngineVersion

The engine version of the global cluster.

Type: String

Required: No

SourceDBClusterIdentifier

The Amazon Resource Name (ARN) to use as the primary cluster of the global cluster. This parameter is optional.

Type: String

Required: No

StorageEncrypted

The storage encryption setting for the new global cluster.

Type: Boolean

Required: No

Response Elements

The following element is returned by the service.

GlobalCluster

A data type representing an Amazon DocumentDB global cluster.

Type: [GlobalCluster \(p. 738\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBClusterNotFoundFault

`DBClusterIdentifier` doesn't refer to an existing cluster.

HTTP Status Code: 404

GlobalClusterAlreadyExistsFault

The `GlobalClusterIdentifier` already exists. Choose a new global cluster identifier (unique name) to create a new global cluster.

HTTP Status Code: 400

GlobalClusterQuotaExceededFault

The number of global clusters for this account is already at the maximum allowed.

HTTP Status Code: 400

InvalidDBClusterStateFault

The cluster isn't in a valid state.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DeleteDBCluster

Deletes a previously provisioned cluster. When you delete a cluster, all automated backups for that cluster are deleted and can't be recovered. Manual DB cluster snapshots of the specified cluster are not deleted.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBClusterIdentifier

The cluster identifier for the cluster to be deleted. This parameter isn't case sensitive.

Constraints:

- Must match an existing DBClusterIdentifier.

Type: String

Required: Yes

FinalDBSnapshotIdentifier

The cluster snapshot identifier of the new cluster snapshot created when SkipFinalSnapshot is set to false.

Note

Specifying this parameter and also setting the SkipFinalSnapshot parameter to true results in an error.

Constraints:

- Must be from 1 to 255 letters, numbers, or hyphens.
- The first character must be a letter.
- Cannot end with a hyphen or contain two consecutive hyphens.

Type: String

Required: No

SkipFinalSnapshot

Determines whether a final cluster snapshot is created before the cluster is deleted. If true is specified, no cluster snapshot is created. If false is specified, a cluster snapshot is created before the DB cluster is deleted.

Note

If SkipFinalSnapshot is false, you must specify a FinalDBSnapshotIdentifier parameter.

Default: false

Type: Boolean

Required: No

Response Elements

The following element is returned by the service.

DBCluster

Detailed information about a cluster.

Type: [DBCluster \(p. 708\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBClusterNotFoundFault

`DBClusterIdentifier` doesn't refer to an existing cluster.

HTTP Status Code: 404

DBClusterSnapshotAlreadyExistsFault

You already have a cluster snapshot with the given identifier.

HTTP Status Code: 400

InvalidDBClusterSnapshotStateFault

The provided value isn't a valid cluster snapshot state.

HTTP Status Code: 400

InvalidDBClusterStateFault

The cluster isn't in a valid state.

HTTP Status Code: 400

SnapshotQuotaExceeded

The request would cause you to exceed the allowed number of snapshots.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DeleteDBClusterParameterGroup

Deletes a specified cluster parameter group. The cluster parameter group to be deleted can't be associated with any clusters.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBClusterParameterGroupName

The name of the cluster parameter group.

Constraints:

- Must be the name of an existing cluster parameter group.
- You can't delete a default cluster parameter group.
- Cannot be associated with any clusters.

Type: String

Required: Yes

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBParameterGroupNotFound

DBParameterGroupName doesn't refer to an existing parameter group.

HTTP Status Code: 404

InvalidDBParameterGroupState

The parameter group is in use, or it is in a state that is not valid. If you are trying to delete the parameter group, you can't delete it when the parameter group is in this state.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DeleteDBClusterSnapshot

Deletes a cluster snapshot. If the snapshot is being copied, the copy operation is terminated.

Note

The cluster snapshot must be in the available state to be deleted.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBClusterSnapshotIdentifier

The identifier of the cluster snapshot to delete.

Constraints: Must be the name of an existing cluster snapshot in the available state.

Type: String

Required: Yes

Response Elements

The following element is returned by the service.

DBClusterSnapshot

Detailed information about a cluster snapshot.

Type: [DBClusterSnapshot \(p. 716\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBClusterSnapshotNotFoundFault

`DBClusterSnapshotIdentifier` doesn't refer to an existing cluster snapshot.

HTTP Status Code: 404

InvalidDBClusterSnapshotStateFault

The provided value isn't a valid cluster snapshot state.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)

- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DeleteDBInstance

Deletes a previously provisioned instance.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBInstanceIdentifier

The instance identifier for the instance to be deleted. This parameter isn't case sensitive.

Constraints:

- Must match the name of an existing instance.

Type: String

Required: Yes

Response Elements

The following element is returned by the service.

DBInstance

Detailed information about an instance.

Type: [DBInstance \(p. 723\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBInstanceNotFound

`DBInstanceIdentifier` doesn't refer to an existing instance.

HTTP Status Code: 404

DBSnapshotAlreadyExists

`DBSnapshotIdentifier` is already being used by an existing snapshot.

HTTP Status Code: 400

InvalidDBClusterStateFault

The cluster isn't in a valid state.

HTTP Status Code: 400

InvalidDBInstanceState

The specified instance isn't in the *available* state.

HTTP Status Code: 400

SnapshotQuotaExceeded

The request would cause you to exceed the allowed number of snapshots.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DeleteDBSubnetGroup

Deletes a subnet group.

Note

The specified database subnet group must not be associated with any DB instances.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBSubnetGroupName

The name of the database subnet group to delete.

Note

You can't delete the default subnet group.

Constraints:

Must match the name of an existing DBSubnetGroup. Must not be default.

Example: mySubnetgroup

Type: String

Required: Yes

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBSubnetGroupNotFoundFault

DBSubnetGroupName doesn't refer to an existing subnet group.

HTTP Status Code: 404

InvalidDBSubnetGroupStateFault

The subnet group can't be deleted because it's in use.

HTTP Status Code: 400

InvalidDBSubnetStateFault

The subnet isn't in the *available* state.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)

- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DeleteEventSubscription

Deletes an Amazon DocumentDB event notification subscription.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

SubscriptionName

The name of the Amazon DocumentDB event notification subscription that you want to delete.

Type: String

Required: Yes

Response Elements

The following element is returned by the service.

EventSubscription

Detailed information about an event to which you have subscribed.

Type: [EventSubscription \(p. 735\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

InvalidEventSubscriptionState

Someone else might be modifying a subscription. Wait a few seconds, and try again.

HTTP Status Code: 400

SubscriptionNotFound

The subscription name does not exist.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)

- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DeleteGlobalCluster

Deletes a global cluster. The primary and secondary clusters must already be detached or deleted before attempting to delete a global cluster.

Note

This action only applies to Amazon DocumentDB clusters.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

GlobalClusterIdentifier

The cluster identifier of the global cluster being deleted.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [A-Za-z][0-9A-Za-z-:_]*

Required: Yes

Response Elements

The following element is returned by the service.

GlobalCluster

A data type representing an Amazon DocumentDB global cluster.

Type: [GlobalCluster \(p. 738\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

GlobalClusterNotFoundFault

The `GlobalClusterIdentifier` doesn't refer to an existing global cluster.

HTTP Status Code: 404

InvalidGlobalClusterStateFault

The requested operation can't be performed while the cluster is in this state.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)

- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeCertificates

Returns a list of certificate authority (CA) certificates provided by Amazon DocumentDB for this AWS account.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

CertificateIdentifier

The user-supplied certificate identifier. If this parameter is specified, information for only the specified certificate is returned. If this parameter is omitted, a list of up to MaxRecords certificates is returned. This parameter is not case sensitive.

Constraints

- Must match an existing CertificateIdentifier.

Type: String

Required: No

Filters.Filter.N

This parameter is not currently supported.

Type: Array of [Filter \(p. 737\)](#) objects

Required: No

Marker

An optional pagination token provided by a previous DescribeCertificates request. If this parameter is specified, the response includes only records beyond the marker, up to the value specified by MaxRecords.

Type: String

Required: No

MaxRecords

The maximum number of records to include in the response. If more records exist than the specified MaxRecords value, a pagination token called a marker is included in the response so that the remaining results can be retrieved.

Default: 100

Constraints:

- Minimum: 20
- Maximum: 100

Type: Integer

Required: No

Response Elements

The following elements are returned by the service.

Certificates.Certificate.N

A list of certificates for this AWS account.

Type: Array of [Certificate \(p. 705\)](#) objects

Marker

An optional pagination token provided if the number of records retrieved is greater than MaxRecords. If this parameter is specified, the marker specifies the next record in the list. Including the value of Marker in the next call to DescribeCertificates results in the next page of certificates.

Type: String

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

CertificateNotFound

`CertificateIdentifier` doesn't refer to an existing certificate.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeDBClusterParameterGroups

Returns a list of `DBClusterParameterGroup` descriptions. If a `DBClusterParameterGroupName` parameter is specified, the list contains only the description of the specified cluster parameter group.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBClusterParameterGroupName

The name of a specific cluster parameter group to return details for.

Constraints:

- If provided, must match the name of an existing `DBClusterParameterGroup`.

Type: String

Required: No

Filters.Filter.N

This parameter is not currently supported.

Type: Array of [Filter \(p. 737\)](#) objects

Required: No

Marker

An optional pagination token provided by a previous request. If this parameter is specified, the response includes only records beyond the marker, up to the value specified by `MaxRecords`.

Type: String

Required: No

MaxRecords

The maximum number of records to include in the response. If more records exist than the specified `MaxRecords` value, a pagination token (marker) is included in the response so that the remaining results can be retrieved.

Default: 100

Constraints: Minimum 20, maximum 100.

Type: Integer

Required: No

Response Elements

The following elements are returned by the service.

DBClusterParameterGroups.DBClusterParameterGroup.N

A list of cluster parameter groups.

Type: Array of [DBClusterParameterGroup \(p. 714\)](#) objects

Marker

An optional pagination token provided by a previous request. If this parameter is specified, the response includes only records beyond the marker, up to the value specified by `MaxRecords`.

Type: String

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBParameterGroupNotFound

`DBParameterGroupName` doesn't refer to an existing parameter group.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeDBClusterParameters

Returns the detailed parameter list for a particular cluster parameter group.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBClusterParameterGroupName

The name of a specific cluster parameter group to return parameter details for.

Constraints:

- If provided, must match the name of an existing `DBClusterParameterGroup`.

Type: String

Required: Yes

Filters.Filter.N

This parameter is not currently supported.

Type: Array of [Filter \(p. 737\)](#) objects

Required: No

Marker

An optional pagination token provided by a previous request. If this parameter is specified, the response includes only records beyond the marker, up to the value specified by `MaxRecords`.

Type: String

Required: No

MaxRecords

The maximum number of records to include in the response. If more records exist than the specified `MaxRecords` value, a pagination token (marker) is included in the response so that the remaining results can be retrieved.

Default: 100

Constraints: Minimum 20, maximum 100.

Type: Integer

Required: No

Source

A value that indicates to return only parameters for a specific source. Parameter sources can be `engine`, `service`, or `customer`.

Type: String

Required: No

Response Elements

The following elements are returned by the service.

Marker

An optional pagination token provided by a previous request. If this parameter is specified, the response includes only records beyond the marker, up to the value specified by `MaxRecords`.

Type: String

Parameters.Parameter.N

Provides a list of parameters for the cluster parameter group.

Type: Array of [Parameter \(p. 743\)](#) objects

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBParameterGroupNotFound

`DBParameterGroupName` doesn't refer to an existing parameter group.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeDBClusters

Returns information about provisioned Amazon DocumentDB clusters. This API operation supports pagination. For certain management features such as cluster and instance lifecycle management, Amazon DocumentDB leverages operational technology that is shared with Amazon RDS and Amazon Neptune. Use the `filterName=engine,Values=docdb` filter parameter to return only Amazon DocumentDB clusters.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBClusterIdentifier

The user-provided cluster identifier. If this parameter is specified, information from only the specific cluster is returned. This parameter isn't case sensitive.

Constraints:

- If provided, must match an existing DBClusterIdentifier.

Type: String

Required: No

Filters.Filter.N

A filter that specifies one or more clusters to describe.

Supported filters:

- `db-cluster-id` - Accepts cluster identifiers and cluster Amazon Resource Names (ARNs). The results list only includes information about the clusters identified by these ARNs.

Type: Array of [Filter \(p. 737\)](#) objects

Required: No

Marker

An optional pagination token provided by a previous request. If this parameter is specified, the response includes only records beyond the marker, up to the value specified by MaxRecords.

Type: String

Required: No

MaxRecords

The maximum number of records to include in the response. If more records exist than the specified MaxRecords value, a pagination token (marker) is included in the response so that the remaining results can be retrieved.

Default: 100

Constraints: Minimum 20, maximum 100.

Type: Integer

Required: No

Response Elements

The following elements are returned by the service.

DBClusters.DBCluster.N

A list of clusters.

Type: Array of [DBCluster \(p. 708\)](#) objects

Marker

An optional pagination token provided by a previous request. If this parameter is specified, the response includes only records beyond the marker, up to the value specified by MaxRecords.

Type: String

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBClusterNotFoundFault

`DBClusterIdentifier` doesn't refer to an existing cluster.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeDBClusterSnapshotAttributes

Returns a list of cluster snapshot attribute names and values for a manual DB cluster snapshot.

When you share snapshots with other AWS accounts, `DescribeDBClusterSnapshotAttributes` returns the `restore` attribute and a list of IDs for the AWS accounts that are authorized to copy or restore the manual cluster snapshot. If `all` is included in the list of values for the `restore` attribute, then the manual cluster snapshot is public and can be copied or restored by all AWS accounts.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBClusterSnapshotIdentifier

The identifier for the cluster snapshot to describe the attributes for.

Type: String

Required: Yes

Response Elements

The following element is returned by the service.

DBClusterSnapshotAttributesResult

Detailed information about the attributes that are associated with a cluster snapshot.

Type: [DBClusterSnapshotAttributesResult \(p. 720\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBClusterSnapshotNotFoundFault

`DBClusterSnapshotIdentifier` doesn't refer to an existing cluster snapshot.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)

- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeDBClusterSnapshots

Returns information about cluster snapshots. This API operation supports pagination.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBClusterIdentifier

The ID of the cluster to retrieve the list of cluster snapshots for. This parameter can't be used with the `DBClusterSnapshotIdentifier` parameter. This parameter is not case sensitive.

Constraints:

- If provided, must match the identifier of an existing `DBCluster`.

Type: String

Required: No

DBClusterSnapshotIdentifier

A specific cluster snapshot identifier to describe. This parameter can't be used with the `DBClusterIdentifier` parameter. This value is stored as a lowercase string.

Constraints:

- If provided, must match the identifier of an existing `DBClusterSnapshot`.
- If this identifier is for an automated snapshot, the `SnapshotType` parameter must also be specified.

Type: String

Required: No

Filters.Filter.N

This parameter is not currently supported.

Type: Array of [Filter \(p. 737\)](#) objects

Required: No

IncludePublic

Set to `true` to include manual cluster snapshots that are public and can be copied or restored by any AWS account, and otherwise `false`. The default is `false`.

Type: Boolean

Required: No

IncludeShared

Set to `true` to include shared manual cluster snapshots from other AWS accounts that this AWS account has been given permission to copy or restore, and otherwise `false`. The default is `false`.

Type: Boolean

Required: No

Marker

An optional pagination token provided by a previous request. If this parameter is specified, the response includes only records beyond the marker, up to the value specified by `MaxRecords`.

Type: String

Required: No

MaxRecords

The maximum number of records to include in the response. If more records exist than the specified `MaxRecords` value, a pagination token (marker) is included in the response so that the remaining results can be retrieved.

Default: 100

Constraints: Minimum 20, maximum 100.

Type: Integer

Required: No

SnapshotType

The type of cluster snapshots to be returned. You can specify one of the following values:

- `automated` - Return all cluster snapshots that Amazon DocumentDB has automatically created for your AWS account.
- `manual` - Return all cluster snapshots that you have manually created for your AWS account.
- `shared` - Return all manual cluster snapshots that have been shared to your AWS account.
- `public` - Return all cluster snapshots that have been marked as public.

If you don't specify a `SnapshotType` value, then both automated and manual cluster snapshots are returned. You can include shared cluster snapshots with these results by setting the `IncludeShared` parameter to `true`. You can include public cluster snapshots with these results by setting the `IncludePublic` parameter to `true`.

The `IncludeShared` and `IncludePublic` parameters don't apply for `SnapshotType` values of `manual` or `automated`. The `IncludePublic` parameter doesn't apply when `SnapshotType` is set to `shared`. The `IncludeShared` parameter doesn't apply when `SnapshotType` is set to `public`.

Type: String

Required: No

Response Elements

The following elements are returned by the service.

DBClusterSnapshots.DBClusterSnapshot.N

Provides a list of cluster snapshots.

Type: Array of [DBClusterSnapshot \(p. 716\)](#) objects

Marker

An optional pagination token provided by a previous request. If this parameter is specified, the response includes only records beyond the marker, up to the value specified by `MaxRecords`.

Type: String

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBClusterSnapshotNotFoundFault

`DBClusterSnapshotIdentifier` doesn't refer to an existing cluster snapshot.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeDBEngineVersions

Returns a list of the available engines.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBParameterGroupFamily

The name of a specific parameter group family to return details for.

Constraints:

- If provided, must match an existing DBParameterGroupFamily.

Type: String

Required: No

DefaultOnly

Indicates that only the default version of the specified engine or engine and major version combination is returned.

Type: Boolean

Required: No

Engine

The database engine to return.

Type: String

Required: No

EngineVersion

The database engine version to return.

Example: 3.6.0

Type: String

Required: No

Filters.Filter.N

This parameter is not currently supported.

Type: Array of [Filter \(p. 737\)](#) objects

Required: No

ListSupportedCharacterSets

If this parameter is specified and the requested engine supports the CharacterSetName parameter for CreateDBInstance, the response includes a list of supported character sets for each engine version.

Type: Boolean

Required: No

ListSupportedTimezones

If this parameter is specified and the requested engine supports the `TimeZone` parameter for `CreateDBInstance`, the response includes a list of supported time zones for each engine version.

Type: Boolean

Required: No

Marker

An optional pagination token provided by a previous request. If this parameter is specified, the response includes only records beyond the marker, up to the value specified by `MaxRecords`.

Type: String

Required: No

MaxRecords

The maximum number of records to include in the response. If more records exist than the specified `MaxRecords` value, a pagination token (marker) is included in the response so that the remaining results can be retrieved.

Default: 100

Constraints: Minimum 20, maximum 100.

Type: Integer

Required: No

Response Elements

The following elements are returned by the service.

DBEngineVersions.DBEngineVersion.N

Detailed information about one or more engine versions.

Type: Array of [DBEngineVersion \(p. 721\)](#) objects

Marker

An optional pagination token provided by a previous request. If this parameter is specified, the response includes only records beyond the marker, up to the value specified by `MaxRecords`.

Type: String

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeDBInstances

Returns information about provisioned Amazon DocumentDB instances. This API supports pagination.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBInstanceIdentifier

The user-provided instance identifier. If this parameter is specified, information from only the specific instance is returned. This parameter isn't case sensitive.

Constraints:

- If provided, must match the identifier of an existing DBInstance.

Type: String

Required: No

Filters.Filter.N

A filter that specifies one or more instances to describe.

Supported filters:

- db-cluster-id - Accepts cluster identifiers and cluster Amazon Resource Names (ARNs). The results list includes only the information about the instances that are associated with the clusters that are identified by these ARNs.
- db-instance-id - Accepts instance identifiers and instance ARNs. The results list includes only the information about the instances that are identified by these ARNs.

Type: Array of [Filter \(p. 737\)](#) objects

Required: No

Marker

An optional pagination token provided by a previous request. If this parameter is specified, the response includes only records beyond the marker, up to the value specified by MaxRecords.

Type: String

Required: No

MaxRecords

The maximum number of records to include in the response. If more records exist than the specified MaxRecords value, a pagination token (marker) is included in the response so that the remaining results can be retrieved.

Default: 100

Constraints: Minimum 20, maximum 100.

Type: Integer

Required: No

Response Elements

The following elements are returned by the service.

DBInstances.DBInstance.N

Detailed information about one or more instances.

Type: Array of [DBInstance \(p. 723\)](#) objects

Marker

An optional pagination token provided by a previous request. If this parameter is specified, the response includes only records beyond the marker, up to the value specified by MaxRecords.

Type: String

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBInstanceNotFound

`DBInstanceIdentifier` doesn't refer to an existing instance.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeDBSubnetGroups

Returns a list of DBSubnetGroup descriptions. If a DBSubnetGroupName is specified, the list will contain only the descriptions of the specified DBSubnetGroup.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBSubnetGroupName

The name of the subnet group to return details for.

Type: String

Required: No

Filters.Filter.N

This parameter is not currently supported.

Type: Array of [Filter \(p. 737\)](#) objects

Required: No

Marker

An optional pagination token provided by a previous request. If this parameter is specified, the response includes only records beyond the marker, up to the value specified by MaxRecords.

Type: String

Required: No

MaxRecords

The maximum number of records to include in the response. If more records exist than the specified MaxRecords value, a pagination token (marker) is included in the response so that the remaining results can be retrieved.

Default: 100

Constraints: Minimum 20, maximum 100.

Type: Integer

Required: No

Response Elements

The following elements are returned by the service.

DBSubnetGroups.DBSubnetGroup.N

Detailed information about one or more subnet groups.

Type: Array of [DBSubnetGroup \(p. 728\)](#) objects

Marker

An optional pagination token provided by a previous request. If this parameter is specified, the response includes only records beyond the marker, up to the value specified by MaxRecords.

Type: String

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBSubnetGroupNotFoundFault

DBSubnetGroupName doesn't refer to an existing subnet group.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeEngineDefaultClusterParameters

Returns the default engine and system parameter information for the cluster database engine.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBParameterGroupFamily

The name of the cluster parameter group family to return the engine parameter information for.

Type: String

Required: Yes

Filters.Filter.N

This parameter is not currently supported.

Type: Array of [Filter \(p. 737\)](#) objects

Required: No

Marker

An optional pagination token provided by a previous request. If this parameter is specified, the response includes only records beyond the marker, up to the value specified by MaxRecords.

Type: String

Required: No

MaxRecords

The maximum number of records to include in the response. If more records exist than the specified MaxRecords value, a pagination token (marker) is included in the response so that the remaining results can be retrieved.

Default: 100

Constraints: Minimum 20, maximum 100.

Type: Integer

Required: No

Response Elements

The following element is returned by the service.

EngineDefaults

Contains the result of a successful invocation of the `DescribeEngineDefaultClusterParameters` operation.

Type: [EngineDefaults \(p. 731\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeEventCategories

Displays a list of categories for all event source types, or, if specified, for a specified source type.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

Filters.Filter.N

This parameter is not currently supported.

Type: Array of [Filter \(p. 737\)](#) objects

Required: No

SourceType

The type of source that is generating the events.

Valid values: db-instance, db-parameter-group, db-security-group

Type: String

Required: No

Response Elements

The following element is returned by the service.

EventCategoriesMapList.EventCategoriesMap.N

A list of event category maps.

Type: Array of [EventCategoriesMap \(p. 734\)](#) objects

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeEvents

Returns events related to instances, security groups, snapshots, and DB parameter groups for the past 14 days. You can obtain events specific to a particular DB instance, security group, snapshot, or parameter group by providing the name as a parameter. By default, the events of the past hour are returned.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

Duration

The number of minutes to retrieve events for.

Default: 60

Type: Integer

Required: No

EndTime

The end of the time interval for which to retrieve events, specified in ISO 8601 format.

Example: 2009-07-08T18:00Z

Type: Timestamp

Required: No

EventCategories.EventCategory.N

A list of event categories that trigger notifications for an event notification subscription.

Type: Array of strings

Required: No

Filters.Filter.N

This parameter is not currently supported.

Type: Array of [Filter \(p. 737\)](#) objects

Required: No

Marker

An optional pagination token provided by a previous request. If this parameter is specified, the response includes only records beyond the marker, up to the value specified by MaxRecords.

Type: String

Required: No

MaxRecords

The maximum number of records to include in the response. If more records exist than the specified MaxRecords value, a pagination token (marker) is included in the response so that the remaining results can be retrieved.

Default: 100

Constraints: Minimum 20, maximum 100.

Type: Integer

Required: No

SourceIdentifier

The identifier of the event source for which events are returned. If not specified, then all sources are included in the response.

Constraints:

- If `SourceIdentifier` is provided, `SourceType` must also be provided.
- If the source type is `DBInstance`, a `DBInstanceIdentifier` must be provided.
- If the source type is `DBSecurityGroup`, a `DBSecurityGroupName` must be provided.
- If the source type is `DBParameterGroup`, a `DBParameterGroupName` must be provided.
- If the source type is `DBSnapshot`, a `DBSnapshotIdentifier` must be provided.
- Cannot end with a hyphen or contain two consecutive hyphens.

Type: String

Required: No

SourceType

The event source to retrieve events for. If no value is specified, all events are returned.

Type: String

Valid Values: `db-instance` | `db-parameter-group` | `db-security-group` | `db-snapshot` | `db-cluster` | `db-cluster-snapshot`

Required: No

StartTime

The beginning of the time interval to retrieve events for, specified in ISO 8601 format.

Example: `2009-07-08T18:00Z`

Type: Timestamp

Required: No

Response Elements

The following elements are returned by the service.

Events.Event.N

Detailed information about one or more events.

Type: Array of [Event \(p. 732\)](#) objects

Marker

An optional pagination token provided by a previous request. If this parameter is specified, the response includes only records beyond the marker, up to the value specified by `MaxRecords`.

Type: String

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeEventSubscriptions

Lists all the subscription descriptions for a customer account. The description for a subscription includes SubscriptionName, SNSTopicARN, CustomerID, SourceType, SourceID, CreationTime, and Status.

If you specify a SubscriptionName, lists the description for that subscription.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

Filters.Filter.N

This parameter is not currently supported.

Type: Array of [Filter \(p. 737\)](#) objects

Required: No

Marker

An optional pagination token provided by a previous request. If this parameter is specified, the response includes only records beyond the marker, up to the value specified by MaxRecords.

Type: String

Required: No

MaxRecords

The maximum number of records to include in the response. If more records exist than the specified MaxRecords value, a pagination token (marker) is included in the response so that the remaining results can be retrieved.

Default: 100

Constraints: Minimum 20, maximum 100.

Type: Integer

Required: No

SubscriptionName

The name of the Amazon DocumentDB event notification subscription that you want to describe.

Type: String

Required: No

Response Elements

The following elements are returned by the service.

EventSubscriptionsList.EventSubscription.N

A list of event subscriptions.

Type: Array of [EventSubscription \(p. 735\)](#) objects

Marker

An optional pagination token provided by a previous request. If this parameter is specified, the response includes only records beyond the marker, up to the value specified by `MaxRecords`.

Type: String

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

SubscriptionNotFound

The subscription name does not exist.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeGlobalClusters

Returns information about Amazon DocumentDB global clusters. This API supports pagination.

Note

This action only applies to Amazon DocumentDB clusters.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

Filters.Filter.N

A filter that specifies one or more global DB clusters to describe.

Supported filters: db-cluster-id accepts cluster identifiers and cluster Amazon Resource Names (ARNs). The results list will only include information about the clusters identified by these ARNs.

Type: Array of [Filter \(p. 737\)](#) objects

Required: No

GlobalClusterIdentifier

The user-supplied cluster identifier. If this parameter is specified, information from only the specific cluster is returned. This parameter isn't case-sensitive.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [A-Za-z][0-9A-Za-z-:_]*

Required: No

Marker

An optional pagination token provided by a previous `DescribeGlobalClusters` request. If this parameter is specified, the response includes only records beyond the marker, up to the value specified by `MaxRecords`.

Type: String

Required: No

MaxRecords

The maximum number of records to include in the response. If more records exist than the specified `MaxRecords` value, a pagination token called a marker is included in the response so that you can retrieve the remaining results.

Type: Integer

Required: No

Response Elements

The following elements are returned by the service.

GlobalClusters.GlobalClusterMember.N

Type: Array of [GlobalCluster \(p. 738\)](#) objects

Marker

Type: String

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

GlobalClusterNotFoundFault

The `GlobalClusterIdentifier` doesn't refer to an existing global cluster.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeOrderableDBInstanceState

Returns a list of orderable instance options for the specified engine.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

Engine

The name of the engine to retrieve instance options for.

Type: String

Required: Yes

DBInstanceState

The instance class filter value. Specify this parameter to show only the available offerings that match the specified instance class.

Type: String

Required: No

EngineVersion

The engine version filter value. Specify this parameter to show only the available offerings that match the specified engine version.

Type: String

Required: No

Filters.Filter.N

This parameter is not currently supported.

Type: Array of [Filter \(p. 737\)](#) objects

Required: No

LicenseModel

The license model filter value. Specify this parameter to show only the available offerings that match the specified license model.

Type: String

Required: No

Marker

An optional pagination token provided by a previous request. If this parameter is specified, the response includes only records beyond the marker, up to the value specified by MaxRecords.

Type: String

Required: No

MaxRecords

The maximum number of records to include in the response. If more records exist than the specified MaxRecords value, a pagination token (marker) is included in the response so that the remaining results can be retrieved.

Default: 100

Constraints: Minimum 20, maximum 100.

Type: Integer

Required: No

Vpc

The virtual private cloud (VPC) filter value. Specify this parameter to show only the available VPC or non-VPC offerings.

Type: Boolean

Required: No

Response Elements

The following elements are returned by the service.

Marker

An optional pagination token provided by a previous request. If this parameter is specified, the response includes only records beyond the marker, up to the value specified by MaxRecords.

Type: String

OrderableDBInstanceState.OrderableDBInstanceStateOption.N

The options that are available for a particular orderable instance.

Type: Array of [OrderableDBInstanceStateOption \(p. 741\)](#) objects

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribePendingMaintenanceActions

Returns a list of resources (for example, instances) that have at least one pending maintenance action.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

Filters.Filter.N

A filter that specifies one or more resources to return pending maintenance actions for.

Supported filters:

- `db-cluster-id` - Accepts cluster identifiers and cluster Amazon Resource Names (ARNs). The results list includes only pending maintenance actions for the clusters identified by these ARNs.
- `db-instance-id` - Accepts instance identifiers and instance ARNs. The results list includes only pending maintenance actions for the DB instances identified by these ARNs.

Type: Array of [Filter \(p. 737\)](#) objects

Required: No

Marker

An optional pagination token provided by a previous request. If this parameter is specified, the response includes only records beyond the marker, up to the value specified by `MaxRecords`.

Type: String

Required: No

MaxRecords

The maximum number of records to include in the response. If more records exist than the specified `MaxRecords` value, a pagination token (marker) is included in the response so that the remaining results can be retrieved.

Default: 100

Constraints: Minimum 20, maximum 100.

Type: Integer

Required: No

ResourceIdentifier

The ARN of a resource to return pending maintenance actions for.

Type: String

Required: No

Response Elements

The following elements are returned by the service.

Marker

An optional pagination token provided by a previous request. If this parameter is specified, the response includes only records beyond the marker, up to the value specified by `MaxRecords`.

Type: String

PendingMaintenanceActions.ResourcePendingMaintenanceActions.N

The maintenance actions to be applied.

Type: Array of [ResourcePendingMaintenanceActions \(p. 751\)](#) objects

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

ResourceNotFoundFault

The specified resource ID was not found.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

FailoverDBCluster

Forces a failover for a cluster.

A failover for a cluster promotes one of the Amazon DocumentDB replicas (read-only instances) in the cluster to be the primary instance (the cluster writer).

If the primary instance fails, Amazon DocumentDB automatically fails over to an Amazon DocumentDB replica, if one exists. You can force a failover when you want to simulate a failure of a primary instance for testing.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBClusterIdentifier

A cluster identifier to force a failover for. This parameter is not case sensitive.

Constraints:

- Must match the identifier of an existing DBCluster.

Type: String

Required: No

TargetDBInstanceIdentifier

The name of the instance to promote to the primary instance.

You must specify the instance identifier for an Amazon DocumentDB replica in the cluster. For example, mydbcluster-replica1.

Type: String

Required: No

Response Elements

The following element is returned by the service.

DBCluster

Detailed information about a cluster.

Type: [DBCluster \(p. 708\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBClusterNotFoundFault

DBClusterIdentifier doesn't refer to an existing cluster.

HTTP Status Code: 404

InvalidDBClusterStateFault

The cluster isn't in a valid state.

HTTP Status Code: 400

InvalidDBInstanceState

The specified instance isn't in the *available* state.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ListTagsForResource

Lists all tags on an Amazon DocumentDB resource.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

ResourceName

The Amazon DocumentDB resource with tags to be listed. This value is an Amazon Resource Name (ARN).

Type: String

Required: Yes

Filters.Filter.N

This parameter is not currently supported.

Type: Array of [Filter \(p. 737\)](#) objects

Required: No

Response Elements

The following element is returned by the service.

TagList.Tag.N

A list of one or more tags.

Type: Array of [Tag \(p. 753\)](#) objects

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBClusterNotFoundFault

`DBClusterIdentifier` doesn't refer to an existing cluster.

HTTP Status Code: 404

DBInstanceNotFound

`DBInstanceIdentifier` doesn't refer to an existing instance.

HTTP Status Code: 404

DBSnapshotNotFound

`DBSnapshotIdentifier` doesn't refer to an existing snapshot.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ModifyDBCluster

Modifies a setting for an Amazon DocumentDB cluster. You can change one or more database configuration parameters by specifying these parameters and the new values in the request.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBClusterIdentifier

The cluster identifier for the cluster that is being modified. This parameter is not case sensitive.

Constraints:

- Must match the identifier of an existing DBCluster.

Type: String

Required: Yes

ApplyImmediately

A value that specifies whether the changes in this request and any pending changes are asynchronously applied as soon as possible, regardless of the PreferredMaintenanceWindow setting for the cluster. If this parameter is set to false, changes to the cluster are applied during the next maintenance window.

The ApplyImmediately parameter affects only the NewDBClusterIdentifier and MasterUserPassword values. If you set this parameter value to false, the changes to the NewDBClusterIdentifier and MasterUserPassword values are applied during the next maintenance window. All other changes are applied immediately, regardless of the value of the ApplyImmediately parameter.

Default: false

Type: Boolean

Required: No

BackupRetentionPeriod

The number of days for which automated backups are retained. You must specify a minimum value of 1.

Default: 1

Constraints:

- Must be a value from 1 to 35.

Type: Integer

Required: No

CloudwatchLogsExportConfiguration

The configuration setting for the log types to be enabled for export to Amazon CloudWatch Logs for a specific instance or cluster. The EnableLogTypes and DisableLogTypes arrays determine which logs are exported (or not exported) to CloudWatch Logs.

Type: [CloudwatchLogsExportConfiguration \(p. 707\)](#) object

Required: No

DBClusterParameterGroupName

The name of the cluster parameter group to use for the cluster.

Type: String

Required: No

DeletionProtection

Specifies whether this cluster can be deleted. If DeletionProtection is enabled, the cluster cannot be deleted unless it is modified and DeletionProtection is disabled. DeletionProtection protects clusters from being accidentally deleted.

Type: Boolean

Required: No

EngineVersion

The version number of the database engine to which you want to upgrade. Modifying engine version is not supported on Amazon DocumentDB.

Type: String

Required: No

MasterUserPassword

The password for the master database user. This password can contain any printable ASCII character except forward slash (/), double quote ("), or the "at" symbol (@).

Constraints: Must contain from 8 to 100 characters.

Type: String

Required: No

NewDBClusterIdentifier

The new cluster identifier for the cluster when renaming a cluster. This value is stored as a lowercase string.

Constraints:

- Must contain from 1 to 63 letters, numbers, or hyphens.
- The first character must be a letter.
- Cannot end with a hyphen or contain two consecutive hyphens.

Example: my-cluster2

Type: String

Required: No

Port

The port number on which the cluster accepts connections.

Constraints: Must be a value from 1150 to 65535.

Default: The same port as the original cluster.

Type: Integer

Required: No

PreferredBackupWindow

The daily time range during which automated backups are created if automated backups are enabled, using the BackupRetentionPeriod parameter.

The default is a 30-minute window selected at random from an 8-hour block of time for each AWS Region.

Constraints:

- Must be in the format hh24:mi-hh24:mi.
- Must be in Universal Coordinated Time (UTC).
- Must not conflict with the preferred maintenance window.
- Must be at least 30 minutes.

Type: String

Required: No

PreferredMaintenanceWindow

The weekly time range during which system maintenance can occur, in Universal Coordinated Time (UTC).

Format: ddd:hh24:mi-ddd:hh24:mi

The default is a 30-minute window selected at random from an 8-hour block of time for each AWS Region, occurring on a random day of the week.

Valid days: Mon, Tue, Wed, Thu, Fri, Sat, Sun

Constraints: Minimum 30-minute window.

Type: String

Required: No

VpcSecurityGroupIds.VpcSecurityGroupId.N

A list of virtual private cloud (VPC) security groups that the cluster will belong to.

Type: Array of strings

Required: No

Response Elements

The following element is returned by the service.

DBCluster

Detailed information about a cluster.

Type: [DBCluster \(p. 708\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBClusterAlreadyExistsFault

You already have a cluster with the given identifier.

HTTP Status Code: 400

DBClusterNotFoundFault

DBClusterIdentifier doesn't refer to an existing cluster.

HTTP Status Code: 404

DBClusterParameterGroupNotFound

DBClusterParameterGroupName doesn't refer to an existing cluster parameter group.

HTTP Status Code: 404

DBSubnetGroupNotFoundFault

DBSubnetGroupName doesn't refer to an existing subnet group.

HTTP Status Code: 404

InvalidDBClusterStateFault

The cluster isn't in a valid state.

HTTP Status Code: 400

InvalidDBInstanceState

The specified instance isn't in the *available* state.

HTTP Status Code: 400

InvalidDBSecurityGroupState

The state of the security group doesn't allow deletion.

HTTP Status Code: 400

InvalidDBSubnetGroupStateFault

The subnet group can't be deleted because it's in use.

HTTP Status Code: 400

InvalidSubnet

The requested subnet is not valid, or multiple subnets were requested that are not all in a common virtual private cloud (VPC).

HTTP Status Code: 400

InvalidVPCNetworkStateFault

The subnet group doesn't cover all Availability Zones after it is created because of changes that were made.

HTTP Status Code: 400

StorageQuotaExceeded

The request would cause you to exceed the allowed amount of storage available across all instances.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ModifyDBClusterParameterGroup

Modifies the parameters of a cluster parameter group. To modify more than one parameter, submit a list of the following: ParameterName, ParameterValue, and ApplyMethod. A maximum of 20 parameters can be modified in a single request.

Note

Changes to dynamic parameters are applied immediately. Changes to static parameters require a reboot or maintenance window before the change can take effect.

Important

After you create a cluster parameter group, you should wait at least 5 minutes before creating your first cluster that uses that cluster parameter group as the default parameter group. This allows Amazon DocumentDB to fully complete the create action before the parameter group is used as the default for a new cluster. This step is especially important for parameters that are critical when creating the default database for a cluster, such as the character set for the default database defined by the character_set_database parameter.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBClusterParameterGroupName

The name of the cluster parameter group to modify.

Type: String

Required: Yes

Parameters.Parameter.N

A list of parameters in the cluster parameter group to modify.

Type: Array of [Parameter \(p. 743\)](#) objects

Required: Yes

Response Elements

The following element is returned by the service.

DBClusterParameterGroupName

The name of a cluster parameter group.

Constraints:

- Must be from 1 to 255 letters or numbers.
- The first character must be a letter.
- Cannot end with a hyphen or contain two consecutive hyphens.

Note

This value is stored as a lowercase string.

Type: String

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBParameterGroupNotFound

DBParameterGroupName doesn't refer to an existing parameter group.

HTTP Status Code: 404

InvalidDBParameterGroupState

The parameter group is in use, or it is in a state that is not valid. If you are trying to delete the parameter group, you can't delete it when the parameter group is in this state.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ModifyDBClusterSnapshotAttribute

Adds an attribute and values to, or removes an attribute and values from, a manual cluster snapshot.

To share a manual cluster snapshot with other AWS accounts, specify `restore` as the `AttributeName`, and use the `ValuesToAdd` parameter to add a list of IDs of the AWS accounts that are authorized to restore the manual cluster snapshot. Use the value `all` to make the manual cluster snapshot public, which means that it can be copied or restored by all AWS accounts. Do not add the `all` value for any manual cluster snapshots that contain private information that you don't want available to all AWS accounts. If a manual cluster snapshot is encrypted, it can be shared, but only by specifying a list of authorized AWS account IDs for the `ValuesToAdd` parameter. You can't use `all` as a value for that parameter in this case.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

AttributeName

The name of the cluster snapshot attribute to modify.

To manage authorization for other AWS accounts to copy or restore a manual cluster snapshot, set this value to `restore`.

Type: String

Required: Yes

DBClusterSnapshotIdentifier

The identifier for the cluster snapshot to modify the attributes for.

Type: String

Required: Yes

ValuesToAdd.AttributeValue.N

A list of cluster snapshot attributes to add to the attribute specified by `AttributeName`.

To authorize other AWS accounts to copy or restore a manual cluster snapshot, set this list to include one or more AWS account IDs. To make the manual cluster snapshot restorable by any AWS account, set it to `all`. Do not add the `all` value for any manual cluster snapshots that contain private information that you don't want to be available to all AWS accounts.

Type: Array of strings

Required: No

ValuesToRemove.AttributeValue.N

A list of cluster snapshot attributes to remove from the attribute specified by `AttributeName`.

To remove authorization for other AWS accounts to copy or restore a manual cluster snapshot, set this list to include one or more AWS account identifiers. To remove authorization for any AWS account to copy or restore the cluster snapshot, set it to `all`. If you specify `all`, an AWS account whose account ID is explicitly added to the `restore` attribute can still copy or restore a manual cluster snapshot.

Type: Array of strings

Required: No

Response Elements

The following element is returned by the service.

DBClusterSnapshotAttributesResult

Detailed information about the attributes that are associated with a cluster snapshot.

Type: [DBClusterSnapshotAttributesResult \(p. 720\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBClusterSnapshotNotFoundFault

`DBClusterSnapshotIdentifier` doesn't refer to an existing cluster snapshot.

HTTP Status Code: 404

InvalidDBClusterSnapshotStateFault

The provided value isn't a valid cluster snapshot state.

HTTP Status Code: 400

SharedSnapshotQuotaExceeded

You have exceeded the maximum number of accounts that you can share a manual DB snapshot with.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ModifyDBInstance

Modifies settings for an instance. You can change one or more database configuration parameters by specifying these parameters and the new values in the request.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBInstanceIdentifier

The instance identifier. This value is stored as a lowercase string.

Constraints:

- Must match the identifier of an existing DBInstance.

Type: String

Required: Yes

ApplyImmediately

Specifies whether the modifications in this request and any pending modifications are asynchronously applied as soon as possible, regardless of the PreferredMaintenanceWindow setting for the instance.

If this parameter is set to false, changes to the instance are applied during the next maintenance window. Some parameter changes can cause an outage and are applied on the next reboot.

Default: false

Type: Boolean

Required: No

AutoMinorVersionUpgrade

This parameter does not apply to Amazon DocumentDB. Amazon DocumentDB does not perform minor version upgrades regardless of the value set.

Type: Boolean

Required: No

CACertificateIdentifier

Indicates the certificate that needs to be associated with the instance.

Type: String

Required: No

CopyTagsToSnapshot

A value that indicates whether to copy all tags from the DB instance to snapshots of the DB instance. By default, tags are not copied.

Type: Boolean

Required: No

DBInstanceClass

The new compute and memory capacity of the instance; for example, db.r5.large. Not all instance classes are available in all AWS Regions.

If you modify the instance class, an outage occurs during the change. The change is applied during the next maintenance window, unless `ApplyImmediately` is specified as `true` for this request.

Default: Uses existing setting.

Type: String

Required: No

EnablePerformanceInsights

A value that indicates whether to enable Performance Insights for the DB Instance. For more information, see [Using Amazon Performance Insights](#).

Type: Boolean

Required: No

NewDBInstanceIdentifier

The new instance identifier for the instance when renaming an instance. When you change the instance identifier, an instance reboot occurs immediately if you set `Apply Immediately` to `true`. It occurs during the next maintenance window if you set `Apply Immediately` to `false`. This value is stored as a lowercase string.

Constraints:

- Must contain from 1 to 63 letters, numbers, or hyphens.
- The first character must be a letter.
- Cannot end with a hyphen or contain two consecutive hyphens.

Example: mydbinstance

Type: String

Required: No

PerformanceInsightsKMSKeyId

The AWS KMS key identifier for encryption of Performance Insights data.

The AWS KMS key identifier is the key ARN, key ID, alias ARN, or alias name for the KMS key.

If you do not specify a value for `PerformanceInsightsKMSKeyId`, then Amazon DocumentDB uses your default KMS key. There is a default KMS key for your Amazon Web Services account. Your Amazon Web Services account has a different default KMS key for each Amazon Web Services region.

Type: String

Required: No

PreferredMaintenanceWindow

The weekly time range (in UTC) during which system maintenance can occur, which might result in an outage. Changing this parameter doesn't result in an outage except in the following situation, and the change is asynchronously applied as soon as possible. If there are pending actions that cause a reboot, and the maintenance window is changed to include the current time, changing

this parameter causes a reboot of the instance. If you are moving this window to the current time, there must be at least 30 minutes between the current time and end of the window to ensure that pending changes are applied.

Default: Uses existing setting.

Format: ddd:hh24:mi-ddd:hh24:mi

Valid days: Mon, Tue, Wed, Thu, Fri, Sat, Sun

Constraints: Must be at least 30 minutes.

Type: String

Required: No

PromotionTier

A value that specifies the order in which an Amazon DocumentDB replica is promoted to the primary instance after a failure of the existing primary instance.

Default: 1

Valid values: 0-15

Type: Integer

Required: No

Response Elements

The following element is returned by the service.

DBInstance

Detailed information about an instance.

Type: [DBInstance \(p. 723\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

AuthorizationNotFound

The specified CIDR IP or Amazon EC2 security group isn't authorized for the specified security group.

Amazon DocumentDB also might not be authorized to perform necessary actions on your behalf using IAM.

HTTP Status Code: 404

CertificateNotFound

CertificateIdentifier doesn't refer to an existing certificate.

HTTP Status Code: 404

DBInstanceAlreadyExists

You already have a instance with the given identifier.

HTTP Status Code: 400

DBInstanceNotFound

DBInstanceIdentifier doesn't refer to an existing instance.

HTTP Status Code: 404

DBParameterGroupNotFound

DBParameterGroupName doesn't refer to an existing parameter group.

HTTP Status Code: 404

DBSecurityGroupNotFound

DBSecurityGroupName doesn't refer to an existing security group.

HTTP Status Code: 404

DBUpgradeDependencyFailure

The upgrade failed because a resource that depends on can't be modified.

HTTP Status Code: 400

InsufficientDBInstanceStateCapacity

The specified instance class isn't available in the specified Availability Zone.

HTTP Status Code: 400

InvalidDBInstanceState

The specified instance isn't in the *available* state.

HTTP Status Code: 400

InvalidDBSecurityGroupState

The state of the security group doesn't allow deletion.

HTTP Status Code: 400

InvalidVPCNetworkStateFault

The subnet group doesn't cover all Availability Zones after it is created because of changes that were made.

HTTP Status Code: 400

StorageQuotaExceeded

The request would cause you to exceed the allowed amount of storage available across all instances.

HTTP Status Code: 400

StorageTypeNotSupported

Storage of the specified StorageType can't be associated with the DB instance.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ModifyDBSubnetGroup

Modifies an existing subnet group. subnet groups must contain at least one subnet in at least two Availability Zones in the AWS Region.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBSubnetGroupName

The name for the subnet group. This value is stored as a lowercase string. You can't modify the default subnet group.

Constraints: Must match the name of an existing DBSubnetGroup. Must not be default.

Example: mySubnetgroup

Type: String

Required: Yes

SubnetIds.SubnetIdentifier.N

The Amazon EC2 subnet IDs for the subnet group.

Type: Array of strings

Required: Yes

DBSubnetGroupDescription

The description for the subnet group.

Type: String

Required: No

Response Elements

The following element is returned by the service.

DBSubnetGroup

Detailed information about a subnet group.

Type: [DBSubnetGroup \(p. 728\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBSubnetGroupDoesNotCoverEnoughAZs

Subnets in the subnet group should cover at least two Availability Zones unless there is only one Availability Zone.

HTTP Status Code: 400

DBSubnetGroupNotFoundFault

DBSubnetGroupName doesn't refer to an existing subnet group.

HTTP Status Code: 404

DBSubnetQuotaExceededFault

The request would cause you to exceed the allowed number of subnets in a subnet group.

HTTP Status Code: 400

InvalidSubnet

The requested subnet is not valid, or multiple subnets were requested that are not all in a common virtual private cloud (VPC).

HTTP Status Code: 400

SubnetAlreadyInUse

The subnet is already in use in the Availability Zone.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ModifyEventSubscription

Modifies an existing Amazon DocumentDB event notification subscription.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

SubscriptionName

The name of the Amazon DocumentDB event notification subscription.

Type: String

Required: Yes

Enabled

A Boolean value; set to `true` to activate the subscription.

Type: Boolean

Required: No

EventCategories.EventCategory.N

A list of event categories for a `SourceType` that you want to subscribe to.

Type: Array of strings

Required: No

SnsTopicArn

The Amazon Resource Name (ARN) of the SNS topic created for event notification. The ARN is created by Amazon SNS when you create a topic and subscribe to it.

Type: String

Required: No

SourceType

The type of source that is generating the events. For example, if you want to be notified of events generated by an instance, set this parameter to `db-instance`. If this value is not specified, all events are returned.

Valid values: `db-instance`, `db-parameter-group`, `db-security-group`

Type: String

Required: No

Response Elements

The following element is returned by the service.

EventSubscription

Detailed information about an event to which you have subscribed.

Type: [EventSubscription \(p. 735\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

EventSubscriptionQuotaExceeded

You have reached the maximum number of event subscriptions.

HTTP Status Code: 400

SNSInvalidTopic

Amazon SNS has responded that there is a problem with the specified topic.

HTTP Status Code: 400

SNSNoAuthorization

You do not have permission to publish to the SNS topic Amazon Resource Name (ARN).

HTTP Status Code: 400

SNSTopicArnNotFound

The SNS topic Amazon Resource Name (ARN) does not exist.

HTTP Status Code: 404

SubscriptionCategoryNotFound

The provided category does not exist.

HTTP Status Code: 404

SubscriptionNotFound

The subscription name does not exist.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ModifyGlobalCluster

Modify a setting for an Amazon DocumentDB global cluster. You can change one or more configuration parameters (for example: deletion protection), or the global cluster identifier by specifying these parameters and the new values in the request.

Note

This action only applies to Amazon DocumentDB clusters.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

GlobalClusterIdentifier

The identifier for the global cluster being modified. This parameter isn't case-sensitive.

Constraints:

- Must match the identifier of an existing global cluster.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [A-Za-z][0-9A-Za-z-:_]*

Required: Yes

DeletionProtection

Indicates if the global cluster has deletion protection enabled. The global cluster can't be deleted when deletion protection is enabled.

Type: Boolean

Required: No

NewGlobalClusterIdentifier

The new identifier for a global cluster when you modify a global cluster. This value is stored as a lowercase string.

- Must contain from 1 to 63 letters, numbers, or hyphens

The first character must be a letter

Can't end with a hyphen or contain two consecutive hyphens

Example: my-cluster2

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [A-Za-z][0-9A-Za-z-:_]*

Required: No

Response Elements

The following element is returned by the service.

GlobalCluster

A data type representing an Amazon DocumentDB global cluster.

Type: [GlobalCluster \(p. 738\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

GlobalClusterNotFoundFault

The `GlobalClusterIdentifier` doesn't refer to an existing global cluster.

HTTP Status Code: 404

InvalidGlobalClusterStateFault

The requested operation can't be performed while the cluster is in this state.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

RebootDBInstance

You might need to reboot your instance, usually for maintenance reasons. For example, if you make certain changes, or if you change the cluster parameter group that is associated with the instance, you must reboot the instance for the changes to take effect.

Rebooting an instance restarts the database engine service. Rebooting an instance results in a momentary outage, during which the instance status is set to *rebooting*.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBInstanceIdentifier

The instance identifier. This parameter is stored as a lowercase string.

Constraints:

- Must match the identifier of an existing DBInstance.

Type: String

Required: Yes

ForceFailover

When true, the reboot is conducted through a Multi-AZ failover.

Constraint: You can't specify true if the instance is not configured for Multi-AZ.

Type: Boolean

Required: No

Response Elements

The following element is returned by the service.

DBInstance

Detailed information about an instance.

Type: [DBInstance \(p. 723\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBInstanceNotFound

`DBInstanceIdentifier` doesn't refer to an existing instance.

HTTP Status Code: 404

InvalidDBInstanceState

The specified instance isn't in the *available* state.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

RemoveFromGlobalCluster

Detaches an Amazon DocumentDB secondary cluster from a global cluster. The cluster becomes a standalone cluster with read-write capability instead of being read-only and receiving data from a primary in a different region.

Note

This action only applies to Amazon DocumentDB clusters.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DbClusterIdentifier

The Amazon Resource Name (ARN) identifying the cluster that was detached from the Amazon DocumentDB global cluster.

Type: String

Required: Yes

GlobalClusterIdentifier

The cluster identifier to detach from the Amazon DocumentDB global cluster.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [A-Za-z][0-9A-Za-z-:_]*

Required: Yes

Response Elements

The following element is returned by the service.

GlobalCluster

A data type representing an Amazon DocumentDB global cluster.

Type: [GlobalCluster \(p. 738\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBClusterNotFoundFault

`DBClusterIdentifier` doesn't refer to an existing cluster.

HTTP Status Code: 404

GlobalClusterNotFoundFault

The `GlobalClusterIdentifier` doesn't refer to an existing global cluster.

HTTP Status Code: 404

InvalidGlobalClusterStateFault

The requested operation can't be performed while the cluster is in this state.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

RemoveSourceIdentifierFromSubscription

Removes a source identifier from an existing Amazon DocumentDB event notification subscription.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

SourceIdentifier

The source identifier to be removed from the subscription, such as the instance identifier for an instance, or the name of a security group.

Type: String

Required: Yes

SubscriptionName

The name of the Amazon DocumentDB event notification subscription that you want to remove a source identifier from.

Type: String

Required: Yes

Response Elements

The following element is returned by the service.

EventSubscription

Detailed information about an event to which you have subscribed.

Type: [EventSubscription \(p. 735\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

SourceNotFound

The requested source could not be found.

HTTP Status Code: 404

SubscriptionNotFound

The subscription name does not exist.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

RemoveTagsFromResource

Removes metadata tags from an Amazon DocumentDB resource.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

ResourceName

The Amazon DocumentDB resource that the tags are removed from. This value is an Amazon Resource Name (ARN).

Type: String

Required: Yes

TagKeys.member.N

The tag key (name) of the tag to be removed.

Type: Array of strings

Required: Yes

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBClusterNotFoundFault

`DBClusterIdentifier` doesn't refer to an existing cluster.

HTTP Status Code: 404

DBInstanceNotFound

`DBInstanceIdentifier` doesn't refer to an existing instance.

HTTP Status Code: 404

DBSnapshotNotFound

`DBSnapshotIdentifier` doesn't refer to an existing snapshot.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)

- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

ResetDBClusterParameterGroup

Modifies the parameters of a cluster parameter group to the default value. To reset specific parameters, submit a list of the following: ParameterName and ApplyMethod. To reset the entire cluster parameter group, specify the DBClusterParameterGroupName and ResetAllParameters parameters.

When you reset the entire group, dynamic parameters are updated immediately and static parameters are set to pending-reboot to take effect on the next DB instance reboot.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBClusterParameterGroupName

The name of the cluster parameter group to reset.

Type: String

Required: Yes

Parameters.Parameter.N

A list of parameter names in the cluster parameter group to reset to the default values. You can't use this parameter if the ResetAllParameters parameter is set to true.

Type: Array of [Parameter \(p. 743\)](#) objects

Required: No

ResetAllParameters

A value that is set to true to reset all parameters in the cluster parameter group to their default values, and false otherwise. You can't use this parameter if there is a list of parameter names specified for the Parameters parameter.

Type: Boolean

Required: No

Response Elements

The following element is returned by the service.

DBClusterParameterGroupName

The name of a cluster parameter group.

Constraints:

- Must be from 1 to 255 letters or numbers.
- The first character must be a letter.
- Cannot end with a hyphen or contain two consecutive hyphens.

Note

This value is stored as a lowercase string.

Type: String

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBParameterGroupNotFound

DBParameterGroupName doesn't refer to an existing parameter group.

HTTP Status Code: 404

InvalidDBParameterGroupState

The parameter group is in use, or it is in a state that is not valid. If you are trying to delete the parameter group, you can't delete it when the parameter group is in this state.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

RestoreDBClusterFromSnapshot

Creates a new cluster from a snapshot or cluster snapshot.

If a snapshot is specified, the target cluster is created from the source DB snapshot with a default configuration and default security group.

If a cluster snapshot is specified, the target cluster is created from the source cluster restore point with the same configuration as the original source DB cluster, except that the new cluster is created with the default security group.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBClusterIdentifier

The name of the cluster to create from the snapshot or cluster snapshot. This parameter isn't case sensitive.

Constraints:

- Must contain from 1 to 63 letters, numbers, or hyphens.
- The first character must be a letter.
- Cannot end with a hyphen or contain two consecutive hyphens.

Example: my-snapshot-id

Type: String

Required: Yes

Engine

The database engine to use for the new cluster.

Default: The same as source.

Constraint: Must be compatible with the engine of the source.

Type: String

Required: Yes

SnapshotIdentifier

The identifier for the snapshot or cluster snapshot to restore from.

You can use either the name or the Amazon Resource Name (ARN) to specify a cluster snapshot. However, you can use only the ARN to specify a snapshot.

Constraints:

- Must match the identifier of an existing snapshot.

Type: String

Required: Yes

AvailabilityZones.AvailabilityZone.N

Provides the list of Amazon EC2 Availability Zones that instances in the restored DB cluster can be created in.

Type: Array of strings

Required: No

DBSubnetGroupName

The name of the subnet group to use for the new cluster.

Constraints: If provided, must match the name of an existing DBSubnetGroup.

Example: mySubnetgroup

Type: String

Required: No

DeletionProtection

Specifies whether this cluster can be deleted. If DeletionProtection is enabled, the cluster cannot be deleted unless it is modified and DeletionProtection is disabled. DeletionProtection protects clusters from being accidentally deleted.

Type: Boolean

Required: No

EnableCloudwatchLogsExports.member.N

A list of log types that must be enabled for exporting to Amazon CloudWatch Logs.

Type: Array of strings

Required: No

EngineVersion

The version of the database engine to use for the new cluster.

Type: String

Required: No

KmsKeyId

The AWS KMS key identifier to use when restoring an encrypted cluster from a DB snapshot or cluster snapshot.

The AWS KMS key identifier is the Amazon Resource Name (ARN) for the AWS KMS encryption key. If you are restoring a cluster with the same AWS account that owns the AWS KMS encryption key used to encrypt the new cluster, then you can use the AWS KMS key alias instead of the ARN for the AWS KMS encryption key.

If you do not specify a value for the KmsKeyId parameter, then the following occurs:

- If the snapshot or cluster snapshot in SnapshotIdentifier is encrypted, then the restored cluster is encrypted using the AWS KMS key that was used to encrypt the snapshot or the cluster snapshot.
- If the snapshot or the cluster snapshot in SnapshotIdentifier is not encrypted, then the restored DB cluster is not encrypted.

Type: String

Required: No

Port

The port number on which the new cluster accepts connections.

Constraints: Must be a value from 1150 to 65535.

Default: The same port as the original cluster.

Type: Integer

Required: No

Tags.Tag.N

The tags to be assigned to the restored cluster.

Type: Array of [Tag \(p. 753\)](#) objects

Required: No

VpcSecurityGroupIds.VpcSecurityGroupId.N

A list of virtual private cloud (VPC) security groups that the new cluster will belong to.

Type: Array of strings

Required: No

Response Elements

The following element is returned by the service.

DBCluster

Detailed information about a cluster.

Type: [DBCluster \(p. 708\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBClusterAlreadyExistsFault

You already have a cluster with the given identifier.

HTTP Status Code: 400

DBClusterQuotaExceededFault

The cluster can't be created because you have reached the maximum allowed quota of clusters.

HTTP Status Code: 403

DBClusterSnapshotNotFoundFault

`DBClusterSnapshotIdentifier` doesn't refer to an existing cluster snapshot.

HTTP Status Code: 404

DBSnapshotNotFound

`DBSnapshotIdentifier` doesn't refer to an existing snapshot.

HTTP Status Code: 404

DBSubnetGroupNotFoundFault

DBSubnetGroupName doesn't refer to an existing subnet group.

HTTP Status Code: 404

DBSubnetGroupNotFoundFault

DBSubnetGroupName doesn't refer to an existing subnet group.

HTTP Status Code: 404

InsufficientDBClusterCapacityFault

The cluster doesn't have enough capacity for the current operation.

HTTP Status Code: 403

InsufficientStorageClusterCapacity

There is not enough storage available for the current action. You might be able to resolve this error by updating your subnet group to use different Availability Zones that have more storage available.

HTTP Status Code: 400

InvalidDBClusterSnapshotStateFault

The provided value isn't a valid cluster snapshot state.

HTTP Status Code: 400

InvalidDBSnapshotState

The state of the snapshot doesn't allow deletion.

HTTP Status Code: 400

InvalidRestoreFault

You cannot restore from a virtual private cloud (VPC) backup to a non-VPC DB instance.

HTTP Status Code: 400

InvalidSubnet

The requested subnet is not valid, or multiple subnets were requested that are not all in a common virtual private cloud (VPC).

HTTP Status Code: 400

InvalidVPCNetworkStateFault

The subnet group doesn't cover all Availability Zones after it is created because of changes that were made.

HTTP Status Code: 400

KMSKeyNotAccessibleFault

An error occurred when accessing an AWS KMS key.

HTTP Status Code: 400

StorageQuotaExceeded

The request would cause you to exceed the allowed amount of storage available across all instances.

HTTP Status Code: 400

StorageQuotaExceeded

The request would cause you to exceed the allowed amount of storage available across all instances.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

RestoreDBClusterToPointInTime

Restores a cluster to an arbitrary point in time. Users can restore to any point in time before LatestRestorableTime for up to BackupRetentionPeriod days. The target cluster is created from the source cluster with the same configuration as the original cluster, except that the new cluster is created with the default security group.

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBClusterIdentifier

The name of the new cluster to be created.

Constraints:

- Must contain from 1 to 63 letters, numbers, or hyphens.
- The first character must be a letter.
- Cannot end with a hyphen or contain two consecutive hyphens.

Type: String

Required: Yes

SourceDBClusterIdentifier

The identifier of the source cluster from which to restore.

Constraints:

- Must match the identifier of an existing DBCluster.

Type: String

Required: Yes

DBSubnetGroupName

The subnet group name to use for the new cluster.

Constraints: If provided, must match the name of an existing DBSubnetGroup.

Example: mySubnetgroup

Type: String

Required: No

DeletionProtection

Specifies whether this cluster can be deleted. If DeletionProtection is enabled, the cluster cannot be deleted unless it is modified and DeletionProtection is disabled. DeletionProtection protects clusters from being accidentally deleted.

Type: Boolean

Required: No

EnableCloudwatchLogsExports.member.N

A list of log types that must be enabled for exporting to Amazon CloudWatch Logs.

Type: Array of strings

Required: No

KmsKeyId

The AWS KMS key identifier to use when restoring an encrypted cluster from an encrypted cluster.

The AWS KMS key identifier is the Amazon Resource Name (ARN) for the AWS KMS encryption key. If you are restoring a cluster with the same AWS account that owns the AWS KMS encryption key used to encrypt the new cluster, then you can use the AWS KMS key alias instead of the ARN for the AWS KMS encryption key.

You can restore to a new cluster and encrypt the new cluster with an AWS KMS key that is different from the AWS KMS key used to encrypt the source cluster. The new DB cluster is encrypted with the AWS KMS key identified by the `KmsKeyId` parameter.

If you do not specify a value for the `KmsKeyId` parameter, then the following occurs:

- If the cluster is encrypted, then the restored cluster is encrypted using the AWS KMS key that was used to encrypt the source cluster.
- If the cluster is not encrypted, then the restored cluster is not encrypted.

If `DBClusterIdentifier` refers to a cluster that is not encrypted, then the restore request is rejected.

Type: String

Required: No

Port

The port number on which the new cluster accepts connections.

Constraints: Must be a value from 1150 to 65535.

Default: The default port for the engine.

Type: Integer

Required: No

RestoreToTime

The date and time to restore the cluster to.

Valid values: A time in Universal Coordinated Time (UTC) format.

Constraints:

- Must be before the latest restorable time for the instance.
- Must be specified if the `UseLatestRestorableTime` parameter is not provided.
- Cannot be specified if the `UseLatestRestorableTime` parameter is true.
- Cannot be specified if the `RestoreType` parameter is copy-on-write.

Example: 2015-03-07T23:45:00Z

Type: Timestamp

Required: No

RestoreType

The type of restore to be performed. You can specify one of the following values:

- **full-copy** - The new DB cluster is restored as a full copy of the source DB cluster.
- **copy-on-write** - The new DB cluster is restored as a clone of the source DB cluster.

Constraints: You can't specify **copy-on-write** if the engine version of the source DB cluster is earlier than 1.11.

If you don't specify a **RestoreType** value, then the new DB cluster is restored as a full copy of the source DB cluster.

Type: String

Required: No

Tags.Tag.N

The tags to be assigned to the restored cluster.

Type: Array of [Tag \(p. 753\)](#) objects

Required: No

UseLatestRestorableTime

A value that is set to `true` to restore the cluster to the latest restorable backup time, and `false` otherwise.

Default: `false`

Constraints: Cannot be specified if the **RestoreToTime** parameter is provided.

Type: Boolean

Required: No

VpcSecurityGroupIds.VpcSecurityGroupId.N

A list of VPC security groups that the new cluster belongs to.

Type: Array of strings

Required: No

Response Elements

The following element is returned by the service.

DBCluster

Detailed information about a cluster.

Type: [DBCluster \(p. 708\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBClusterAlreadyExistsFault

You already have a cluster with the given identifier.

HTTP Status Code: 400

DBClusterNotFoundFault

`DBClusterIdentifier` doesn't refer to an existing cluster.

HTTP Status Code: 404

DBClusterQuotaExceededFault

The cluster can't be created because you have reached the maximum allowed quota of clusters.

HTTP Status Code: 403

DBClusterSnapshotNotFoundFault

`DBClusterSnapshotIdentifier` doesn't refer to an existing cluster snapshot.

HTTP Status Code: 404

DBSubnetGroupNotFoundFault

`DBSubnetGroupName` doesn't refer to an existing subnet group.

HTTP Status Code: 404

InsufficientDBClusterCapacityFault

The cluster doesn't have enough capacity for the current operation.

HTTP Status Code: 403

InsufficientStorageClusterCapacity

There is not enough storage available for the current action. You might be able to resolve this error by updating your subnet group to use different Availability Zones that have more storage available.

HTTP Status Code: 400

InvalidDBClusterSnapshotStateFault

The provided value isn't a valid cluster snapshot state.

HTTP Status Code: 400

InvalidDBClusterStateFault

The cluster isn't in a valid state.

HTTP Status Code: 400

InvalidDBSnapshotState

The state of the snapshot doesn't allow deletion.

HTTP Status Code: 400

InvalidRestoreFault

You cannot restore from a virtual private cloud (VPC) backup to a non-VPC DB instance.

HTTP Status Code: 400

InvalidSubnet

The requested subnet is not valid, or multiple subnets were requested that are not all in a common virtual private cloud (VPC).

HTTP Status Code: 400

InvalidVPCNetworkStateFault

The subnet group doesn't cover all Availability Zones after it is created because of changes that were made.

HTTP Status Code: 400

KMSKeyNotAccessibleFault

An error occurred when accessing an AWS KMS key.

HTTP Status Code: 400

StorageQuotaExceeded

The request would cause you to exceed the allowed amount of storage available across all instances.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

StartDBCluster

Restarts the stopped cluster that is specified by `DBClusterIdentifier`. For more information, see [Stopping and Starting an Amazon DocumentDB Cluster](#).

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBClusterIdentifier

The identifier of the cluster to restart. Example: docdb-2019-05-28-15-24-52

Type: String

Required: Yes

Response Elements

The following element is returned by the service.

DBCluster

Detailed information about a cluster.

Type: [DBCluster \(p. 708\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBClusterNotFoundFault

`DBClusterIdentifier` doesn't refer to an existing cluster.

HTTP Status Code: 404

InvalidDBClusterStateFault

The cluster isn't in a valid state.

HTTP Status Code: 400

InvalidDBInstanceState

The specified instance isn't in the *available* state.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

StopDBCluster

Stops the running cluster that is specified by `DBClusterIdentifier`. The cluster must be in the *available* state. For more information, see [Stopping and Starting an Amazon DocumentDB Cluster](#).

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters \(p. 757\)](#).

DBClusterIdentifier

The identifier of the cluster to stop. Example: docdb-2019-05-28-15-24-52

Type: String

Required: Yes

Response Elements

The following element is returned by the service.

DBCluster

Detailed information about a cluster.

Type: [DBCluster \(p. 708\)](#) object

Errors

For information about the errors that are common to all actions, see [Common Errors \(p. 755\)](#).

DBClusterNotFoundFault

`DBClusterIdentifier` doesn't refer to an existing cluster.

HTTP Status Code: 404

InvalidDBClusterStateFault

The cluster isn't in a valid state.

HTTP Status Code: 400

InvalidDBInstanceState

The specified instance isn't in the *available* state.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Data Types

The following data types are supported:

- AvailabilityZone (p. 704)
- Certificate (p. 705)
- CloudwatchLogsExportConfiguration (p. 707)
- DBCluster (p. 708)
- DBClusterMember (p. 713)
- DBClusterParameterGroup (p. 714)
- DBClusterRole (p. 715)
- DBClusterSnapshot (p. 716)
- DBClusterSnapshotAttribute (p. 719)
- DBClusterSnapshotAttributesResult (p. 720)
- DBEngineVersion (p. 721)
- DBInstance (p. 723)
- DBInstanceStateInfo (p. 727)
- DBSubnetGroup (p. 728)
- Endpoint (p. 730)
- EngineDefaults (p. 731)
- Event (p. 732)
- EventCategoriesMap (p. 734)
- EventSubscription (p. 735)
- Filter (p. 737)
- GlobalCluster (p. 738)
- GlobalClusterMember (p. 740)
- OrderableDBInstanceState (p. 741)
- Parameter (p. 743)
- PendingCloudwatchLogsExports (p. 745)
- PendingMaintenanceAction (p. 746)
- PendingModifiedValues (p. 748)
- ResourcePendingMaintenanceActions (p. 751)
- Subnet (p. 752)
- Tag (p. 753)
- UpgradeTarget (p. 754)
- VpcSecurityGroupMembership (p. 755)

AvailabilityZone

Information about an Availability Zone.

Contents

Note

In the following list, the required parameters are described first.

Name

The name of the Availability Zone.

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

Certificate

A certificate authority (CA) certificate for an AWS account.

Contents

Note

In the following list, the required parameters are described first.

CertificateArn

The Amazon Resource Name (ARN) for the certificate.

Example: arn:aws:rds:us-east-1::cert:rds-ca-2019

Type: String

Required: No

CertificateIdentifier

The unique key that identifies a certificate.

Example: rds-ca-2019

Type: String

Required: No

CertificateType

The type of the certificate.

Example: CA

Type: String

Required: No

Thumbprint

The thumbprint of the certificate.

Type: String

Required: No

ValidFrom

The starting date-time from which the certificate is valid.

Example: 2019-07-31T17:57:09Z

Type: Timestamp

Required: No

ValidTill

The date-time after which the certificate is no longer valid.

Example: 2024-07-31T17:57:09Z

Type: Timestamp

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

CloudwatchLogsExportConfiguration

The configuration setting for the log types to be enabled for export to Amazon CloudWatch Logs for a specific instance or cluster.

The `EnableLogTypes` and `DisableLogTypes` arrays determine which logs are exported (or not exported) to CloudWatch Logs. The values within these arrays depend on the engine that is being used.

Contents

Note

In the following list, the required parameters are described first.

DisableLogTypes.member.N

The list of log types to disable.

Type: Array of strings

Required: No

EnableLogTypes.member.N

The list of log types to enable.

Type: Array of strings

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DBCluster

Detailed information about a cluster.

Contents

Note

In the following list, the required parameters are described first.

AssociatedRoles.DBClusterRole.N

Provides a list of the AWS Identity and Access Management (IAM) roles that are associated with the cluster. (IAM) roles that are associated with a cluster grant permission for the cluster to access other AWS services on your behalf.

Type: Array of [DBClusterRole \(p. 715\)](#) objects

Required: No

AvailabilityZones.AvailabilityZone.N

Provides the list of Amazon EC2 Availability Zones that instances in the cluster can be created in.

Type: Array of strings

Required: No

BackupRetentionPeriod

Specifies the number of days for which automatic snapshots are retained.

Type: Integer

Required: No

CloneGroupId

Identifies the clone group to which the DB cluster is associated.

Type: String

Required: No

ClusterCreateTime

Specifies the time when the cluster was created, in Universal Coordinated Time (UTC).

Type: Timestamp

Required: No

DBClusterArn

The Amazon Resource Name (ARN) for the cluster.

Type: String

Required: No

DBClusterIdentifier

Contains a user-supplied cluster identifier. This identifier is the unique key that identifies a cluster.

Type: String

Required: No

DBClusterMembers.DBClusterMember.N

Provides the list of instances that make up the cluster.

Type: Array of [DBClusterMember \(p. 713\)](#) objects

Required: No

DBClusterParameterGroup

Specifies the name of the cluster parameter group for the cluster.

Type: String

Required: No

DbClusterResourceId

The AWS Region-unique, immutable identifier for the cluster. This identifier is found in AWS CloudTrail log entries whenever the AWS KMS key for the cluster is accessed.

Type: String

Required: No

DBSubnetGroup

Specifies information on the subnet group that is associated with the cluster, including the name, description, and subnets in the subnet group.

Type: String

Required: No

DeletionProtection

Specifies whether this cluster can be deleted. If DeletionProtection is enabled, the cluster cannot be deleted unless it is modified and DeletionProtection is disabled. DeletionProtection protects clusters from being accidentally deleted.

Type: Boolean

Required: No

EarliestRestorableTime

The earliest time to which a database can be restored with point-in-time restore.

Type: Timestamp

Required: No

EnabledCloudwatchLogsExports.member.N

A list of log types that this cluster is configured to export to Amazon CloudWatch Logs.

Type: Array of strings

Required: No

Endpoint

Specifies the connection endpoint for the primary instance of the cluster.

Type: String

Required: No

Engine

Provides the name of the database engine to be used for this cluster.

Type: String

Required: No

EngineVersion

Indicates the database engine version.

Type: String

Required: No

HostedZoneId

Specifies the ID that Amazon Route 53 assigns when you create a hosted zone.

Type: String

Required: No

KmsKeyId

If StorageEncrypted is true, the AWS KMS key identifier for the encrypted cluster.

Type: String

Required: No

LatestRestorableTime

Specifies the latest time to which a database can be restored with point-in-time restore.

Type: Timestamp

Required: No

MasterUsername

Contains the master user name for the cluster.

Type: String

Required: No

MultiAZ

Specifies whether the cluster has instances in multiple Availability Zones.

Type: Boolean

Required: No

PercentProgress

Specifies the progress of the operation as a percentage.

Type: String

Required: No

Port

Specifies the port that the database engine is listening on.

Type: Integer

Required: No

PreferredBackupWindow

Specifies the daily time range during which automated backups are created if automated backups are enabled, as determined by the BackupRetentionPeriod.

Type: String

Required: No

PreferredMaintenanceWindow

Specifies the weekly time range during which system maintenance can occur, in Universal Coordinated Time (UTC).

Type: String

Required: No

ReaderEndpoint

The reader endpoint for the cluster. The reader endpoint for a cluster load balances connections across the Amazon DocumentDB replicas that are available in a cluster. As clients request new connections to the reader endpoint, Amazon DocumentDB distributes the connection requests among the Amazon DocumentDB replicas in the cluster. This functionality can help balance your read workload across multiple Amazon DocumentDB replicas in your cluster.

If a failover occurs, and the Amazon DocumentDB replica that you are connected to is promoted to be the primary instance, your connection is dropped. To continue sending your read workload to other Amazon DocumentDB replicas in the cluster, you can then reconnect to the reader endpoint.

Type: String

Required: No

ReadReplicaIdentifiers.ReadReplicaIdentifier.N

Contains one or more identifiers of the secondary clusters that are associated with this cluster.

Type: Array of strings

Required: No

ReplicationSourceIdentifier

Contains the identifier of the source cluster if this cluster is a secondary cluster.

Type: String

Required: No

Status

Specifies the current state of this cluster.

Type: String

Required: No

StorageEncrypted

Specifies whether the cluster is encrypted.

Type: Boolean

Required: No

VpcSecurityGroups.VpcSecurityGroupMembership.N

Provides a list of virtual private cloud (VPC) security groups that the cluster belongs to.

Type: Array of [VpcSecurityGroupMembership \(p. 755\)](#) objects

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DBClusterMember

Contains information about an instance that is part of a cluster.

Contents

Note

In the following list, the required parameters are described first.

DBClusterParameterGroupStatus

Specifies the status of the cluster parameter group for this member of the DB cluster.

Type: String

Required: No

DBInstanceIdentifier

Specifies the instance identifier for this member of the cluster.

Type: String

Required: No

IsClusterWriter

A value that is `true` if the cluster member is the primary instance for the cluster and `false` otherwise.

Type: Boolean

Required: No

PromotionTier

A value that specifies the order in which an Amazon DocumentDB replica is promoted to the primary instance after a failure of the existing primary instance.

Type: Integer

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DBClusterParameterGroup

Detailed information about a cluster parameter group.

Contents

Note

In the following list, the required parameters are described first.

DBClusterParameterGroupArn

The Amazon Resource Name (ARN) for the cluster parameter group.

Type: String

Required: No

DBClusterParameterGroupName

Provides the name of the cluster parameter group.

Type: String

Required: No

DBParameterGroupFamily

Provides the name of the parameter group family that this cluster parameter group is compatible with.

Type: String

Required: No

Description

Provides the customer-specified description for this cluster parameter group.

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DBClusterRole

Describes an AWS Identity and Access Management (IAM) role that is associated with a cluster.

Contents

Note

In the following list, the required parameters are described first.

RoleArn

The Amazon Resource Name (ARN) of the IAMrole that is associated with the DB cluster.

Type: String

Required: No

Status

Describes the state of association between the IAMrole and the cluster. The Status property returns one of the following values:

- ACTIVE - The IAMrole ARN is associated with the cluster and can be used to access other AWS services on your behalf.
- PENDING - The IAMrole ARN is being associated with the cluster.
- INVALID - The IAMrole ARN is associated with the cluster, but the cluster cannot assume the IAMrole to access other AWS services on your behalf.

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DBClusterSnapshot

Detailed information about a cluster snapshot.

Contents

Note

In the following list, the required parameters are described first.

AvailabilityZones.AvailabilityZone.N

Provides the list of Amazon EC2 Availability Zones that instances in the cluster snapshot can be restored in.

Type: Array of strings

Required: No

ClusterCreateTime

Specifies the time when the cluster was created, in Universal Coordinated Time (UTC).

Type: Timestamp

Required: No

DBClusterIdentifier

Specifies the cluster identifier of the cluster that this cluster snapshot was created from.

Type: String

Required: No

DBClusterSnapshotArn

The Amazon Resource Name (ARN) for the cluster snapshot.

Type: String

Required: No

DBClusterSnapshotIdentifier

Specifies the identifier for the cluster snapshot.

Type: String

Required: No

Engine

Specifies the name of the database engine.

Type: String

Required: No

EngineVersion

Provides the version of the database engine for this cluster snapshot.

Type: String

Required: No

KmsKeyId

If StorageEncrypted is true, the AWS KMS key identifier for the encrypted cluster snapshot.

Type: String

Required: No

MasterUsername

Provides the master user name for the cluster snapshot.

Type: String

Required: No

PercentProgress

Specifies the percentage of the estimated data that has been transferred.

Type: Integer

Required: No

Port

Specifies the port that the cluster was listening on at the time of the snapshot.

Type: Integer

Required: No

SnapshotCreateTime

Provides the time when the snapshot was taken, in UTC.

Type: Timestamp

Required: No

SnapshotType

Provides the type of the cluster snapshot.

Type: String

Required: No

SourceDBClusterSnapshotArn

If the cluster snapshot was copied from a source cluster snapshot, the ARN for the source cluster snapshot; otherwise, a null value.

Type: String

Required: No

Status

Specifies the status of this cluster snapshot.

Type: String

Required: No

StorageEncrypted

Specifies whether the cluster snapshot is encrypted.

Type: Boolean

Required: No

VpcId

Provides the virtual private cloud (VPC) ID that is associated with the cluster snapshot.

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DBClusterSnapshotAttribute

Contains the name and values of a manual cluster snapshot attribute.

Manual cluster snapshot attributes are used to authorize other AWS accounts to restore a manual cluster snapshot.

Contents

Note

In the following list, the required parameters are described first.

AttributeName

The name of the manual cluster snapshot attribute.

The attribute named `restore` refers to the list of AWS accounts that have permission to copy or restore the manual cluster snapshot.

Type: String

Required: No

AttributeValues.AttributeValue.N

The values for the manual cluster snapshot attribute.

If the `AttributeName` field is set to `restore`, then this element returns a list of IDs of the AWS accounts that are authorized to copy or restore the manual cluster snapshot. If a value of `all` is in the list, then the manual cluster snapshot is public and available for any AWS account to copy or restore.

Type: Array of strings

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DBClusterSnapshotAttributesResult

Detailed information about the attributes that are associated with a cluster snapshot.

Contents

Note

In the following list, the required parameters are described first.

DBClusterSnapshotAttributes.DBClusterSnapshotAttribute.N

The list of attributes and values for the cluster snapshot.

Type: Array of [DBClusterSnapshotAttribute \(p. 719\)](#) objects

Required: No

DBClusterSnapshotIdentifier

The identifier of the cluster snapshot that the attributes apply to.

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DBEngineVersion

Detailed information about an engine version.

Contents

Note

In the following list, the required parameters are described first.

DBEngineDescription

The description of the database engine.

Type: String

Required: No

DBEngineVersionDescription

The description of the database engine version.

Type: String

Required: No

DBParameterGroupFamily

The name of the parameter group family for the database engine.

Type: String

Required: No

Engine

The name of the database engine.

Type: String

Required: No

EngineVersion

The version number of the database engine.

Type: String

Required: No

ExportableLogTypes.member.N

The types of logs that the database engine has available for export to Amazon CloudWatch Logs.

Type: Array of strings

Required: No

SupportsLogExportsToCloudwatchLogs

A value that indicates whether the engine version supports exporting the log types specified by ExportableLogTypes to CloudWatch Logs.

Type: Boolean

Required: No

ValidUpgradeTarget.UpgradeTarget.N

A list of engine versions that this database engine version can be upgraded to.

Type: Array of [UpgradeTarget \(p. 754\)](#) objects

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DBInstance

Detailed information about an instance.

Contents

Note

In the following list, the required parameters are described first.

AutoMinorVersionUpgrade

Does not apply. This parameter does not apply to Amazon DocumentDB. Amazon DocumentDB does not perform minor version upgrades regardless of the value set.

Type: Boolean

Required: No

AvailabilityZone

Specifies the name of the Availability Zone that the instance is located in.

Type: String

Required: No

BackupRetentionPeriod

Specifies the number of days for which automatic snapshots are retained.

Type: Integer

Required: No

CACertificateIdentifier

The identifier of the CA certificate for this DB instance.

Type: String

Required: No

CopyTagsToSnapshot

A value that indicates whether to copy tags from the DB instance to snapshots of the DB instance. By default, tags are not copied.

Type: Boolean

Required: No

DBClusterIdentifier

Contains the name of the cluster that the instance is a member of if the instance is a member of a cluster.

Type: String

Required: No

DBInstanceArn

The Amazon Resource Name (ARN) for the instance.

Type: String

Required: No

DBInstanceClass

Contains the name of the compute and memory capacity class of the instance.

Type: String

Required: No

DBInstanceIdentifier

Contains a user-provided database identifier. This identifier is the unique key that identifies an instance.

Type: String

Required: No

DBInstanceState

Specifies the current state of this database.

Type: String

Required: No

DbiResourceId

The AWS Region-unique, immutable identifier for the instance. This identifier is found in AWS CloudTrail log entries whenever the AWS KMS key for the instance is accessed.

Type: String

Required: No

DBSubnetGroup

Specifies information on the subnet group that is associated with the instance, including the name, description, and subnets in the subnet group.

Type: [DBSubnetGroup \(p. 728\)](#) object

Required: No

EnabledCloudwatchLogsExports.member.N

A list of log types that this instance is configured to export to CloudWatch Logs.

Type: Array of strings

Required: No

Endpoint

Specifies the connection endpoint.

Type: [Endpoint \(p. 730\)](#) object

Required: No

Engine

Provides the name of the database engine to be used for this instance.

Type: String

Required: No

EngineVersion

Indicates the database engine version.

Type: String

Required: No

InstanceCreateTime

Provides the date and time that the instance was created.

Type: Timestamp

Required: No

KmsKeyId

If StorageEncrypted is true, the AWS KMS key identifier for the encrypted instance.

Type: String

Required: No

LatestRestorableTime

Specifies the latest time to which a database can be restored with point-in-time restore.

Type: Timestamp

Required: No

PendingModifiedValues

Specifies that changes to the instance are pending. This element is included only when changes are pending. Specific changes are identified by subelements.

Type: [PendingModifiedValues \(p. 748\)](#) object

Required: No

PreferredBackupWindow

Specifies the daily time range during which automated backups are created if automated backups are enabled, as determined by the BackupRetentionPeriod.

Type: String

Required: No

PreferredMaintenanceWindow

Specifies the weekly time range during which system maintenance can occur, in Universal Coordinated Time (UTC).

Type: String

Required: No

PromotionTier

A value that specifies the order in which an Amazon DocumentDB replica is promoted to the primary instance after a failure of the existing primary instance.

Type: Integer

Required: No

PubliclyAccessible

Not supported. Amazon DocumentDB does not currently support public endpoints. The value of `PubliclyAccessible` is always `false`.

Type: Boolean

Required: No

StatusInfos.DBInstanceStatusInfo.N

The status of a read replica. If the instance is not a read replica, this is blank.

Type: Array of [DBInstanceStatusInfo \(p. 727\)](#) objects

Required: No

StorageEncrypted

Specifies whether or not the instance is encrypted.

Type: Boolean

Required: No

VpcSecurityGroups.VpcSecurityGroupMembership.N

Provides a list of VPC security group elements that the instance belongs to.

Type: Array of [VpcSecurityGroupMembership \(p. 755\)](#) objects

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DBInstanceStateInfo

Provides a list of status information for an instance.

Contents

Note

In the following list, the required parameters are described first.

Message

Details of the error if there is an error for the instance. If the instance is not in an error state, this value is blank.

Type: String

Required: No

Normal

A Boolean value that is true if the instance is operating normally, or false if the instance is in an error state.

Type: Boolean

Required: No

Status

Status of the instance. For a StatusType of read replica, the values can be replicating, error, stopped, or terminated.

Type: String

Required: No

StatusType

This value is currently "read replication."

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

DBSubnetGroup

Detailed information about a subnet group.

Contents

Note

In the following list, the required parameters are described first.

DBSubnetGroupArn

The Amazon Resource Name (ARN) for the DB subnet group.

Type: String

Required: No

DBSubnetGroupDescription

Provides the description of the subnet group.

Type: String

Required: No

DBSubnetGroupName

The name of the subnet group.

Type: String

Required: No

SubnetGroupStatus

Provides the status of the subnet group.

Type: String

Required: No

Subnets.Subnet.N

Detailed information about one or more subnets within a subnet group.

Type: Array of [Subnet \(p. 752\)](#) objects

Required: No

VpcId

Provides the virtual private cloud (VPC) ID of the subnet group.

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)

- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

Endpoint

Network information for accessing a cluster or instance. Client programs must specify a valid endpoint to access these Amazon DocumentDB resources.

Contents

Note

In the following list, the required parameters are described first.

Address

Specifies the DNS address of the instance.

Type: String

Required: No

HostedZoneId

Specifies the ID that Amazon Route 53 assigns when you create a hosted zone.

Type: String

Required: No

Port

Specifies the port that the database engine is listening on.

Type: Integer

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

EngineDefaults

Contains the result of a successful invocation of the `DescribeEngineDefaultClusterParameters` operation.

Contents

Note

In the following list, the required parameters are described first.

DBParameterGroupFamily

The name of the cluster parameter group family to return the engine parameter information for.

Type: String

Required: No

Marker

An optional pagination token provided by a previous request. If this parameter is specified, the response includes only records beyond the marker, up to the value specified by `MaxRecords`.

Type: String

Required: No

Parameters.Parameter.N

The parameters of a particular cluster parameter group family.

Type: Array of [Parameter \(p. 743\)](#) objects

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

Event

Detailed information about an event.

Contents

Note

In the following list, the required parameters are described first.

Date

Specifies the date and time of the event.

Type: Timestamp

Required: No

EventCategories.EventCategory.N

Specifies the category for the event.

Type: Array of strings

Required: No

Message

Provides the text of this event.

Type: String

Required: No

SourceArn

The Amazon Resource Name (ARN) for the event.

Type: String

Required: No

SourceIdentifier

Provides the identifier for the source of the event.

Type: String

Required: No

SourceType

Specifies the source type for this event.

Type: String

Valid Values: db-instance | db-parameter-group | db-security-group | db-snapshot | db-cluster | db-cluster-snapshot

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

EventCategoriesMap

An event source type, accompanied by one or more event category names.

Contents

Note

In the following list, the required parameters are described first.

EventCategories.EventCategory.N

The event categories for the specified source type.

Type: Array of strings

Required: No

SourceType

The source type that the returned categories belong to.

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

EventSubscription

Detailed information about an event to which you have subscribed.

Contents

Note

In the following list, the required parameters are described first.

CustomerAwsId

The AWS customer account that is associated with the Amazon DocumentDB event notification subscription.

Type: String

Required: No

CustSubscriptionId

The Amazon DocumentDB event notification subscription ID.

Type: String

Required: No

Enabled

A Boolean value indicating whether the subscription is enabled. A value of `true` indicates that the subscription is enabled.

Type: Boolean

Required: No

EventCategoriesList.EventCategory.N

A list of event categories for the Amazon DocumentDB event notification subscription.

Type: Array of strings

Required: No

EventSubscriptionArn

The Amazon Resource Name (ARN) for the event subscription.

Type: String

Required: No

SnsTopicArn

The topic ARN of the Amazon DocumentDB event notification subscription.

Type: String

Required: No

SourceldsList.Sourceld.N

A list of source IDs for the Amazon DocumentDB event notification subscription.

Type: Array of strings

Required: No

SourceType

The source type for the Amazon DocumentDB event notification subscription.

Type: String

Required: No

Status

The status of the Amazon DocumentDB event notification subscription.

Constraints:

Can be one of the following: `creating`, `modifying`, `deleting`, `active`, `no-permission`, `topic-not-exist`

The `no-permission` status indicates that Amazon DocumentDB no longer has permission to post to the SNS topic. The `topic-not-exist` status indicates that the topic was deleted after the subscription was created.

Type: String

Required: No

SubscriptionCreationTime

The time at which the Amazon DocumentDB event notification subscription was created.

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

Filter

A named set of filter values, used to return a more specific list of results. You can use a filter to match a set of resources by specific criteria, such as IDs.

Wildcards are not supported in filters.

Contents

Note

In the following list, the required parameters are described first.

Name

The name of the filter. Filter names are case sensitive.

Type: String

Required: Yes

Values.Value.N

One or more filter values. Filter values are case sensitive.

Type: Array of strings

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

GlobalCluster

A data type representing an Amazon DocumentDB global cluster.

Contents

Note

In the following list, the required parameters are described first.

DatabaseName

The default database name within the new global cluster.

Type: String

Required: No

DeletionProtection

The deletion protection setting for the new global cluster.

Type: Boolean

Required: No

Engine

The Amazon DocumentDB database engine used by the global cluster.

Type: String

Required: No

EngineVersion

Indicates the database engine version.

Type: String

Required: No

GlobalClusterArn

The Amazon Resource Name (ARN) for the global cluster.

Type: String

Required: No

GlobalClusterIdentifier

Contains a user-supplied global cluster identifier. This identifier is the unique key that identifies a global cluster.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [A-Za-z][0-9A-Za-z-:_]*

Required: No

GlobalClusterMembers.GlobalClusterMember.N

The list of cluster IDs for secondary clusters within the global cluster. Currently limited to one item.

Type: Array of [GlobalClusterMember \(p. 740\)](#) objects

Required: No

GlobalClusterResourceId

The AWS Region-unique, immutable identifier for the global database cluster. This identifier is found in AWS CloudTrail log entries whenever the AWS KMS customer master key (CMK) for the cluster is accessed.

Type: String

Required: No

Status

Specifies the current state of this global cluster.

Type: String

Required: No

StorageEncrypted

The storage encryption setting for the global cluster.

Type: Boolean

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

GlobalClusterMember

A data structure with information about any primary and secondary clusters associated with an Amazon DocumentDB global clusters.

Contents

Note

In the following list, the required parameters are described first.

DBClusterArn

The Amazon Resource Name (ARN) for each Amazon DocumentDB cluster.

Type: String

Required: No

IsWriter

Specifies whether the Amazon DocumentDB cluster is the primary cluster (that is, has read-write capability) for the Amazon DocumentDB global cluster with which it is associated.

Type: Boolean

Required: No

Readers.member.N

The Amazon Resource Name (ARN) for each read-only secondary cluster associated with the Aurora global cluster.

Type: Array of strings

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

OrderableDBInstanceOption

The options that are available for an instance.

Contents

Note

In the following list, the required parameters are described first.

AvailabilityZones.AvailabilityZone.N

A list of Availability Zones for an instance.

Type: Array of [AvailabilityZone \(p. 704\)](#) objects

Required: No

DBInstanceClass

The instance class for an instance.

Type: String

Required: No

Engine

The engine type of an instance.

Type: String

Required: No

EngineVersion

The engine version of an instance.

Type: String

Required: No

LicenseModel

The license model for an instance.

Type: String

Required: No

Vpc

Indicates whether an instance is in a virtual private cloud (VPC).

Type: Boolean

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)

- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

Parameter

Detailed information about an individual parameter.

Contents

Note

In the following list, the required parameters are described first.

AllowedValues

Specifies the valid range of values for the parameter.

Type: String

Required: No

ApplyMethod

Indicates when to apply parameter updates.

Type: String

Valid Values: immediate | pending-reboot

Required: No

ApplyType

Specifies the engine-specific parameters type.

Type: String

Required: No

DataType

Specifies the valid data type for the parameter.

Type: String

Required: No

Description

Provides a description of the parameter.

Type: String

Required: No

IsModifiable

Indicates whether (true) or not (false) the parameter can be modified. Some parameters have security or operational implications that prevent them from being changed.

Type: Boolean

Required: No

MinimumEngineVersion

The earliest engine version to which the parameter can apply.

Type: String

Required: No

ParameterName

Specifies the name of the parameter.

Type: String

Required: No

ParameterValue

Specifies the value of the parameter.

Type: String

Required: No

Source

Indicates the source of the parameter value.

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

PendingCloudwatchLogsExports

A list of the log types whose configuration is still pending. These log types are in the process of being activated or deactivated.

Contents

Note

In the following list, the required parameters are described first.

LogTypesToDisable.member.N

Log types that are in the process of being enabled. After they are enabled, these log types are exported to Amazon CloudWatch Logs.

Type: Array of strings

Required: No

LogTypesToEnable.member.N

Log types that are in the process of being deactivated. After they are deactivated, these log types aren't exported to CloudWatch Logs.

Type: Array of strings

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

PendingMaintenanceAction

Provides information about a pending maintenance action for a resource.

Contents

Note

In the following list, the required parameters are described first.

Action

The type of pending maintenance action that is available for the resource.

Type: String

Required: No

AutoAppliedAfterDate

The date of the maintenance window when the action is applied. The maintenance action is applied to the resource during its first maintenance window after this date. If this date is specified, any next-maintenance opt-in requests are ignored.

Type: Timestamp

Required: No

CurrentApplyDate

The effective date when the pending maintenance action is applied to the resource.

Type: Timestamp

Required: No

Description

A description providing more detail about the maintenance action.

Type: String

Required: No

ForcedApplyDate

The date when the maintenance action is automatically applied. The maintenance action is applied to the resource on this date regardless of the maintenance window for the resource. If this date is specified, any immediate opt-in requests are ignored.

Type: Timestamp

Required: No

OptInStatus

Indicates the type of opt-in request that has been received for the resource.

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

PendingModifiedValues

One or more modified settings for an instance. These modified settings have been requested, but haven't been applied yet.

Contents

Note

In the following list, the required parameters are described first.

AllocatedStorage

Contains the new AllocatedStorage size for the instance that will be applied or is currently being applied.

Type: Integer

Required: No

BackupRetentionPeriod

Specifies the pending number of days for which automated backups are retained.

Type: Integer

Required: No

CACertificateIdentifier

Specifies the identifier of the certificate authority (CA) certificate for the DB instance.

Type: String

Required: No

DBInstanceClass

Contains the new DBInstanceClass for the instance that will be applied or is currently being applied.

Type: String

Required: No

DBInstanceIdentifier

Contains the new DBInstanceIdentifier for the instance that will be applied or is currently being applied.

Type: String

Required: No

DBSubnetGroupName

The new subnet group for the instance.

Type: String

Required: No

EngineVersion

Indicates the database engine version.

Type: String

Required: No

Iops

Specifies the new Provisioned IOPS value for the instance that will be applied or is currently being applied.

Type: Integer

Required: No

LicenseModel

The license model for the instance.

Valid values: license-included, bring-your-own-license, general-public-license

Type: String

Required: No

MasterUserPassword

Contains the pending or currently in-progress change of the master credentials for the instance.

Type: String

Required: No

MultiAZ

Indicates that the Single-AZ instance is to change to a Multi-AZ deployment.

Type: Boolean

Required: No

PendingCloudwatchLogsExports

A list of the log types whose configuration is still pending. These log types are in the process of being activated or deactivated.

Type: [PendingCloudwatchLogsExports \(p. 745\)](#) object

Required: No

Port

Specifies the pending port for the instance.

Type: Integer

Required: No

StorageType

Specifies the storage type to be associated with the instance.

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

ResourcePendingMaintenanceActions

Represents the output of [ApplyPendingMaintenanceAction \(p. 576\)](#).

Contents

Note

In the following list, the required parameters are described first.

PendingMaintenanceActionDetails.PendingMaintenanceAction.N

A list that provides details about the pending maintenance actions for the resource.

Type: Array of [PendingMaintenanceAction \(p. 746\)](#) objects

Required: No

ResourceIdentifier

The Amazon Resource Name (ARN) of the resource that has pending maintenance actions.

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

Subnet

Detailed information about a subnet.

Contents

Note

In the following list, the required parameters are described first.

SubnetAvailabilityZone

Specifies the Availability Zone for the subnet.

Type: [AvailabilityZone \(p. 704\)](#) object

Required: No

SubnetIdentifier

Specifies the identifier of the subnet.

Type: String

Required: No

SubnetStatus

Specifies the status of the subnet.

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

Tag

Metadata assigned to an Amazon DocumentDB resource consisting of a key-value pair.

Contents

Note

In the following list, the required parameters are described first.

Key

The required name of the tag. The string value can be from 1 to 128 Unicode characters in length and can't be prefixed with "aws :" or "rds :". The string can contain only the set of Unicode letters, digits, white space, '_', '.', '/', '=', '+', '-' (Java regex: " $^(\backslash\p{L}\backslash\p{Z}\backslash\p{N}_.:/=+\backslash-)*$$ ").

Type: String

Required: No

Value

The optional value of the tag. The string value can be from 1 to 256 Unicode characters in length and can't be prefixed with "aws :" or "rds :". The string can contain only the set of Unicode letters, digits, white space, '_', '.', '/', '=', '+', '-' (Java regex: " $^(\backslash\p{L}\backslash\p{Z}\backslash\p{N}_.:/=+\backslash-)*$$ ").

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

UpgradeTarget

The version of the database engine that an instance can be upgraded to.

Contents

Note

In the following list, the required parameters are described first.

AutoUpgrade

A value that indicates whether the target version is applied to any source DB instances that have AutoMinorVersionUpgrade set to true.

Type: Boolean

Required: No

Description

The version of the database engine that an instance can be upgraded to.

Type: String

Required: No

Engine

The name of the upgrade target database engine.

Type: String

Required: No

EngineVersion

The version number of the upgrade target database engine.

Type: String

Required: No

IsMajorVersionUpgrade

A value that indicates whether a database engine is upgraded to a major version.

Type: Boolean

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

VpcSecurityGroupMembership

Used as a response element for queries on virtual private cloud (VPC) security group membership.

Contents

Note

In the following list, the required parameters are described first.

Status

The status of the VPC security group.

Type: String

Required: No

VpcSecurityGroupId

The name of the VPC security group.

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for Ruby V3](#)

Common Errors

This section lists the errors common to the API actions of all AWS services. For errors specific to an API action for this service, see the topic for that API action.

AccessDeniedException

You do not have sufficient access to perform this action.

HTTP Status Code: 400

IncompleteSignature

The request signature does not conform to AWS standards.

HTTP Status Code: 400

InternalFailure

The request processing has failed because of an unknown error, exception or failure.

HTTP Status Code: 500

InvalidAction

The action or operation requested is invalid. Verify that the action is typed correctly.

HTTP Status Code: 400

InvalidClientTokenId

The X.509 certificate or AWS access key ID provided does not exist in our records.

HTTP Status Code: 403

InvalidParameterCombination

Parameters that must not be used together were used together.

HTTP Status Code: 400

InvalidParameterValue

An invalid or out-of-range value was supplied for the input parameter.

HTTP Status Code: 400

InvalidQueryParameter

The AWS query string is malformed or does not adhere to AWS standards.

HTTP Status Code: 400

MalformedQueryString

The query string contains a syntax error.

HTTP Status Code: 404

MissingAction

The request is missing an action or a required parameter.

HTTP Status Code: 400

MissingAuthenticationToken

The request must contain either a valid (registered) AWS access key ID or X.509 certificate.

HTTP Status Code: 403

MissingParameter

A required parameter for the specified action is not supplied.

HTTP Status Code: 400

NotAuthorized

You do not have permission to perform this action.

HTTP Status Code: 400

OptInRequired

The AWS access key ID needs a subscription for the service.

HTTP Status Code: 403

RequestExpired

The request reached the service more than 15 minutes after the date stamp on the request or more than 15 minutes after the request expiration date (such as for pre-signed URLs), or the date stamp on the request is more than 15 minutes in the future.

HTTP Status Code: 400

ServiceUnavailable

The request has failed due to a temporary failure of the server.

HTTP Status Code: 503

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

ValidationException

The input fails to satisfy the constraints specified by an AWS service.

HTTP Status Code: 400

Common Parameters

The following list contains the parameters that all actions use for signing Signature Version 4 requests with a query string. Any action-specific parameters are listed in the topic for that action. For more information about Signature Version 4, see [Signature Version 4 Signing Process](#) in the *Amazon Web Services General Reference*.

Action

The action to be performed.

Type: string

Required: Yes

Version

The API version that the request is written for, expressed in the format YYYY-MM-DD.

Type: string

Required: Yes

X-Amz-Algorithm

The hash algorithm that you used to create the request signature.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Valid Values: AWS4-HMAC-SHA256

Required: Conditional

X-Amz-Credential

The credential scope value, which is a string that includes your access key, the date, the region you are targeting, the service you are requesting, and a termination string ("aws4_request"). The value is expressed in the following format: *access_key/YYYYMMDD/region/service/aws4_request*.

For more information, see [Task 2: Create a String to Sign for Signature Version 4](#) in the *Amazon Web Services General Reference*.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

X-Amz-Date

The date that is used to create the signature. The format must be ISO 8601 basic format (YYYYMMDD'T'HHMMSS'Z'). For example, the following date time is a valid X-Amz-Date value: 20120325T120000Z.

Condition: X-Amz-Date is optional for all requests; it can be used to override the date used for signing requests. If the Date header is specified in the ISO 8601 basic format, X-Amz-Date is not required. When X-Amz-Date is used, it always overrides the value of the Date header. For more information, see [Handling Dates in Signature Version 4](#) in the *Amazon Web Services General Reference*.

Type: string

Required: Conditional

X-Amz-Security-Token

The temporary security token that was obtained through a call to AWS Security Token Service (AWS STS). For a list of services that support temporary security credentials from AWS Security Token Service, go to [AWS Services That Work with IAM](#) in the *IAM User Guide*.

Condition: If you're using temporary security credentials from the AWS Security Token Service, you must include the security token.

Type: string

Required: Conditional

X-Amz-Signature

Specifies the hex-encoded signature that was calculated from the string to sign and the derived signing key.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

X-Amz-SignedHeaders

Specifies all the HTTP headers that were included as part of the canonical request. For more information about specifying signed headers, see [Task 1: Create a Canonical Request For Signature Version 4](#) in the *Amazon Web Services General Reference*.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

Release Notes

The release notes section describes the Amazon DocumentDB (with MongoDB compatibility) features, improvements, and bug fixes by release date. With the launch of MongoDB 4.0 compatibility on November 11, 2020, the release notes will include updates for both the Amazon DocumentDB 3.6 and 4.0 engine versions.

You can determine the current Amazon DocumentDB engine version by running the following command:

```
db.runCommand({getEngineVersion: 1})
```

If your cluster is not on the latest version of the engine, it is likely that you have pending maintenance available that will upgrade your engine. For more information, see [Maintaining Amazon DocumentDB \(p. 392\)](#).

Topics

- [January 21, 2022 \(p. 759\)](#)
- [October 25, 2021 \(p. 760\)](#)
- [June 24, 2021 \(p. 760\)](#)
- [May 4, 2021 \(p. 760\)](#)
- [January 15, 2021 \(p. 761\)](#)
- [November 9, 2020 \(p. 761\)](#)
- [October 30, 2020 \(p. 763\)](#)
- [September 22, 2020 \(p. 763\)](#)
- [July 10, 2020 \(p. 764\)](#)
- [June 30, 2020 \(p. 764\)](#)

January 21, 2022

New Features

Amazon DocumentDB 4.0 (Engine Version 2.0.5706)

- Amazon DocumentDB Graviton2 (r6g.large, r6g.2xlarge, r6g.4xlarge, r6g.8xlarge, r6g.12xlarge, r6g.16xlarge and t4g.medium) instances are now supported

Amazon DocumentDB 3.6 (Engine Version 1.0.207781) and 4.0 (Engine Version 2.0.5706)

- Added support for the following MongoDB APIs
 - \$reduce
 - \$mergeObjects
 - \$geoWithin
 - \$geoIntersects

October 25, 2021

New Features

Amazon DocumentDB 3.6 (Engine Version 1.0.207780) and 4.0 (Engine Version 2.0.5704)

- Added support for the following MongoDB APIs
 - \$literal
 - \$map
 - \$\$ROOT
- Support for GeoSpatial Query capabilities. See this [blog post](#) for more details
- Support for access control with user-defined roles. See this [blog post](#) for more details
- Amazon DocumentDB JDBC Driver to enable connectivity from BI tools such as Tableau and query tools such as SQL Workbench

Bug fixes and other changes

Amazon DocumentDB 3.6 (Engine Version 1.0.207780) and 4.0 (Engine Version 2.0.5704)

- Bug fix for \$natural to sort correctly when an explicit .sort() is present along with \$natural
- Bug fix for change stream to work with \$redact
- Bug fix for \$ifNull to work with empty array
- Bug fix for excessive resource consumption/server crash when a currently logged-in user is deleted or that user's privilege for an ongoing activity is revoked
- Bug fix in listDatabase and listCollection privilege check
- Bug Fix dedupe logic for multi-key elements

June 24, 2021

New Features

Amazon DocumentDB 3.6 (Engine Version 1.0.207117) and 4.0 (Engine Version 2.0.3371)

- r5.8xlarge and r5.16xlarge instances are now supported. Learn more at the blog post [Amazon DocumentDB Now Supports r5.8xlarge and r5.16xlarge Instances](#).
- [Global clusters](#) are now supported to provide disaster recovery from region-wide outages and enable low-latency global reads by allowing reads from the nearest Amazon DocumentDB cluster.

May 4, 2021

New Features

See all the new features in this [blog post](#).

Amazon DocumentDB 3.6 (Engine Version 1.0.207117) and 4.0 (Engine Version 2.0.3371)

- `renameCollection`
- `$zip`
- `$indexOfArray`
- `$reverseArray`
- `$natural`
- `$hint` support for update
- Index scan for `distinct`

Bug fixes and other changes

Amazon DocumentDB 3.6 (Engine Version 1.0.207117) and 4.0 (Engine Version 2.0.3371)

- Reduced memory usage for `$in` queries
- Fixed a memory leak in multikey indexes
- Fixed the explain plan and profiler output for `$out`
- Added a timeout for operations from internal monitoring system to improve reliability
- Fixed a defect impacting the query predicates passed to multikey indexes

January 15, 2021

New Features

Amazon DocumentDB 4.0 (Engine Version 2.0.722)

- None

Amazon DocumentDB 3.6 (Engine Version 1.0.206295)

- Ability to use an index with the `$lookup` aggregation stage
- `find()` queries with projections can be served direction from an index (covered query)
- Ability to use `hint()` with the `findAndModify`
- Performance optimizations for `$addToSet` operator
- Improvements to reduce overall index sizes
- New aggregation operators: `$ifNull`, `$replaceRoot`, `$setIsSubset`, `$setIntersection`, `$setUnion`, and `$setEquals`
- Users can also end their own cursors without requiring the `KillCursor` role

November 9, 2020

New Features

See all the new features in [this blog post](#).

Amazon DocumentDB 4.0 (Engine Version 2.0.722)

- MongoDB 4.0 compatibility

- ACID transactions
- Support for `cluster(client.watch())` or `mongo.watch()` and the database level (`db.watch()`) change streams
- Ability to start or resume a change streams using `startAtOperationTime`
- Extend your change stream retention period to 7 days (previously 24 hours)
- AWS DMS target for Amazon DocumentDB 4.0
- CloudWatch metrics: `TransactionsOpen`, `TransactionsOpenMax`, `TransactionsAborted`, `TransactionsStarted`, and `TransactionsCommitted`
- New fields for transactions in `currentOp`, `ServerStatus`, and `profiler`.
- Ability to use an index with the `$lookup` aggregation stage
- `find()` queries with projections can be served direction from an index (covered query)
- Ability to use `hint()` with the `findAndModify`
- Performance optimizations for `$addToSet` operator
- Improvements to reduce overall index sizes.
- New aggregation operators: `$ifNull`, `$replaceRoot`, `$setIsSubset`, `$setIntersection`, `$setUnion`, and `$setEquals`
- With the `ListCollection` and `ListDatabase` commands, you can now optionally use the `authorizedCollections` and `authorizedDatabases` parameters to allow users to list the collections and databases that they have permission to access without requiring the `listCollections` and `listDatabase` roles, respectively
- Users can also end their own cursors without requiring the `KillCursor` role
- Comparing numeric types of subdocuments is now consistent with comparing numeric types of first-level documents. The behavior in Amazon DocumentDB 4.0 is now compatible with MongoDB.

Amazon DocumentDB 3.6 (Engine Version 1.0.206295)

- None

Bug Fixes and Other Changes

Amazon DocumentDB 4.0 (Engine Version 2.0.722)

- `$setOnInsert` no longer allow updates when using the positional operator `$`. The behavior in Amazon DocumentDB 4.0 is now compatible with MongoDB.
- Fixed issue with `$createCollection` and set `autoIndexId`
- Projection for nested documents
- Changed default setting for working memory to scale with instance memory size
- Garbage collection improvements
- Lookup with empty key in path, behavior difference with mongo
- Fixed `dateToString` bug in timezone behavior
- Fixed `$push (aggregation)` to respect sort order
- Fixed bug in `$currentOp` with aggregate
- Fixed issue with `readPreference` on secondary
- Fixed issue with validating `$createIndex` is the same database as the command was issued
- Fixed inconsistent behavior for `minKey`, `maxKey` lookup fails
- Fixed issue with `$size` operator not working with composite array
- Fixed issue with the negation of `$in` with regex

- Fixed issue with `$distinct` command run against a view
- Fixed issue with aggregations and find commands sorting missing fields differently
- Fixed `$eq` to regular expression not checking type
- Fixed `$currentDate` bug in timestamp ordinal position behavior
- Fixed millisecond granularity for `$currentDate`

Amazon DocumentDB 3.6 (Engine Version 1.0.206295)

- None

October 30, 2020

New Features

See all the new features in [this blog post](#).

Amazon DocumentDB 3.6 (Engine Version 1.0.206295)

- Added the ability to open a change stream cursor at the cluster level (`client.watch()` or `mongo.watch()`) and the database (`db.watch()`)
- Ability to increase the change stream retention period to 7 days (previously 24 hours)

Bug Fixes and Other Changes

Amazon DocumentDB 3.6 (Engine Version 1.0.206295)

- Various general case performance improvements
- A targeted security improvement
- Fixed an issue with skip sort on second field of a compound index
- Enable regular index for equality on single field of a multi-key index (not compound)
- Fixed authentication race condition
- Fixed issue that caused an infrequent garbage collection crash
- RBAC security improvement
- Added `databaseConnectionsMax` metric
- Performance improvements for certain workloads on `r5.24xlarge` instances

September 22, 2020

New Features

See all the new features in [this blog post](#).

Amazon DocumentDB 3.6 (Engine Version 1.0.206295)

- `$out` aggregation stage
- Increased the maximum number of connections and cursor per instance by as much as 10x

Bug Fixes and Other Changes

Amazon DocumentDB 3.6 (Engine Version 1.0.206295)

- None

July 10, 2020

New Features

See all the new features in [this blog post](#).

Amazon DocumentDB 3.6 (Engine Version 1.0.206295)

- Cross Region Snapshot Copy

Bug Fixes and Other Changes

Amazon DocumentDB 3.6 (Engine Version 1.0.206295)

- None

June 30, 2020

New Features

See all the new features in [this blog post](#).

Amazon DocumentDB 3.6 (Engine Version 1.0.206295)

- T3 medium instances

Bug Fixes and Other Changes

Amazon DocumentDB 3.6 (Engine Version 1.0.206295)

- Idle memory reclamation for t3 instances
- Authentication improvements
- Improved SASL authentication performance
- Fixed currentOp issue when exceeding maximum possible ops
- Fixed kill10ps issue for bulk update and delete
- Improvements to \$sample performance with \$match
- Fixed support for \$\$ in cond case in redact stage
- Fixed various recurring crash root causes
- Improvements to TTL sweeping to reduce IOs and latency
- Optimized memory utilization for \$unwind
- Fixed collection stats race condition with drop index

- Fixed race condition during concurrent index build
- Fixed infrequent crash in hash_search in index

Document History for the Amazon DocumentDB Developer Guide

- **API version:** 2014-10-31
- **Latest documentation update:** November 11, 2020

The following table describes the documentation for this release of the *Amazon DocumentDB Developer Guide*.

Change	Description	Date
Global Clusters	Added documentation on how to use Global Clusters.	June 2, 2021
Event Subscriptions	Added event subscription documentation.	March 26, 2021
Version 3.6 Upgrades	Documented improvements to version 3.6 in role-based access controls, aggregation operators, and performance.	January 15, 2021
MongoDB 4.0 Compatibility	Amazon DocumentDB is now compatible with version 4.0 of MongoDB.	November 9, 2020
Get Started Guides	New Get Started Guides for getting started with Amazon DocumentDB using , Amazon EC2, Robo3T or Studio3T.	August 15, 2020
Additional Availability Zones supported	Amazon DocumentDB added support for an additional Availability Zone in Asia Pacific (Seoul) (ap-northeast-2).	July 14, 2020
Added support for copying snapshots across Regions.	Amazon DocumentDB added support for copying cluster snapshots across AWS Regions. For more information, see Copying Snapshots Across Regions .	July 10, 2020
Added support for T3 instance class.	Added support for T3 instance types in all Regions supporting Amazon DocumentDB. For more information, see Supported Instance Classes by Region and Instance Class Specifications .	June 30, 2020
Added support for AWS GovCloud (US).	Amazon DocumentDB is now available in the AWS GovCloud (US) Region (us-gov-west-1).	June 29, 2020

Added 16 new CloudWatch metrics.	Amazon DocumentDB added support for 16 new Amazon CloudWatch metrics. For more information, see Monitoring Amazon DocumentDB with CloudWatch .	June 23, 2020
Added support for null characters and \$regex operator.	Amazon DocumentDB added support for null characters in strings and the ability to use an index for \$regex. To view the supported MongoDB APIs and aggregation pipeline capabilities for Amazon DocumentDB, see Functional Differences with MongoDB .	June 22, 2020
Added support for improved multi-key indexing capabilities.	Amazon DocumentDB added support for improved multi-key indexing capabilities that include indexing of arrays larger than 2,048 bytes and the ability to create a compound multi-key index with multiple keys in the same array. For more information, see Functional Differences with MongoDB .	April 23, 2020
Added support for deletion protection for an Amazon DocumentDB AWS CloudFormation stack.	Amazon DocumentDB added support for enabling deletion protection when creating an Amazon DocumentDB AWS CloudFormation stack.	April 20, 2020
Added support for role-based access control.	Amazon DocumentDB added support for role-based access control using built-in roles.	March 26, 2020
Added support for an additional Availability Zone in Canada (Central) (ca-central-1).	Amazon DocumentDB is now available in the Canada (Central) Region (ca-central-1) with R5 class instances and 3 Availability Zones.	March 26, 2020
Added support for two additional MongoDB APIs.	Amazon DocumentDB added support for \$dateFromString and executionStats MongoDB APIs.	March 23, 2020
Added support for five additional MongoDB APIs.	Amazon DocumentDB added support for \$objectToArray, \$arrayToObject, \$slice, \$mod, and \$range MongoDB APIs.	February 6, 2020

Added support for Canada (Central).	Amazon DocumentDB is now available in the Canada (Central) Region (ca-central-1) with R5 class instances.	December 11, 2019
Added support for ChangeStreamLogSize.	Amazon DocumentDB added support for ChangeStreamLogSize for Cloudwatch metrics.	November 22, 2019
Added support for Europe (Paris) region	Amazon DocumentDB is now available in the Europe (Paris) region (eu-west-3) with R5 class instances.	October 30, 2019
Added support for Asia Pacific (Mumbai) region	Amazon DocumentDB is now available in the Asia Pacific (Mumbai) region (ap-south-1) with R5 class instances.	October 17, 2019
Added support for three additional MongoDB APIs	Amazon DocumentDB added support for the \$addFields, \$concatArrays, and \$lookup MongoDB APIs.	October 16, 2019
Added support for Asia Pacific (Singapore) region	Amazon DocumentDB is now available in the Asia Pacific (Singapore) region (ap-southeast-1) with R5 class instances.	October 14, 2019
Added new document for updating TLS certificates	Added instructions for updating CA certificates to use the new CA certificate to create TLS connections.	October 2, 2019
Added API support for certificates	Amazon DocumentDB a new Certificate data type for instances. For more information, see DBInstance .	October 1, 2019
Support for query profiling	Amazon DocumentDB added the ability to profile supported operations on your cluster's instances and databases.	August 19, 2019
Added third AZ in Asia Pacific (Tokyo)	Amazon DocumentDB added a third Availability Zone (AZ) for your compute instances in Asia Pacific (Tokyo).	August 9, 2019

Support for additional Mongo APIs	Added support for additional aggregation pipeline capabilities that include the \$in, \$isoWeek, \$isoWeekYear, \$isoDayOfWeek, and \$dateToString aggregation operators and the \$addToSet aggregation stage. Amazon DocumentDB also added support for the top() command for collection level diagnostics and the ability to modify the expireAfterSeconds parameter for TTL indexes using the collMod() command.	July 31, 2019
Added support for Europe (London)	Amazon DocumentDB is now available in Europe (London) (eu-west-2) with R5 class instances.	July 18, 2019
Added code samples	Added code examples in R and Ruby for programmatically connecting to Amazon DocumentDB.	July 17, 2019
Added best practice	Added a Best Practice to help you manage your Amazon DocumentDB costs.	July 17, 2019
Support for stopping and starting a cluster	Amazon DocumentDB added support for stopping and starting clusters to manage costs for development and test environments.	July 1, 2019
Support for cluster deletion protection	To protect your clusters from accidental deletion, Amazon DocumentDB added deletion protection. For more information, see the following topics: Creating an Amazon DocumentDB Cluster , Modifying an Amazon DocumentDB Cluster , Deleting an Amazon DocumentDB Cluster , and DeletionProtection in the API topic DBCluster .	July 1, 2019
Functional differences update	Added Implicit Transactions to Functional Differences.	June 26, 2019
Functional differences addition	Added note regarding storage and index compression in Amazon DocumentDB.	June 13, 2019

Additional region supported	Amazon DocumentDB is now available in Asia Pacific (Sydney) (ap-southeast-2) with R5 class instances.	June 5, 2019
R5 instance class supported in additional regions	Added R5 instance class support for 4 additional regions: US East (Ohio), US East (N. Virginia), US West (Oregon), and EU (Ireland). With this change, R5 instances are supported in all regions supporting Amazon DocumentDB.	May 17, 2019
Additional regions supported	Added support for 2 additional regions, Asia Pacific (Tokyo) (ap-northeast-1) and Asia Pacific (Seoul) (ap-northeast-2) with R5 instance classes. For more information, see Supported Instance Classes by Region and Instance Class Specifications .	May 8, 2019
Added more connection code examples	Added code examples in Java and C# for connecting to Amazon DocumentDB.	April 24, 2019
Additional Mongo API support	Added support for seven aggregation string operators (\$index0fBytes, \$index0fCP, \$strLenBytes, \$strLenCP, \$toLower, \$toUpper, and \$split), nine date-time operators (\$day0fYear, \$day0fMonth, \$day0fWeek, \$year, \$month, \$hour, \$minute, \$second, and \$millisecond), and the \$sample aggregation pipeline stage.	April 4, 2019
Added connection code examples	Added code examples in Python, Node.js, PHP, and Go for connecting to Amazon DocumentDB.	March 21, 2019
Support for Frankfurt Region and R5 instances	Added support for Europe (Frankfurt) Region (eu-central-1) with R5 instance classes. For more information, see Supported Instance Classes by Region and Instance Class Specifications .	March 13, 2019

Aggregation pipeline operators support	Added support for new aggregation string operators (<code>\$concat</code> , <code>\$substr</code> , <code>\$substrBytes</code> , <code>\$substrCP</code> , <code>\$strcasecmp</code>), an array aggregation operator (<code>\$size</code>), an aggregation group accumulator operator (<code>\$push</code>), and aggregation stages (<code>\$redact</code> and <code>\$indexStats</code>). We also added support for positional array operators (<code>\$[]</code> and <code>\$[<identifier>]</code>) and <code>hint()</code> .	February 28, 2019
Engine upgrades	Added documentation for determining pending cluster modifications and upgrading your cluster's engine version.	February 15, 2019
Auditing events	Added support for auditing database events with Amazon CloudWatch Logs.	February 12, 2019
Quick Start	Added a Quick Start topic to help you easily start with Amazon DocumentDB using AWS CloudFormation.	January 11, 2019
Public Release (p. 766)	This is the initial public release of Amazon DocumentDB (with MongoDB compatibility). This release includes the Developer Guide and the integrated Resource Management API Reference .	January 9, 2019