Amazon DocumentDB: Developer Guide
Copyright © 2020 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 .................................................................................. 3  
Regions ............................................................................................. 3  
Availability Zones ............................................................................... 4  
Pricing ................................................................................................. 5  
Monitoring .......................................................................................... 5  
Interfaces ........................................................................................... 5  
- AWS Management Console .............................................................. 5  
- AWS CLI ......................................................................................... 5  
- The mongo Shell .............................................................................. 6  
- MongoDB Drivers ......................................................................... 6  
What’s Next? ....................................................................................... 6  
How It Works ....................................................................................... 6  
- Amazon DocumentDB Endpoints ................................................... 7  
- TLS Support .................................................................................. 10  
- Amazon DocumentDB Storage ...................................................... 10  
- Amazon DocumentDB Replication ................................................. 10  
- Amazon DocumentDB Reliability .................................................. 10  
- Read Preference Options ............................................................... 11  
- TTL Deletes ................................................................................... 14  
What is a Document Database? .......................................................... 14  
- Use Cases ...................................................................................... 14  
- Understanding Documents ............................................................ 15  
- Working with Documents .............................................................. 19  
Quick Start using AWS CloudFormation .......................................... 28  
Prerequisites ....................................................................................... 28  
- Required IAM Permissions ............................................................ 28  
- Amazon EC2 Key Pair ................................................................. 29  
Termination Protection and Deletion Protection ............................... 30  
Launching the Amazon DocumentDB AWS CloudFormation Stack .......................... 30  
Accessing the Amazon DocumentDB Cluster .................................. 31  
Getting Started .................................................................................. 32  
Prerequisites ....................................................................................... 33  
Step 1: Create an Amazon DocumentDB Cluster ............................. 34  
  - Using the AWS Management Console ....................................... 34  
  - Using the AWS CLI ................................................................. 37  
Step 2: Launch an Amazon EC2 Instance ......................................... 40  
Step 3: Access and Use Your Cluster Using the mongo Shell .......... 42  
  Step 3.a: Install the mongo shell ................................................... 42  
  Step 3.b: Manage Amazon DocumentDB TLS ............................ 43  
  Step 3.c: Connect and Use Amazon DocumentDB ....................... 43  
Step 4: (Optional) Delete the Amazon DocumentDB Instance and Cluster .......................... 47  
Best Practices ....................................................................................... 49  
Basic Operational Guidelines ............................................................. 49  
Instance Sizing ................................................................................... 50  
Working with Indexes ....................................................................... 51  
  Building Indexes .......................................................................... 51  
  Impact of Indexes on Writing Data ............................................... 51  
  Identifying Missing Indexes ......................................................... 51  
  Identifying Unused Indexes ......................................................... 51  
Security Best Practices ..................................................................... 51
Cost Optimization ................................................................. 52
Using Metrics to Identify Performance Issues ............................. 52
Viewing Performance Metrics .................................................. 52
Setting a CloudWatch Alarm ................................................... 53
Evaluating Performance Metrics .............................................. 53
Tuning Queries ................................................................. 54
TTL and Timeseries Workloads .............................................. 54
Working with Cluster Parameter Groups ................................. 54
Aggregation Pipeline Queries ............................................... 55
batchInsert and batchUpdate .............................................. 55
Updating Certificates ......................................................... 56
Updating Your Application and Amazon DocumentDB Cluster ................................. 56
Step 1: Download the New CA Certificate and Update Your Application ................................. 56
Step 2: Update the Server Certificate ..................................... 57
Troubleshooting .................................................................. 58
Frequently Asked Questions ................................................... 58
What if I have questions or issues? ........................................ 58
How do I know whether I’m using TLS to connect to my Amazon DocumentDB cluster? .......... 58
Why are you updating the CA and server certificates? ............... 59
What happens if I don’t take any action by March 5, 2020? .......... 59
How do I know which of my Amazon DocumentDB instances are using the old/new server certificate? .......... 59
How do I modify individual instances in my Amazon DocumentDB cluster to update the server certificate? .......... 60
What happens if I add a new instance to an existing cluster? .......... 61
What happens if there is an instance replacement or failover on my cluster? ......................... 61
If I’m not using TLS to connect to my cluster, do I still need to update each of my instances? ....... 61
If I’m not using TLS to connect to my cluster but I plan to in the future, what should I do? .......... 61
Can the deadline be extended beyond March 5, 2020? ............... 61
How can I be sure that I’m using the newest CA bundle? .......... 61
Why do I see “RDS” in the name of the CA bundle? .................. 61
When will the new certificate expire? ..................................... 61
If I applied the new server certificate, can I revert it back to the old server certificate? .......... 63
If I restore from a snapshot or a point in time restore, will it have the new server certificate? ....... 63
What if I’m having issues connecting directly to my Amazon DocumentDB cluster from Mac OS X Catalina? .................................................. 64
Functional Differences with MongoDB .................................. 65
Admin Databases and Collections ........................................ 65
Array Indexing ............................................................. 65
explain() ........................................................................ 65
Field Name Restrictions ..................................................... 66
Implicit Transactions .......................................................... 66
Index Builds ....................................................................... 67
MongoDB APIs, Operations, and Data Types ........................ 67
The mongodump and mongorestore Utilities ......................... 67
Multi-key Index ............................................................. 67
Result Ordering .............................................................. 67
Retryable Writes ............................................................. 67
Role-Based Access Control ................................................ 68
Sparse Index ...................................................................... 68
Storage Compression ........................................................ 68
Strings ............................................................................ 68
Using $elemMatch Within an $all Expression ......................... 69
$distinct, $regex, $elemMatch, and $lookup Indexing ............... 69
$lookup ................................................................. 69
Supported MongoDB APIs, Operations, and Data Types .......... 70
Database Commands ......................................................... 70
DeleteDBClusterParameterGroup ................................................................. 410
DeleteDBClusterSnapshot ........................................................................ 412
DeleteDBInstance ................................................................. 414
DeleteDBSubnetGroup ........................................................................... 416
DescribeCertificates ................................................................................. 418
DescribeDBClusterParameterGroups ......................................................... 420
DescribeDBClusterParameters ................................................................. 422
DescribeDBClusters .................................................................................. 424
DescribeDBClusterSnapshotAttributes .................................................... 426
DescribeDBClusterSnapshots ................................................................... 428
DescribeDBEngineVersions ............................................................ 431
DescribeDBInstances ................................................................................ 434
DescribeDBSubnetGroups ........................................................................ 436
DescribeEngineDefaultClusterParameters ............................................. 438
DescribeEventCategories ................................................................. 440
DescribeEvents ....................................................................................... 442
DescribeOrderableDBInstanceOptions .................................................... 445
DescribePendingMaintenanceActions ...................................................... 447
FailoverDBCluster .................................................................................... 449
ListTagsForResource .................................................................................. 451
 ModifyDBCluster ....................................................................................... 453
 ModifyDBClusterParameterGroup ......................................................... 458
 ModifyDBClusterSnapshotAttribute ..................................................... 460
 ModifyDBInstance .................................................................................... 462
 ModifyDBSubnetGroup ............................................................................ 466
 RebootDBInstance .................................................................................... 468
 RemoveTagsFromResource ....................................................................... 470
 ResetDBClusterParameterGroup ......................................................... 472
 RestoreDBClusterFromSnapshot ........................................................... 474
 RestoreDBClusterToPointInTime ............................................................ 479
 StartDBCluster ......................................................................................... 484
 StopDBCluster ......................................................................................... 486
Data Types ................................................................................................. 487
AvailabilityZone ........................................................................................ 488
Certificate ................................................................................................. 489
CloudwatchLogsExportConfiguration ...................................................... 491
DBCluster ................................................................................................. 492
DBClusterMember ..................................................................................... 496
DBClusterParameterGroup ....................................................................... 497
DBClusterRole .......................................................................................... 498
DBClusterSnapshot ................................................................................... 499
DBClusterSnapshotAttribute ................................................................. 502
DBClusterSnapshotAttributesResult ...................................................... 503
DBEngineVersion ..................................................................................... 504
DBInstance ................................................................................................. 506
DBInstanceStatusInfo ............................................................................... 510
DBSubnetGroup ........................................................................................ 511
Endpoint ................................................................................................. 513
EngineDefaults ......................................................................................... 514
Event .......................................................................................................... 515
EventCategoriesMap ............................................................................... 517
Filter ......................................................................................................... 518
OrderableDBInstanceOption ................................................................. 519
Parameter ................................................................................................. 521
PendingCloudwatchLogsExports ............................................................ 523
PendingMaintenanceAction ................................................................. 524
PendingModifiedValues ........................................................................... 526
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResourcePendingMaintenanceActions</td>
<td>529</td>
</tr>
<tr>
<td>Subnet</td>
<td>530</td>
</tr>
<tr>
<td>Tag</td>
<td>531</td>
</tr>
<tr>
<td>UpgradeTarget</td>
<td>532</td>
</tr>
<tr>
<td>VpcSecurityGroupMembership</td>
<td>533</td>
</tr>
<tr>
<td>Common Errors</td>
<td>533</td>
</tr>
<tr>
<td>Common Parameters</td>
<td>535</td>
</tr>
<tr>
<td>Document History</td>
<td>537</td>
</tr>
</tbody>
</table>
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. 6). After that, complete the steps in Getting Started (p. 32).

Topics
- Overview of Amazon DocumentDB (p. 1)
- Clusters (p. 2)
- Instances (p. 2)
- Regions and Availability Zones (p. 3)
- Amazon DocumentDB Pricing (p. 5)
- Monitoring (p. 5)
- Interfaces (p. 5)
- What's Next? (p. 6)
- Amazon DocumentDB: How It Works (p. 6)
- What is a Document Database? (p. 14)

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 TB. 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.
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 is 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. 220).

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

<table>
<thead>
<tr>
<th>Region</th>
<th>R5</th>
<th>R4</th>
</tr>
</thead>
<tbody>
<tr>
<td>US East (Ohio)</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>US East (N. Virginia)</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>US West (Oregon)</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Asia Pacific (Mumbai)</td>
<td>Supported</td>
<td></td>
</tr>
<tr>
<td>Asia Pacific (Seoul)</td>
<td>Supported</td>
<td></td>
</tr>
<tr>
<td>Asia Pacific (Sydney)</td>
<td>Supported</td>
<td></td>
</tr>
<tr>
<td>Asia Pacific (Singapore)</td>
<td>Supported</td>
<td></td>
</tr>
<tr>
<td>Asia Pacific (Tokyo)</td>
<td>Supported</td>
<td></td>
</tr>
<tr>
<td>Canada (Central)</td>
<td>Supported</td>
<td></td>
</tr>
<tr>
<td>Europe (Frankfurt)</td>
<td>Supported</td>
<td></td>
</tr>
<tr>
<td>Europe (Ireland)</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Europe (London)</td>
<td>Supported</td>
<td></td>
</tr>
<tr>
<td>Europe (Paris)</td>
<td>Supported</td>
<td></td>
</tr>
</tbody>
</table>

---

**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.

<table>
<thead>
<tr>
<th>Region Name</th>
<th>Region</th>
<th>Availability Zones (compute)</th>
</tr>
</thead>
<tbody>
<tr>
<td>US East (Ohio)</td>
<td>us-east-2</td>
<td>3</td>
</tr>
<tr>
<td>US East (N. Virginia)</td>
<td>us-east-1</td>
<td>6</td>
</tr>
<tr>
<td>US West (Oregon)</td>
<td>us-west-2</td>
<td>4</td>
</tr>
<tr>
<td>Asia Pacific (Mumbai)</td>
<td>ap-south-1</td>
<td>3</td>
</tr>
<tr>
<td>Asia Pacific (Seoul)</td>
<td>ap-northeast-2</td>
<td>3</td>
</tr>
<tr>
<td>Asia Pacific (Singapore)</td>
<td>ap-southeast-1</td>
<td>3</td>
</tr>
<tr>
<td>Asia Pacific (Sydney)</td>
<td>ap-southeast-2</td>
<td>3</td>
</tr>
<tr>
<td>Asia Pacific (Tokyo)</td>
<td>ap-northeast-1</td>
<td>3</td>
</tr>
<tr>
<td>Canada (Central)</td>
<td>ca-central-1</td>
<td>3</td>
</tr>
<tr>
<td>Europe (Frankfurt)</td>
<td>eu-central-1</td>
<td>3</td>
</tr>
<tr>
<td>Europe (Ireland)</td>
<td>eu-west-1</td>
<td>3</td>
</tr>
<tr>
<td>Europe (London)</td>
<td>eu-west-2</td>
<td>3</td>
</tr>
<tr>
<td>Europe (Paris)</td>
<td>eu-west-3</td>
<td>3</td>
</tr>
</tbody>
</table>
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. 217).
- **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. 155).
- **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. 308)
- Logging Amazon DocumentDB API Calls with AWS CloudTrail (p. 315)

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 3.6 shell, see Step 3: Access and Use Your Amazon DocumentDB Cluster Using the mongo Shell (p. 42).

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. 28).
- Get started with Amazon DocumentDB by creating a cluster and instance using the instructions at Getting Started (p. 32).
- Migrate your MongoDB implementation to Amazon DocumentDB using the guidance at Migrating to Amazon DocumentDB (p. 86)

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.

Topics
- Amazon DocumentDB Endpoints (p. 7)
- TLS Support (p. 10)
- Amazon DocumentDB Storage (p. 10)
- Amazon DocumentDB Replication (p. 10)
- Amazon DocumentDB Reliability (p. 10)
- Read Preference Options (p. 11)
- TTL Deletes (p. 14)

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 TB 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 Viewing CloudWatch Data (p. 308).

**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. 347)) 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. 347)).

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. 280).

### 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. 280).
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. 347)). For more advanced control of instance failover priority, see Understanding Amazon DocumentDB Cluster Fault Tolerance (p. 216).

For information about finding a cluster's endpoints, see Finding an Instance's Endpoint (p. 281).

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. 12).

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. 104).

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. When Amazon DocumentDB data is removed, such as by dropping a table or partition, the overall allocated space remains the same. The free space is reused automatically when data volume increases in the future.

Note
Because 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
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.

**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 primary instance. Attempting to write to a replica 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).

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.

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:

  ```javascript
  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:

  ```javascript
  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. 216).

### 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. 12).

**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.

Instead of using a TTL index to delete documents you no longer need, you can segment documents into collections based on time, and simply drop those 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. Dropping these collections does not incur IO costs, and can be significantly more cost effective than using a TTL index.

**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. 14)
- Understanding Documents (p. 15)
- Working with Documents (p. 19)

**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. 15)
- Real-Time Big Data (p. 15)
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. 17) can be stored in the same document database.

Topics

- SQL vs. Nonrelational Terminology (p. 15)
- Simple Documents (p. 16)
- Embedded Documents (p. 16)
- Example Documents in a Document Database (p. 17)
- Understanding Normalization in a Document Database (p. 18)

SQL vs. Nonrelational Terminology

The following table compares terminology used by document databases (MongoDB) with terminology used by SQL databases.
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.",
    ...
  }
}
```
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:

```json
{
    "_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:

```json
{
    "_id": "0123456789012",
    "Publication": "Programming Today",
    "Issue": {
        "Volume": "14",
        "Number": "09"
    },
    "Articles": [
        {
            "Title": "Is a Document Database Your Best Solution?",
            "Author": {
                "LName": "Major",
                "FName": "Mary"
            }
        },
        {
            "Title": "Databases for Online Solutions",
            "Author": {
                "LName": "Stiles",
                "FName": "John"
            }
        }
    ],
    "Type": "periodical"
}
```
Compare 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",
```

18
"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. 19)
- Querying Documents (p. 21)
- Updating Documents (p. 23)
- Deleting Documents (p. 27)

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 connection 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. 20)
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.

```javascript
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).

```json
{
  "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.

```javascript
db.example.insertMany(
[  
  {  
    "Item": "Pen",
    "Colors": ["Red","Green","Blue","Black"],
    "Inventory": {  
      "OnHand": 244,
      "MinOnHand": 72
    }
  },
  {  
    "Item": "Poster Paint",
    "Colors": ["Red","Green","Blue","Black","White"],
    "Inventory": {  
      "OnHand": 47,
      "MinOnHand": 50
    }
  },
  {  
    "Item": "Spray Paint",
    "Colors": ["Black","Red","Green","Blue"],
    "Inventory": {  
      "OnHand": 47,
      "MinOnHand": 50,
      "OrderQnty": 36
    }
  }
]
```
Output from this operation looks something like the following (JSON format).

```json
{
  "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. 21)
- Retrieving Documents That Match a Field Value (p. 21)
- Retrieving Documents That Match an Embedded Document (p. 22)
- Retrieving Documents That Match a Field Value in an Embedded Document (p. 22)
- Retrieving Documents That Match an Array (p. 22)
- Retrieving Documents That Match a Value in an Array (p. 22)
- Retrieving Documents Using Operators (p. 22)

### 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.

```javascript
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".

```javascript
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 (OrderQnty) than the query document.

```javascript
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.

```javascript
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.

```javascript
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.

```javascript
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.
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. 23)
- Adding a New Field (p. 24)
- Replacing an Embedded Document (p. 25)
- Inserting New Fields into an Embedded Document (p. 25)
- Removing a Field from a Document (p. 26)
- Removing a Field from Multiple Documents (p. 27)

Updating the Values of an Existing Field

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

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

For a listing of supported query operators, see Query and Projection Operators (p. 73).
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".

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

Results from this operation look something like the following.

```javascript
{
  "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.

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

Results from this operation look something like the following.

```javascript
{
  "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).

```json
{
  "Item": "Spray Paint",
  "Colors": ["Black","Red","Green","Blue"],
  "Inventory": {
    "OnHand": 47,
    "MinOnHand": 50,
    "OrderQnty": 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.

```javascript
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.

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

Results from this operation look something like the following.

```json
{
  "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.

```javascript
{
    "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.

```javascript
{
    "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.

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

Results from this operation look something like the following.

```javascript
{
    "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.

```javascript
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.

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

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

```javascript
db.example.remove( { } )
```
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 Getting Started with Amazon DocumentDB (p. 32).

**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 a basis 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. 28)
- Termination Protection and Deletion Protection (p. 30)
- Launching the Amazon DocumentDB AWS CloudFormation Stack (p. 30)
- Accessing the Amazon DocumentDB Cluster (p. 31)

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.

```json
{
    "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:PassRole",
            "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": "***"
    }]
}
```

**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.
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. 212).

Launching the Amazon DocumentDB AWS CloudFormation Stack

To launch the Amazon DocumentDB stack from the AWS CloudFormation console

1. Sign in to the AWS Management Console and open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/home?region=us-east-1#/stacks?filter=active. If you want to work in a different AWS Region than US East (N. Virginia) (us-east-1), choose the Region that you want to create the stack in from the upper-right corner of the console.

2. To launch an Amazon DocumentDB stack in the chosen region, choose the **Launch Stack** button for that region.

<table>
<thead>
<tr>
<th>Region</th>
<th>View Template</th>
<th>View in Designer</th>
<th>Launch</th>
</tr>
</thead>
<tbody>
<tr>
<td>US East (Ohio)</td>
<td>View Template</td>
<td>View in Designer</td>
<td>![Launch Stack](Launch Stack)</td>
</tr>
<tr>
<td>US East (N. Virginia)</td>
<td>View Template</td>
<td>View in Designer</td>
<td>![Launch Stack](Launch Stack)</td>
</tr>
<tr>
<td>US West (Oregon)</td>
<td>View Template</td>
<td>View in Designer</td>
<td>![Launch Stack](Launch Stack)</td>
</tr>
<tr>
<td>Asia Pacific (Mumbai)</td>
<td>View Template</td>
<td>View in Designer</td>
<td>![Launch Stack](Launch Stack)</td>
</tr>
<tr>
<td>Asia Pacific (Seoul)</td>
<td>View Template</td>
<td>View in Designer</td>
<td>![Launch Stack](Launch Stack)</td>
</tr>
<tr>
<td>Asia Pacific (Singapore)</td>
<td>View Template</td>
<td>View in Designer</td>
<td>![Launch Stack](Launch Stack)</td>
</tr>
<tr>
<td>Asia Pacific (Sydney)</td>
<td>View Template</td>
<td>View in Designer</td>
<td>![Launch Stack](Launch Stack)</td>
</tr>
</tbody>
</table>
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 3: Access and Use Your Amazon DocumentDB Cluster Using the mongo Shell (p. 42)
- Step 4: (Optional) Delete the Amazon DocumentDB Instance and Cluster (p. 47)
Getting Started with Amazon DocumentDB

This section describes how to launch an Amazon DocumentDB (with MongoDB compatibility) cluster and interact with it, using the mongo shell.

**Topics**
- Prerequisites (p. 33)
- Step 1: Create an Amazon DocumentDB Cluster (p. 34)
- Step 2: Launch an Amazon EC2 Instance (p. 40)
- Step 3: Access and Use Your Amazon DocumentDB Cluster Using the mongo Shell (p. 42)
- Step 4: (Optional) Delete the Amazon DocumentDB Instance and Cluster (p. 47)

The following short video tutorial walks through deploying an Amazon DocumentDB cluster.
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

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.

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. 110).

If you are using an IAM user to set up and manage your Amazon DocumentDB cluster, attach the AmazonDocDBFullAccess AWS managed policy to the user. The AmazonDocDBFullAccess policy grants full access to all Amazon DocumentDB resources for the AWS account. For more information, see AWS Managed (Predefined) Policies for Amazon DocumentDB (p. 117).

Create an Amazon Virtual Private Cloud (Amazon VPC)

If you don't have a VPC, see Getting Started with Amazon VPC in the Amazon VPC User Guide.

To get started creating the cluster, see Step 1: Create an Amazon DocumentDB Cluster (p. 34).
Step 1: Create an Amazon DocumentDB Cluster

In this step, you create an Amazon DocumentDB cluster in your default Amazon Virtual Private Cloud (Amazon VPC) using the AWS Management Console.

Using the AWS Management Console

The following steps walk you through the basics of creating an Amazon DocumentDB cluster with one or more instances. In these steps, you name your cluster, choose the instance class, and specify the number of instances. You also provide a user name and password that are used to authenticate access to your cluster. Beyond that, this procedure uses the default values for the cluster’s Amazon VPC, port (27017), encryption at rest (enabled), and windows for backups and maintenance. If you prefer to set your own values for these instead of using the default values, follow the procedure at Creating an Amazon DocumentDB Cluster (p. 189).

To create a cluster and add instances to it using the AWS CLI, see both of the following topics:

1. Creating a Cluster Using the AWS CLI (p. 197)
2. Adding an Instance Using the AWS CLI (p. 223)

To create a cluster with instances using the default settings using the AWS Management Console

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. **Instance class**—Accept the default `db.r5.large`, or choose the instance class that you want from the list.

c. **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.

   ![Authentication pane](image)

   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. 194).

   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**

   a. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

   b. In the **Resources** section of the main window, choose **Security groups**.

   ![Security groups](image)

   c. 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.

   ![Security group](image)

   d. From the **Actions** menu, choose **Edit inbound rules** then choose or enter the rule constraints.

   i. **Type**—From the list, choose the protocol to open to network traffic.

   ii. **Protocol**—From the list, choose the type of protocol.

   iii. **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).

   iv. **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).

   v. **Description**—Enter a description for this rule.

   vi. When finished creating the rule, choose **Save**.
Note
You can access the Amazon DocumentDB cluster when the instance status for a given cluster is **available**. This can take several minutes. To monitor an instance's status, see Monitoring an Amazon DocumentDB Instance's Status (p. 305).

While your Amazon DocumentDB cluster is being created, you can proceed to Step 2: Launch an Amazon EC2 Instance (p. 40) and follow the instructions there.

Using the AWS CLI

If you haven’t already done so, go to [http://aws.amazon.com/cli](http://aws.amazon.com/cli) and download the AWS CLI. After you download it, follow the instructions for **Installing the AWS CLI** and **Configuring the AWS CLI** in the AWS Command Line Interface User Guide.

When you create an Amazon DocumentDB cluster using the AWS CLI, you first create the cluster then create instances for the cluster. To create the cluster, use the `create-db-cluster` operation with the following parameters.

- **--db-cluster-identifier**—Required. The name for the cluster.
- **--engine**—Required. Must be `docdb`.
- **--master-username**—Required. The user defined username for accessing this cluster.
- **--master-user-password**—Required. The password for the user to access this cluster.
- For optional parameters, see CreateDBCluster (p. 392) in the Amazon DocumentDB API Reference.

**To create a cluster**

- Create a new Amazon DocumentDB cluster:

  ```sh
  aws docdb create-db-cluster \
  --db-cluster-identifier sample-cluster \
  --engine docdb \
  --master-username yourMasterUsername \
  --master-user-password yourMasterPassword
  ```

  a. Replace `sample-cluster` with the name that you want for your new 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.

b. Replace `yourMasterUsername` with your master user name.

  ```sh
  aws docdb create-db-cluster \
  --db-cluster-identifier sample-cluster \
  --engine docdb \
  --master-username yourMasterUsername \
  --master-user-password yourMasterPassword
  ```

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.

c. Replace `yourMasterPassword` with your 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)

**Note**

To log in to your cluster, you must use the master user name and master password. If you want to configure your cluster beyond what is done here, see Creating an Amazon DocumentDB Cluster (p. 189).

Now create one or more instances for your cluster. To create an instance, use the `create-db-instance` operation with the following parameters.

- `--db-cluster-identifier`—Required. The name of the cluster you want this instance to be part of.
- `--db-instance-class`—Required. The compute and memory class you want for this instance. Example `db.r5.large`.
- `--db-instance-identifier`—Required. The name of this 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, Amazon Neptune, and Amazon DocumentDB) per AWS account, per Region.

- `--engine`—Required. Must be `docdb`.
- For optional parameters, see CreateDBInstance (p. 402) in the Amazon DocumentDB API Reference.

**To create an instance for the cluster**

1. Add an instance to the Amazon DocumentDB cluster:

   ```
   aws docdb create-db-instance \
   --db-cluster-identifier sample-cluster \
   --db-instance-identifier sample-instance \
   --db-instance-class db.r5.large \
   --engine docdb
   ```

   This instance (`sample-instance`) is the primary instance. If you create additional instances, they are the read replica instances. For more information, see Adding an Amazon DocumentDB Instance to a Cluster (p. 221).

2. It takes several minutes to create the cluster and the instances. You can use the AWS CLI to monitor the progress of these actions.

   a. To view the status of the Amazon DocumentDB cluster:
Output from this operation looks something like the following (JSON format).

```
[
  [
    "sample-cluster",
    "available"
  ]
]
```

b. To view the status of the Amazon DocumentDB instance:

```
aws docdb describe-db-instances \
  --db-instance-identifier sample-instance \
  --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceStatus]'
```

**Note**

You can access Amazon DocumentDB when the status for a given cluster is *available*. This can take several minutes.

3. While your Amazon DocumentDB cluster is being created, proceed to Step 2: Launch an Amazon EC2 Instance (p. 40) and follow the instructions there.
Step 2: Launch an Amazon EC2 Instance

Your Amazon DocumentDB cluster should be running in your default virtual private cloud (VPC). To interact with your Amazon DocumentDB cluster, you must launch an Amazon Elastic Compute Cloud (Amazon EC2) instance into your default VPC, in the same AWS Region where you created your Amazon DocumentDB cluster.

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Choose **Launch instance**, and do the following:

   **Step 1: Choose an Amazon Machine Image (AMI)**
   At the top of the list of AMIs, go to **Amazon Linux AMI** and choose **Select**.

   **Step 2: Choose an Instance Type**
   1. At the top of the list of instance types, choose **t2.micro**.
   2. Choose **Next: Configure instance details**.

   **Step 3: Configure Instance Details**
   1. Go to **Network** and choose your default VPC.
   2. Choose **Next: Add Storage**.

   **Step 4: Add Storage**
   Skip this step by choosing **Next: Add tags**.

   **Step 5: Add Tags**
   Skip this step by choosing **Next: Configure security group**.

   **Step 6: Configure Security Group**
   1. Choose **Select an existing security group**.
   2. In the list of security groups, choose **default**. This is the default security group for your VPC. By default, the security group accepts inbound SSH connections on TCP port 22. If this is not the case for your VPC, you can add this rule. For more information, see What Is Amazon VPC? in the Amazon VPC User Guide.
   3. Choose **Next: Review and launch**.

   **Step 7: Review and Launch**
   1. Review all your settings. If any settings need to be changed, go back and change them before continuing.
   2. Choose **Launch**.
   3. In the **Select an existing key pair or create a new key pair** window, do one of the following:
      
      - If you don't have an Amazon EC2 key pair, choose **Create a new key pair** and follow the instructions. You are asked to download a private key file (.pem file). You need this file later when you log in to your Amazon EC2 instance.
      
      - If you already have an Amazon EC2 key pair, go to **Select a key pair** and 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.
4. After you configure your key pair, choose **Launch instances**.

   In the console navigation pane, choose **EC2 Dashboard**, and then choose the instance that you launched. In the lower pane, on the **Description** tab, find the **Public DNS** for your instance, for example: `ec2-11-22-33-44.us-west-2.compute.amazonaws.com`.

   It takes a few minutes for your Amazon EC2 instance to become available.

5. Use the `ssh` operation to log in to your Amazon EC2 instance; for example:

   ```bash
   chmod 400 my-keypair.pem
   ssh -i my-keypair.pem ec2-user@public-dns-name
   ```

   You must specify your private key file (`.pem` file) and the public DNS name of your EC2 instance.

   The login ID is `ec2-user`. No password is required.

   For more information, see **Connecting to Your Linux Instance Using SSH** in the Amazon **EC2 User Guide for Linux Instances**.

Proceed to **Step 3: Access and Use Your Amazon DocumentDB Cluster Using the mongo Shell** (p. 42).
Step 3: Access and Use Your Amazon DocumentDB Cluster Using the mongo Shell

In this step, you connect to your Amazon DocumentDB cluster using the mongo shell. You perform these steps using the Amazon EC2 instance that you created in Step 2: Launch an Amazon EC2 Instance (p. 40).

**Note**
Before you begin, ensure that your Amazon DocumentDB cluster and instance are both available. For information on checking the status of your cluster or instance, see:

- Monitoring an Amazon DocumentDB Cluster's Status (p. 303)
- Monitoring an Amazon DocumentDB Instance's Status (p. 305)

### Step 3.a: Install the mongo shell

Install the mongo shell on your system.

**On Amazon Linux**

**To install the mongo shell on Amazon Linux**

1. Create the repository file `/etc/yum.repos.d/mongodb-org-3.6.repo` with the following contents:

   ```
   [mongodb-org-3.6]
   name=MongoDB Repository
   baseurl=https://repo.mongodb.org/yum/amazon/2013.03/mongodb-org/3.6/x86_64/
gpgcheck=1
   enabled=1
   gpgkey=https://www.mongodb.org/static/pgp/server-3.6.asc
   ```

   The following one line command creates the repository file.

   ```bash
   echo -e "[mongodb-org-3.6] \nname=MongoDB Repository\nbaseurl=https://repo.mongodb.org/yum/amazon/2013.03/mongodb-org/3.6/x86_64/\ngpgcheck=1 \nenabled=1 \ngpgkey=https://www.mongodb.org/static/pgp/server-3.6.asc" | sudo tee /etc/yum.repos.d/mongodb-org-3.6.repo
   ```

2. Install the MongoDB shell.

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

   For information about installing earlier versions of MongoDB on your Amazon Linux system, see Install MongoDB Community Edition on Amazon Linux in the MongoDB documentation.

**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.
Step 3.b: Manage Amazon DocumentDB TLS

Transport Layer Security (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. 104).

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.

```
```

**Important**
The certificate authority (CA) certificate for Amazon DocumentDB 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 and certificates, see Updating Your Amazon DocumentDB TLS Certificates (p. 139).

Step 3.c: Connect and Use Amazon DocumentDB

**To connect to and use your Amazon DocumentDB cluster**

1. Get the mongo connection command for the cluster:

b. If the cluster you want to connect to is not in the US East (N. Virginia) Region, in the upper-right corner of the console, change to your cluster's Region.

c. In the left navigation pane, choose Clusters. Then from the list of clusters, choose the name of the cluster that you want to connect to.

d. On the cluster's detail page, locate the Connect section. Then locate and copy the mongo shell connection string. Use this string in the next step.

2. Use the mongo shell to connect to Amazon DocumentDB.

To connect to your cluster using mongo, paste the connection command you copied in the previous step to your command window. Then, after changing the <insertYourPassword> to your master password, run the command.

Example: Line breaks are added for readability only.

```
mongo --ssl
    --host endpoint
    --sslCAFile rds-combined-ca-bundle.pem
    --username yourMasterUsername
    --password yourMasterPassword
```

3. From the mongo shell, issue a few operations:

a. Insert a new document:

```javascript
db.inventory.insert({
    "SKU" : "38220349",
    "Description" : "14 oz. whole wheat bread",
    "Vendor" : {
        "Name" : "Example Bakery, LLC.",
        "Street" : "321 No. Gilbert",
        "City" : "Anytown",
        "State-Provence" : "CA",
        "Country" : "USA",
        "Phone" : "000-555-0100",
        "Contacts" : [
            {
                "LName" : "Stiles",
                "FName" : "Jonathan",
                "FriendlyName" : "JJ",
                "Role" : "Sales",
                "Phone" : "000-555-0111"
            }
        ],
    },
    "Stock" : {
        "OnHand" : "4",
        "OrderWhenBelow" : "6",
        "OrderQuantity" : "8"
    },
    "UnitPrice" : {
        "Wholesale" : "1.71",
    }
});
```
b. Find all documents:

```javascript
db.inventory.find( {} ).pretty()
```

c. Update a document:

The following example updates the document by replacing the current `UnitPrice` embedded document with a new one, which has new Wholesale and Retail prices, and adding a DayOld price.

```javascript
db.inventory.update(
    { "SKU" : "38220349" },
    {
        "$set": { "UnitPrice": { "Wholesale": "1.87", "Retail": "2.69", "DayOld": "1.89" } }
    }
)
```

When the operation is finished, the embedded `UnitPrice` document looks like the following. The rest of the document is unchanged.

```javascript
{
    "SKU" : "38220349",
    "Description" : "14 oz. whole wheat bread",
    "Vendor" : {
        "Name" : "Example Bakery, LLC.",
        "Street" : "321 No. Gilbert",
        "City" : "Anytown",
        "State-Provence" : "CA",
        "Country" : "USA",
        "Phone" : "000-555-0100",
        "Contacts" : [
            {
                "LName" : "Stiles",
                "FName" : "Jonathan",
                "FriendlyName" : "JJ",
                "Role" : "Sales",
                "Phone" : "000-555-0111"
            }
        ],
    },
    "Stock" : {
        "OnHand" : "4",
        "OrderWhenBelow" : "6",
        "OrderQuantity" : "8"
    },
    "UnitPrice" : {
        "Wholesale" : "1.87",
        "Retail" : "2.69",
        "DayOld" : "1.89"
    }
}
```

d. Delete a document:

```javascript
db.inventory.remove( { "SKU" : "38220349" } )
```
4. When you are finished, exit from the mongo shell:

```
exit
```

To delete your cluster and stop incurring costs, go to Step 4: (Optional) Delete the Amazon DocumentDB Instance and Cluster (p. 47).
Step 4: (Optional) Delete the Amazon DocumentDB Instance and Cluster

If you are no longer using your Amazon DocumentDB cluster, you should either stop or delete it so that you aren't charged for resources that you're not using. When you delete the cluster you lose all its data unless you created a manual backup. When you stop a cluster you save the costs of instance-hours and retain your data. For information on stopping a cluster, see Stopping and Starting an Amazon DocumentDB Cluster (p. 209).

Deleting a cluster cannot be reversed, though you can restore the cluster from a manual or final backup if you make one.

Amazon DocumentDB enforces deletion protection for a cluster whether you perform the delete operation using the console or the AWS CLI. If deletion protection enabled, you can't delete a cluster. To delete a cluster that has deletion projection 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.

To determine whether a cluster has deletion protection enabled

2. Choose the cluster's name, and scroll to the bottom of 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. 203).

After **Deletion protection** is disabled, you are ready to delete the cluster.

To delete a cluster

1. In the navigation pane, choose **Clusters**. Find the cluster that you want to delete, and choose the button to the left of the cluster name.
2. Determine whether the cluster has any instances by checking the **Instances** column.

3. Depending on whether your cluster has any instances, do one of the following steps.
   - If the cluster has no instances, choose **Actions**, and then 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. 234).

It takes several minutes for a cluster to be deleted. To monitor the status of the cluster, see Monitoring an Amazon DocumentDB Cluster's Status (p. 303).
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. 49)
- Instance Sizing (p. 50)
- Working with Indexes (p. 51)
- Security Best Practices (p. 51)
- Cost Optimization (p. 52)
- Using Metrics to Identify Performance Issues (p. 52)
- TTL and Timeseries Workloads (p. 54)
- Working with Cluster Parameter Groups (p. 54)
- Aggregation Pipeline Queries (p. 55)
- batchInsert and batchUpdate (p. 55)

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. 353).
- 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. 247).
- Connect to your Amazon DocumentDB cluster with the cluster endpoint (see Amazon DocumentDB Endpoints (p. 7) and in replica set mode (see Connecting to Amazon DocumentDB as a Replica Set (p. 347) 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. 11).

- 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. 30).

Instance Sizing

It is an Amazon DocumentDB performance 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.

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. Roughly two-thirds of an instance's RAM is available for working set memory. For information about monitoring an instance's metrics, see Viewing CloudWatch Data (p. 308).

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. 86).

**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. 50).

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 and removing 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. 377).

**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. 377).

**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.

- 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.

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. 209).
- 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:
• Viewing CloudWatch Data (Amazon DocumentDB Console) (p. 309)
• Viewing CloudWatch Data (AWS CLI) (p. 309)

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 one-third of the total instance memory, consider scaling up your instances.
• 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. 316)) 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.

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.

**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. 155).
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`. 
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. 56)
- Step 2: Update the Server Certificate (p. 57)

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 certificate.

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.

```bash
get 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 (rds-ca-2015-root.pem) 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. 61) 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. 104) and Connecting with TLS Enabled (p. 325).

Note
Currently, the MongoDB Go Driver 1.2.1 only accepts one CA server certificate in sslcertificateauthorityfile. Please see Connecting with TLS Enabled (p. 325) 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.

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. If your intention is to modify the server certificate immediately, use the Apply Immediately option.
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.

   ```bash
   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.

   ```bash
   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. 231).

- **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. 61).

- **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. 59).

- **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. 323).

- **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
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 previous CA and server certificates expired on Thursday, March 5, 2020.

What happens if I don't take any action by March 5, 2020?

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 March 5, 2020. You must update your applications and clusters to use the new CA certificates before or after March 5, 2020.

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

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**

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.

   ```bash
   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.

   ```bash
   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. 61)

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:
How can I be sure that I'm using the newest CA bundle?

```
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:**
```
a4ded73667097aa2d97d28a469f1fec94912a166 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:**
```
a4ded73667097aa2d97d28a469f1fec94912a166 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:
```
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

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
```
Functional Differences: Amazon DocumentDB and MongoDB

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

Topics
• Admin Databases and Collections (p. 65)
• Array Indexing (p. 65)
• explain() (p. 65)
• Field Name Restrictions (p. 66)
• Implicit Transactions (p. 66)
• Index Builds (p. 67)
• MongoDB APIs, Operations, and Data Types (p. 67)
• The mongodump and mongorestore Utilities (p. 67)
• Multi-key Index (p. 67)
• Result Ordering (p. 67)
• Retryable Writes (p. 67)
• Role-Based Access Control (p. 68)
• Sparse Index (p. 68)
• Storage Compression (p. 68)
• Strings (p. 68)
• Using $elemMatch Within an $all Expression (p. 69)
• $distinct, $regex, $elemMatch, and $lookup Indexing (p. 69)
• $lookup (p. 69)

Admin Databases and Collections

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

Array Indexing

Amazon DocumentDB indexes an array as a single entry. Arrays larger than 2048 bytes cannot currently be indexed.

explain()

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.
Field Name Restrictions

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

Implicit Transactions

In Amazon DocumentDB, all CRUD statements (findAndModify, update, insert, delete) guarantee atomicity and consistency, even for operations that modify multiple documents. This behavior is different than MongoDB 3.6, which only provides atomic guarantees for commands that modify a single document. The following are examples of operations in Amazon DocumentDB that modify multiple documents that satisfy both atomic and consistent behaviors.

```javascript
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.
Index Builds

Amazon DocumentDB allows only one index build to occur on a collection at any given time (foreground or 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.

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

MongoDB APIs, Operations, and Data Types

Amazon DocumentDB is compatible with the MongoDB 3.6 API. For an up-to-date list of supported functionality, see Supported MongoDB APIs, Operations, and Data Types (p. 70).

The `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.

Multi-key Index

Amazon DocumentDB does not yet support creating a compound index with multiple keys on the same array. A workaround is to create individual indexes on the desired array fields because the Amazon DocumentDB query planner can use multiple indexes in a single query.

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.

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

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.
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. 125). Amazon DocumentDB does not yet support custom roles for RBAC.

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 }})

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.

Strings

Amazon DocumentDB does not allow null characters (\'\0\') in strings.

db.inventory.insert({"description": "\0"})
WriteResult({
  "writeError": {
    "code": 303,
    "errmsg": "Unsupported BSON : has null character in string"
  }
})
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:

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

Updated operation:

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

$distinct, $regex, $elemMatch, and $lookup Indexing

Amazon DocumentDB does not currently support the ability to use indexes with the $distinct, $regex, $elemMatch, and $lookup 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.
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 API. This section lists the supported functionality. For functional differences between Amazon DocumentDB and MongoDB, please see Functional Differences: Amazon DocumentDB and MongoDB (p. 65).

MongoDB commands and operators that are No as of version 3.6, internal-only, or not applicable to a fully-managed service are not supported and are not included in the list of supported functionality. All capabilities marked as Yes have been available in the service since launch. New capabilities that are added will be marked with Yes and the date it was launched.

Database Commands

Aggregation

<table>
<thead>
<tr>
<th>Command</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>aggregate</td>
<td>Yes</td>
</tr>
<tr>
<td>count</td>
<td>Yes</td>
</tr>
<tr>
<td>distinct</td>
<td>Yes</td>
</tr>
<tr>
<td>mapReduce</td>
<td>No</td>
</tr>
</tbody>
</table>

Query and Write Operations

<table>
<thead>
<tr>
<th>Command</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>delete</td>
<td>Yes</td>
</tr>
<tr>
<td>find</td>
<td>Yes</td>
</tr>
<tr>
<td>findAndModify</td>
<td>Yes</td>
</tr>
<tr>
<td>getLastError</td>
<td>No</td>
</tr>
<tr>
<td>getMore</td>
<td>Yes</td>
</tr>
<tr>
<td>getPrevError</td>
<td>No</td>
</tr>
<tr>
<td>insert</td>
<td>Yes</td>
</tr>
</tbody>
</table>
## Authentication

<table>
<thead>
<tr>
<th>Command</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>authenticate</td>
<td>Yes</td>
</tr>
<tr>
<td>logout</td>
<td>Yes</td>
</tr>
</tbody>
</table>

## User Management

<table>
<thead>
<tr>
<th>Command</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>createUser</td>
<td>Yes</td>
</tr>
<tr>
<td>dropAllUsersFromDatabase</td>
<td>Yes</td>
</tr>
<tr>
<td>dropUser</td>
<td>Yes</td>
</tr>
<tr>
<td>grantRolesToUser</td>
<td>Yes (03/26/20)</td>
</tr>
<tr>
<td>revokeRolesFromUser</td>
<td>Yes (03/26/20)</td>
</tr>
<tr>
<td>updateUser</td>
<td>Yes</td>
</tr>
<tr>
<td>userInfo</td>
<td>Yes</td>
</tr>
</tbody>
</table>

## Role Management Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>createRole</td>
<td>No</td>
</tr>
<tr>
<td>dropRole</td>
<td>No</td>
</tr>
<tr>
<td>dropAllRolesFromDatabase</td>
<td>No</td>
</tr>
<tr>
<td>grantRolesToRole</td>
<td>No</td>
</tr>
<tr>
<td>revokePrivilegesFromRole</td>
<td>No</td>
</tr>
<tr>
<td>revokeRolesFromRole</td>
<td>No</td>
</tr>
</tbody>
</table>
## Administrative Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>updateRole</td>
<td>No</td>
</tr>
<tr>
<td>rolesInfo</td>
<td>No</td>
</tr>
<tr>
<td>Capped Collections</td>
<td>No</td>
</tr>
<tr>
<td>cloneCollectionAsCapped</td>
<td>No</td>
</tr>
<tr>
<td>collMod</td>
<td>Partial</td>
</tr>
<tr>
<td>collMod: expireAfterSeconds</td>
<td>Yes (8/1/19)</td>
</tr>
<tr>
<td>convertToCapped</td>
<td>No</td>
</tr>
<tr>
<td>copydb</td>
<td>No</td>
</tr>
<tr>
<td>create</td>
<td>Yes</td>
</tr>
<tr>
<td>createIndexes</td>
<td>Yes</td>
</tr>
<tr>
<td>currentOp</td>
<td>Yes</td>
</tr>
<tr>
<td>drop</td>
<td>Yes</td>
</tr>
<tr>
<td>dropDatabase</td>
<td>Yes</td>
</tr>
<tr>
<td>dropIndexes</td>
<td>Yes</td>
</tr>
<tr>
<td>filemd5</td>
<td>No</td>
</tr>
<tr>
<td>killCursors</td>
<td>Yes</td>
</tr>
<tr>
<td>killOp</td>
<td>Yes</td>
</tr>
<tr>
<td>listCollections</td>
<td>Yes</td>
</tr>
<tr>
<td>listDatabases</td>
<td>Yes</td>
</tr>
<tr>
<td>listIndexes</td>
<td>Yes</td>
</tr>
<tr>
<td>relIndex</td>
<td>No</td>
</tr>
<tr>
<td>renameCollection</td>
<td>No</td>
</tr>
</tbody>
</table>

## Diagnostic Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>buildInfo</td>
<td>Yes</td>
</tr>
<tr>
<td>collStats</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### Query and Projection Operators

#### Comparison Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$eq</td>
<td>Yes</td>
</tr>
<tr>
<td>$gt</td>
<td>Yes</td>
</tr>
<tr>
<td>$gte</td>
<td>Yes</td>
</tr>
<tr>
<td>$lt</td>
<td>Yes</td>
</tr>
<tr>
<td>$lte</td>
<td>Yes</td>
</tr>
<tr>
<td>$ne</td>
<td>Yes</td>
</tr>
<tr>
<td>$in</td>
<td>Yes</td>
</tr>
<tr>
<td>$nin</td>
<td>Yes</td>
</tr>
</tbody>
</table>

#### Logical Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$or</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### Element Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$and</td>
<td>Yes</td>
</tr>
<tr>
<td>$not</td>
<td>Yes</td>
</tr>
<tr>
<td>$nor</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### Evaluation Query Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$exists</td>
<td>Yes</td>
</tr>
<tr>
<td>$type</td>
<td>Yes</td>
</tr>
<tr>
<td>$expr</td>
<td>No</td>
</tr>
<tr>
<td>$jsonSchema</td>
<td>No</td>
</tr>
<tr>
<td>$mod</td>
<td>Yes (2/6/20)</td>
</tr>
<tr>
<td>$regex</td>
<td>Yes</td>
</tr>
<tr>
<td>$text</td>
<td>No</td>
</tr>
<tr>
<td>$where</td>
<td>No</td>
</tr>
</tbody>
</table>

### Array Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$all</td>
<td>Yes</td>
</tr>
<tr>
<td>$elemMatch</td>
<td>Yes</td>
</tr>
<tr>
<td>$size</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### Bitwise Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$bitsAllSet</td>
<td>Yes</td>
</tr>
<tr>
<td>$bitsAnySet</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Comment Operator

<table>
<thead>
<tr>
<th>Operator</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$bitsAllClear</td>
<td>Yes</td>
</tr>
<tr>
<td>$bitsAnyClear</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Projection Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>Yes</td>
</tr>
<tr>
<td>$elemMatch</td>
<td>Yes</td>
</tr>
<tr>
<td>$meta</td>
<td>No</td>
</tr>
<tr>
<td>$slice</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Update Operators

Field Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$inc</td>
<td>Yes</td>
</tr>
<tr>
<td>$mul</td>
<td>Yes</td>
</tr>
<tr>
<td>$rename</td>
<td>Yes</td>
</tr>
<tr>
<td>$setOnInsert</td>
<td>Yes</td>
</tr>
<tr>
<td>$set</td>
<td>Yes</td>
</tr>
<tr>
<td>$unset</td>
<td>Yes</td>
</tr>
<tr>
<td>$min</td>
<td>Yes</td>
</tr>
<tr>
<td>$max</td>
<td>Yes</td>
</tr>
<tr>
<td>$currentDate</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Array Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>Yes</td>
</tr>
<tr>
<td>$[]</td>
<td>Yes (2/28/19)</td>
</tr>
<tr>
<td>$[&lt;identifier&gt;]</td>
<td>Yes (2/28/19)</td>
</tr>
<tr>
<td>$addToSet</td>
<td>Yes</td>
</tr>
<tr>
<td>$pop</td>
<td>Yes</td>
</tr>
<tr>
<td>$pullAll</td>
<td>Yes</td>
</tr>
<tr>
<td>$pull</td>
<td>Yes</td>
</tr>
<tr>
<td>$push</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Update Modifiers

<table>
<thead>
<tr>
<th>Operator</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$each</td>
<td>Yes</td>
</tr>
<tr>
<td>$slice</td>
<td>Yes</td>
</tr>
<tr>
<td>$sort</td>
<td>Yes</td>
</tr>
<tr>
<td>$position</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Bitwise Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$bit</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Geospatial

Query Selectors

<table>
<thead>
<tr>
<th>Query Selectors</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$geoIntersects</td>
<td>No</td>
</tr>
<tr>
<td>$geoWithin</td>
<td>No</td>
</tr>
<tr>
<td>$near</td>
<td>No</td>
</tr>
</tbody>
</table>
Geometry Specifiers

<table>
<thead>
<tr>
<th>Query Selectors</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$box</td>
<td>No</td>
</tr>
<tr>
<td>$center</td>
<td>No</td>
</tr>
<tr>
<td>$centerSphere</td>
<td>No</td>
</tr>
<tr>
<td>$nearSphere</td>
<td>No</td>
</tr>
<tr>
<td>$geometry</td>
<td>No</td>
</tr>
<tr>
<td>$maxDistance</td>
<td>No</td>
</tr>
<tr>
<td>$minDistance</td>
<td>No</td>
</tr>
<tr>
<td>$polygon</td>
<td>No</td>
</tr>
<tr>
<td>$uniqueDocs</td>
<td>No</td>
</tr>
</tbody>
</table>

Cursor Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>cursor.batchSize()</td>
<td>Yes</td>
</tr>
<tr>
<td>cursor.close()</td>
<td>Yes</td>
</tr>
<tr>
<td>cursor.isClosed()</td>
<td>Yes</td>
</tr>
<tr>
<td>cursor.collation()</td>
<td>No</td>
</tr>
<tr>
<td>cursor.comment()</td>
<td>Yes</td>
</tr>
<tr>
<td>cursor.count()</td>
<td>Yes</td>
</tr>
<tr>
<td>cursor.explain()</td>
<td>Yes</td>
</tr>
<tr>
<td>cursor.forEach()</td>
<td>Yes</td>
</tr>
<tr>
<td>cursor.hasNext()</td>
<td>Yes</td>
</tr>
<tr>
<td>cursor.hint()</td>
<td>Yes (2/28/19)</td>
</tr>
<tr>
<td>cursor.isExhausted()</td>
<td>Yes</td>
</tr>
<tr>
<td>cursor.itcount()</td>
<td>Yes</td>
</tr>
<tr>
<td>cursor.limit()</td>
<td>Yes</td>
</tr>
<tr>
<td>cursor.map()</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### Method Supported

<table>
<thead>
<tr>
<th>Method</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>cursor.maxScan()</td>
<td>Yes</td>
</tr>
<tr>
<td>cursor.maxTimeMS()</td>
<td>Yes</td>
</tr>
<tr>
<td>cursor.max()</td>
<td>No</td>
</tr>
<tr>
<td>cursor.min()</td>
<td>No</td>
</tr>
<tr>
<td>cursor.next()</td>
<td>Yes</td>
</tr>
<tr>
<td>cursor.noCursorTimeout()</td>
<td>No</td>
</tr>
<tr>
<td>cursor.objsLeftInBatch()</td>
<td>Yes</td>
</tr>
<tr>
<td>cursor.pretty()</td>
<td>Yes</td>
</tr>
<tr>
<td>cursor.readConcern()</td>
<td>Yes</td>
</tr>
<tr>
<td>cursor.readPref()</td>
<td>Yes</td>
</tr>
<tr>
<td>cursor.returnKey()</td>
<td>No</td>
</tr>
<tr>
<td>cursor.showRecordId()</td>
<td>No</td>
</tr>
<tr>
<td>cursor.size()</td>
<td>Yes</td>
</tr>
<tr>
<td>cursor.skip()</td>
<td>Yes</td>
</tr>
<tr>
<td>cursor.sort()</td>
<td>Yes</td>
</tr>
<tr>
<td>cursor.tailable()</td>
<td>No</td>
</tr>
<tr>
<td>cursor.toArray()</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### Aggregation Pipeline Operators

#### Stage Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$collStats</td>
<td>No</td>
</tr>
<tr>
<td>$project</td>
<td>Yes</td>
</tr>
<tr>
<td>$match</td>
<td>Yes</td>
</tr>
<tr>
<td>$redact</td>
<td>Yes (2/28/19)</td>
</tr>
<tr>
<td>$limit</td>
<td>Yes</td>
</tr>
<tr>
<td>$skip</td>
<td>Yes</td>
</tr>
<tr>
<td>$unwind</td>
<td>Yes</td>
</tr>
<tr>
<td>$group</td>
<td>Yes</td>
</tr>
<tr>
<td>$sample</td>
<td>Yes (4/4/19)</td>
</tr>
</tbody>
</table>
### Boolean Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$sort</td>
<td>Yes</td>
</tr>
<tr>
<td>$geoNear</td>
<td>No</td>
</tr>
<tr>
<td>$lookup</td>
<td>Yes (10/15/19)</td>
</tr>
<tr>
<td>$out</td>
<td>No</td>
</tr>
<tr>
<td>$indexStats</td>
<td>Yes (2/28/19)</td>
</tr>
<tr>
<td>$facet</td>
<td>No</td>
</tr>
<tr>
<td>$bucket</td>
<td>No</td>
</tr>
<tr>
<td>$bucketAuto</td>
<td>No</td>
</tr>
<tr>
<td>$sortByCount</td>
<td>No</td>
</tr>
<tr>
<td>$addFields</td>
<td>Yes (10/15/19)</td>
</tr>
<tr>
<td>$replaceRoot</td>
<td>No</td>
</tr>
<tr>
<td>$count</td>
<td>Yes</td>
</tr>
<tr>
<td>$currentOp</td>
<td>Yes</td>
</tr>
<tr>
<td>$listLocalSessions</td>
<td>No</td>
</tr>
<tr>
<td>$listSessions</td>
<td>No</td>
</tr>
<tr>
<td>$graphLookup</td>
<td>No</td>
</tr>
</tbody>
</table>

### Boolean Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$and</td>
<td>Yes</td>
</tr>
<tr>
<td>$or</td>
<td>Yes</td>
</tr>
<tr>
<td>$not</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### Set Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$setEquals</td>
<td>No</td>
</tr>
<tr>
<td>$setIntersection</td>
<td>No</td>
</tr>
<tr>
<td>$setUnion</td>
<td>No</td>
</tr>
<tr>
<td>$setDifference</td>
<td>No</td>
</tr>
<tr>
<td>$setIsSubset</td>
<td>No</td>
</tr>
</tbody>
</table>
### Comparison Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>$anyElementTrue</code></td>
<td>No</td>
</tr>
<tr>
<td><code>$allElementsTrue</code></td>
<td>No</td>
</tr>
</tbody>
</table>

### Arithmetic Operators

<table>
<thead>
<tr>
<th>Command</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>$abs</code></td>
<td>Yes</td>
</tr>
<tr>
<td><code>$add</code></td>
<td>Yes</td>
</tr>
<tr>
<td><code>$ceil</code></td>
<td>No</td>
</tr>
<tr>
<td><code>$divide</code></td>
<td>Yes</td>
</tr>
<tr>
<td><code>$exp</code></td>
<td>No</td>
</tr>
<tr>
<td><code>$floor</code></td>
<td>No</td>
</tr>
<tr>
<td><code>$ln</code></td>
<td>No</td>
</tr>
<tr>
<td><code>$log</code></td>
<td>No</td>
</tr>
<tr>
<td><code>$log10</code></td>
<td>No</td>
</tr>
<tr>
<td><code>$mod</code></td>
<td>Yes (2/6/20)</td>
</tr>
<tr>
<td><code>$multiply</code></td>
<td>Yes</td>
</tr>
<tr>
<td><code>$pow</code></td>
<td>No</td>
</tr>
<tr>
<td><code>$sqrt</code></td>
<td>No</td>
</tr>
<tr>
<td><code>$subtract</code></td>
<td>Yes</td>
</tr>
<tr>
<td><code>$trunc</code></td>
<td>No</td>
</tr>
</tbody>
</table>
## String Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$concat</td>
<td>Yes (2/28/19)</td>
</tr>
<tr>
<td>$indexOfBytes</td>
<td>Yes (4/4/19)</td>
</tr>
<tr>
<td>$indexOfCP</td>
<td>Yes (4/4/19)</td>
</tr>
<tr>
<td>$split</td>
<td>Yes (4/4/19)</td>
</tr>
<tr>
<td>$strcasecmp</td>
<td>Yes (2/28/19)</td>
</tr>
<tr>
<td>$strLenBytes</td>
<td>Yes (4/4/19)</td>
</tr>
<tr>
<td>$strLenCP</td>
<td>Yes (4/4/19)</td>
</tr>
<tr>
<td>$substr</td>
<td>Yes (2/28/19)</td>
</tr>
<tr>
<td>$substrBytes</td>
<td>Yes (2/28/19)</td>
</tr>
<tr>
<td>$substrCP</td>
<td>Yes (2/28/19)</td>
</tr>
<tr>
<td>$toLowerCase</td>
<td>Yes (4/4/19)</td>
</tr>
<tr>
<td>$toUpperCase</td>
<td>Yes (4/4/19)</td>
</tr>
</tbody>
</table>

## Text Search Operator

<table>
<thead>
<tr>
<th>Operator</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$meta</td>
<td>No</td>
</tr>
</tbody>
</table>

## Array Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$arrayElemAt</td>
<td>Yes</td>
</tr>
<tr>
<td>$arrayToObject</td>
<td>Yes (2/6/20)</td>
</tr>
<tr>
<td>$concatArrays</td>
<td>Yes (10/15/19)</td>
</tr>
<tr>
<td>$filter</td>
<td>Yes</td>
</tr>
<tr>
<td>$indexOfArray</td>
<td>No</td>
</tr>
<tr>
<td>$isArray</td>
<td>Yes</td>
</tr>
<tr>
<td>$objectToArray</td>
<td>Yes (2/6/20)</td>
</tr>
<tr>
<td>$range</td>
<td>Yes (2/6/20)</td>
</tr>
<tr>
<td>$reverseArray</td>
<td>No</td>
</tr>
</tbody>
</table>
Variable Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$reduce</td>
<td>No</td>
</tr>
<tr>
<td>$size</td>
<td>Yes (2/28/19)</td>
</tr>
<tr>
<td>$slice</td>
<td>Yes (2/6/20)</td>
</tr>
<tr>
<td>$zip</td>
<td>No</td>
</tr>
<tr>
<td>$in</td>
<td>Yes (8/1/19)</td>
</tr>
</tbody>
</table>

System Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$$CURRENT</td>
<td>No</td>
</tr>
<tr>
<td>$$DESCEND</td>
<td>Yes (2/28/19)</td>
</tr>
<tr>
<td>$$KEEP</td>
<td>Yes (2/28/19)</td>
</tr>
<tr>
<td>$$PRUNE</td>
<td>Yes (2/28/19)</td>
</tr>
<tr>
<td>$$REMOVE</td>
<td>No</td>
</tr>
<tr>
<td>$$ROOT</td>
<td>No</td>
</tr>
</tbody>
</table>

Literal Operator

<table>
<thead>
<tr>
<th>Operator</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$literal</td>
<td>No</td>
</tr>
</tbody>
</table>

Date Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$dayOfYear</td>
<td>Yes (4/4/19)</td>
</tr>
<tr>
<td>$dayOfMonth</td>
<td>Yes (4/4/19)</td>
</tr>
</tbody>
</table>
## Conditional Expression Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$dayOfWeek</td>
<td>Yes (4/4/19)</td>
</tr>
<tr>
<td>$year</td>
<td>Yes (4/4/19)</td>
</tr>
<tr>
<td>$month</td>
<td>Yes (4/4/19)</td>
</tr>
<tr>
<td>$week</td>
<td>Yes (8/1/19)</td>
</tr>
<tr>
<td>$hour</td>
<td>Yes (4/4/19)</td>
</tr>
<tr>
<td>$minute</td>
<td>Yes (4/4/19)</td>
</tr>
<tr>
<td>$second</td>
<td>Yes (4/4/19)</td>
</tr>
<tr>
<td>$millisecond</td>
<td>Yes (4/4/19)</td>
</tr>
<tr>
<td>$dateToString</td>
<td>No</td>
</tr>
<tr>
<td>$isoDayOfWeek</td>
<td>Yes (8/1/19)</td>
</tr>
<tr>
<td>$isoWeek</td>
<td>Yes (8/1/19)</td>
</tr>
<tr>
<td>$dateFromParts</td>
<td>No</td>
</tr>
<tr>
<td>$dateToParts</td>
<td>No</td>
</tr>
<tr>
<td>$dateFromTimeString</td>
<td>Yes (03/23/20)</td>
</tr>
<tr>
<td>$isoWeekYear</td>
<td>Yes (08/01/19)</td>
</tr>
</tbody>
</table>

### Conditional Expression Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$cond</td>
<td>Yes</td>
</tr>
<tr>
<td>$ifNull</td>
<td>No</td>
</tr>
<tr>
<td>$switch</td>
<td>No</td>
</tr>
</tbody>
</table>

### Data Type Operator

<table>
<thead>
<tr>
<th>Operator</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$type</td>
<td>No</td>
</tr>
</tbody>
</table>

### Accumulator Expressions

<table>
<thead>
<tr>
<th>Expression</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$sum</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### Merge Operator

<table>
<thead>
<tr>
<th>Operator</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>$mergeObjects</td>
<td>No</td>
</tr>
</tbody>
</table>

### Data Types

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Double</td>
<td>Yes</td>
</tr>
<tr>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>Object</td>
<td>Yes</td>
</tr>
<tr>
<td>Array</td>
<td>Yes</td>
</tr>
<tr>
<td>Binary Data</td>
<td>Yes</td>
</tr>
<tr>
<td>ObjectId</td>
<td>Yes</td>
</tr>
<tr>
<td>Boolean</td>
<td>Yes</td>
</tr>
<tr>
<td>Date</td>
<td>Yes</td>
</tr>
<tr>
<td>Null</td>
<td>Yes</td>
</tr>
<tr>
<td>32-bit Integer (int)</td>
<td>Yes</td>
</tr>
<tr>
<td>Timestamp</td>
<td>Yes</td>
</tr>
<tr>
<td>64-bit Integer (long)</td>
<td>Yes</td>
</tr>
<tr>
<td>MinKey</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### Indexes and Index Properties

#### Indexes

<table>
<thead>
<tr>
<th>Index Type</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Field Index</td>
<td>Yes</td>
</tr>
<tr>
<td>Compound Index</td>
<td>Yes</td>
</tr>
<tr>
<td>Multikey Index</td>
<td>Yes</td>
</tr>
<tr>
<td>Text Index</td>
<td>No</td>
</tr>
<tr>
<td>2dsphere</td>
<td>No</td>
</tr>
<tr>
<td>2d Index</td>
<td>No</td>
</tr>
<tr>
<td>Hashed Index</td>
<td>No</td>
</tr>
</tbody>
</table>

#### Index Properties

<table>
<thead>
<tr>
<th>Index Property</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTL</td>
<td>Yes</td>
</tr>
<tr>
<td>Unique</td>
<td>Yes</td>
</tr>
<tr>
<td>Partial</td>
<td>No</td>
</tr>
<tr>
<td>Case Insensitive</td>
<td>No</td>
</tr>
<tr>
<td>Sparse</td>
<td>Yes</td>
</tr>
<tr>
<td>Background</td>
<td>Yes</td>
</tr>
</tbody>
</table>
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
- Migration Tools (p. 86)
- Discovery (p. 87)
- Planning: Amazon DocumentDB Cluster Requirements (p. 89)
- Migration Approaches (p. 91)
- Migration Sources (p. 94)
- Migration Connectivity (p. 95)
- Testing (p. 97)
- Performance Testing (p. 99)
- Failover Testing (p. 99)
- Additional Resources (p. 99)

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`.

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.

To help with the cost of migrations, you can use AWS DMS free for six months per instance when migrating to Amazon DocumentDB. For more information, see Free DMS.

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. 91) 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.

• Oplog sizes

  For replica sets or sharded clusters, record the size of the oplog for each replica set member. To 
  find a member’s oplog 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.
<table>
<thead>
<tr>
<th>Availability Goal</th>
<th>Total Instances</th>
<th>Replicas</th>
<th>Availability Zones</th>
</tr>
</thead>
<tbody>
<tr>
<td>99%</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>99.9%</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>99.99%</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

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](https://docs.aws.amazon.com/wellarchitected/latest/reliability-pillar-guide/).

- **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](#).

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. 155).

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.

Topics
• Offline (p. 91)
• Online (p. 92)
• Hybrid (p. 93)

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

The **offline** 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 offline approach minimizes downtime for your cluster, but it is the slowest of the three methods.

The basic process for offline 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

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.
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.

**Important**

AWS DMS does not currently support MongoDB 4.0 or later as a migration source.

## 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.
The following represents a migration to Amazon DocumentDB from an on-premises source using AWS 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. 97)
- Dumping Data (p. 97)
- Restoring Data (p. 97)
- Oplog Sizing (p. 98)
- AWS Database Migration Service Configuration (p. 98)
- Migrating from a Sharded Cluster (p. 98)

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.

97
Oplog Sizing

The MongoDB operations log (oplog) is a capped collection that contains all data modifications to your database. You can view the size of the oplog 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 oplog 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 Oplog in the MongoDB documentation. Determine the minimum required oplog 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 oplog 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 oplog 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. 302).

**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. 100)
- Identity and Access Management in Amazon DocumentDB (p. 110)
- Managing Amazon DocumentDB Users (p. 123)
- Restricting Database Access Using Role-Based Access Control (Built-In Roles) (p. 125)
- Logging and Monitoring in Amazon DocumentDB (p. 138)
- Updating Your Amazon DocumentDB TLS Certificates (p. 139)
- Compliance Validation in Amazon DocumentDB (p. 147)
- Resilience in Amazon DocumentDB (p. 147)
- Infrastructure Security in Amazon DocumentDB (p. 148)
- Auditing Amazon DocumentDB Events (p. 149)

Data Protection in Amazon DocumentDB

Amazon DocumentDB conforms to the AWS shared responsibility model, which includes regulations and guidelines for data protection. AWS is responsible for protecting the global infrastructure that runs all the AWS services. AWS maintains control over data hosted on this infrastructure, including the security configuration controls for handling customer content and personal data. AWS customers and APN Partners, acting either as data controllers or data processors, are responsible for any personal data that they put in the AWS Cloud.

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), so that 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.
- 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 Simple Storage Service (Amazon S3).

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form fields such as a **Name** field. This includes when you work with Amazon DocumentDB or other AWS services using the AWS Management Console, API, AWS CLI, or AWS SDKs. Any data that you enter into Amazon DocumentDB or other services might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credentials information in the URL to validate your request to that server.


**Topics**
- Encrypting Amazon DocumentDB Data at Rest (p. 101)
- Encrypting Data in Transit (p. 104)
- Key Management (p. 110)

### Encrypting Amazon DocumentDB Data at Rest

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 **customer master key** (CMK), Amazon DocumentDB asks AWS KMS to use your CMK to generate a ciphertext data key to encrypt the storage volume. The ciphertext data key is encrypted using the CMK 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 CMK 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 customer master key (CMK). Amazon DocumentDB creates a separate CMK for each AWS Region in your AWS account. For more information, see AWS Key Management Service Concepts.

To get started on creating your own CMK, see Getting Started in the AWS Key Management Service Developer Guide.

Important
You must use a symmetric CMK to encrypt your cluster as Amazon DocumentDB supports only symmetric CMKs. Do not use an asymmetric CMK to attempt to encrypt the data in your Amazon DocumentDB clusters. For more information, see Using Symmetric and Asymmetric Keys 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 cannot read from or write 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 CMK 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 CMK. For more information, see AWS Key Management Service Concepts.

Note
After you create an encrypted cluster, you can't change the CMK 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:

```bash
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:

```bash
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:

```bash
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:

```bash
aws docdb create-db-cluster ^
  --db-cluster-identifier sample-cluster ^
  --port 27017 ^
  --engine docdb ^
  --master-username yourMasterUsername ^
  --master-user-password yourMasterPassword ^
  --storage-encrypted ^
```
Note
After you create an encrypted cluster, you can't change the CMK 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 in two ways:
  1. Create a snapshot of the unencrypted cluster, and then restore the unencrypted snapshot as a new cluster while specifying the encryption at rest option.
  2. Create a snapshot of the unencrypted cluster, create an encrypted copy of the unencrypted snapshot, and then create a new cluster from the encrypted snapshot.

For more information, see the following topics:
- Creating a Manual Cluster Snapshot (p. 160)
- Restoring from a Cluster Snapshot (p. 162)
- Copying a Cluster Snapshot (p. 171)
- 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 CMK.

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. 323).

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. 261).
1. **Determine the cluster parameter group that your cluster is using.**
   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. In the list of clusters, choose the name of the cluster that you’re interested in.
   d. 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. 261)** — If you don’t have a custom cluster parameter group that you can use, create one.
      2. **Modifying an Amazon DocumentDB Cluster (p. 203)** — Modify your cluster to use the custom cluster parameter group.

2. **Determine the current value of the `tls` cluster parameter.**
   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.
   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. 261).

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. 261)](#) — If you don't have a custom cluster parameter group that you can use, create one.
   2. [Modifying an Amazon DocumentDB Cluster (p. 203)](#) — 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.

```bash
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).

```json
[
  [
    "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.

```bash
aws docdb modify-db-cluster-parameter-group
  --db-cluster-parameter-group-name custom3-6-param-grp
  --parameters
    ParameterName=tls
    ParameterValue=disabled
  --apply-method pending-reboot
```

Encrypting Data in Transit

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": {},
    "DBInstanceStatus": "rebooting",
    "DBSubnetGroup": {
      "Subnets": [
      {
        "SubnetStatus": "Active",
        "SubnetAvailabilityZone": { 
            "Name": "us-east-1a"
        },
        "SubnetIdentifier": "subnet-4e26d263"
      },
      {
        "SubnetStatus": "Active",
        "SubnetAvailabilityZone": { 
            "Name": "us-east-1a"
        },
        "SubnetIdentifier": "subnet-4e26d263"
      }
      
```
"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.corcjozrlsfc.us-east-1.docdb.amazonaws.com",  
    "Port": 27017,  
    "HostedZoneId": "Z2R2ITUGPM61AM"  
}
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. 110)
- Overview of Managing Access Permissions to Your Amazon DocumentDB Resources (p. 112)
- Managing Access Using Policies (p. 115)
- Using Identity-Based Policies (IAM Policies) for Amazon DocumentDB (p. 115)
- Amazon DocumentDB API Permissions: Actions, Resources, and Conditions Reference (p. 118)

Authentication

You can access AWS as any of the following types of identities:

- AWS account root user – When you first create an AWS account, you begin with a single 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, even the administrative ones. Instead, adhere to the best practice of using the root user only to create your first IAM user. Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.
• **IAM user** – An IAM user is an identity within your AWS account that has specific custom permissions (for example, permissions to create an instance in Amazon DocumentDB). You can use an IAM user name and password to sign in to secure AWS webpages like the AWS Management Console, AWS Discussion Forums, or the AWS Support Center.

In addition to a user name and password, you can also generate access keys for each user. You can use these keys when you access AWS services programmatically, either through one of the several SDKs or by using the AWS Command Line Interface (CLI). The SDK and CLI tools use the access keys to cryptographically sign your request. If you don't use AWS tools, you must sign the request yourself. Amazon DocumentDB supports Signature Version 4, a protocol for authenticating inbound API requests. For more information about authenticating requests, see Signature Version 4 Signing Process in the AWS General Reference.

• **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** – Instead of creating an IAM user, you can use existing identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as federated users. AWS assigns a role to a federated user when access is requested through an identity provider. For more information about federated users, see Federated Users and Roles in the IAM User Guide.

  • **AWS service access** – A service role is an IAM role that a service assumes to perform actions in your account on your behalf. When you set up some AWS service environments, you must define a role for the service to assume. This service role must include all the permissions that are required for the service to access the AWS resources that it needs. Service roles vary from service to service, but many allow you to choose your permissions as long as you meet the documented requirements for that service. Service roles provide access only within your account and cannot be used to grant access to services in other accounts. You can create, modify, and delete a service role from within IAM. For example, you can create a role that allows Amazon Redshift to access an Amazon S3 bucket on your behalf and then load data from that bucket into an Amazon Redshift cluster. 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. 112)
- Understanding Resource Ownership (p. 113)
- Managing Access to Resources (p. 113)
- Specifying Policy Elements: Actions, Effects, Resources, and Principals (p. 114)
- Specifying Conditions in a Policy (p. 115)

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.

<table>
<thead>
<tr>
<th>Resource Type</th>
<th>ARN Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance</td>
<td>arn:aws:rds:region:account-id:db:db-instance-name</td>
</tr>
<tr>
<td>Subnet group</td>
<td>arn:aws:rds:region:account-id:subgrp:subnet-group-name</td>
</tr>
</tbody>
</table>

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. 113)
- Resource-Based Policies (p. 114)

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.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCreateDBInstanceOnly",
      "Effect": "Allow",
      "Action": [
        "rds:CreateDBInstance"
      ],
      "Resource": [
        "arn:aws:rds:*:123456789012:db:test*",
        "arn:aws:rds:*:123456789012:pg:cluster-pg:default*",
        "arn:aws:rds:*:123456789012:subgrp:default"
      ]
    }
  ]
}
```

For more information about using identity-based policies with Amazon DocumentDB, see Using Identity-Based Policies (IAM Policies) for Amazon DocumentDB (p. 115). 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 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. 112)), 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. 118).

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. 112)
- Using Identity-Based Policies (IAM Policies) for Amazon DocumentDB (p. 115)

Using Identity-Based Policies (IAM Policies) for Amazon DocumentDB

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).

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.

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. 112).

The sections in this topic cover the following:

- Permissions Required to Use the Amazon DocumentDB Console (p. 116)
- AWS Managed (Predefined) Policies for Amazon DocumentDB (p. 117)
- Customer Managed Policy Examples (p. 117)

The following is an example of an IAM policy.
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. 118).

### 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. 117).

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. 116).

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. 117)
- Example 2: Prevent a User from Deleting an Instance (p. 118)

**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.

```json
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid":"AllowRDSDescribe",
            "Effect":"Allow",
            "Action": [" penny-mute-irrelevant-bug\n            "AmazonDocumentDB:Describe*"
        }
    ]
}
```
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.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyDelete1",
            "Effect": "Deny",
            "Action": "rds:DeleteDBInstance",
            "Resource": "arn:aws:rds:us-east-1:123456789012:db:my-db-instance"
        }
    ]
}
```

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. 115) 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. 115).

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. 119)
- Amazon DocumentDB Actions That Don't Support Resource-Level Permissions (p. 122)
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.

<table>
<thead>
<tr>
<th>Amazon DocumentDB API Operations and Actions</th>
<th>Resources</th>
<th>Condition Keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddTagsToResource</td>
<td>Instance</td>
<td>rds:db-tag</td>
</tr>
<tr>
<td>ApplyPendingMaintenanceAction</td>
<td>Instance</td>
<td>rds:db-tag</td>
</tr>
<tr>
<td>CopyDBClusterSnapshot</td>
<td>Cluster snapshot</td>
<td>rds:cluster-snapshot-tag</td>
</tr>
<tr>
<td>CreateDBCluster</td>
<td>Cluster</td>
<td>rds:cluster-tag</td>
</tr>
<tr>
<td>rds:CreateDBCluster</td>
<td>arn:aws:rds:region:account-id:cluster:db-cluster-name</td>
<td></td>
</tr>
<tr>
<td>CreateDBClusterParameterGroup</td>
<td>Cluster parameter group</td>
<td>rds:cluster-pg-tag</td>
</tr>
<tr>
<td>rds:CreateDBClusterParameterGroup</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Amazon DocumentDB API Operations and Actions</td>
<td>Resources</td>
<td>Condition Keys</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------</td>
<td>----------------</td>
</tr>
<tr>
<td>CreateDBClusterSnapshot</td>
<td>Cluster</td>
<td>rds:cluster-tag</td>
</tr>
<tr>
<td>rds:CreateDBClusterSnapshot</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cluster snapshot</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rds:cluster-snapshot:cluster-snapshot-name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CreateDBInstance</td>
<td>Instance</td>
<td>rds:DatabaseClass</td>
</tr>
<tr>
<td>rds:CreateDBInstance</td>
<td>Instance</td>
<td>rds:db-tag</td>
</tr>
<tr>
<td>Cluster</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rds:cluster:db-cluster-name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CreateDBSubnetGroup</td>
<td>Subnet group</td>
<td>rds:subgrp-tag</td>
</tr>
<tr>
<td>rds:CreateDBSubnetGroup</td>
<td>Subnet group</td>
<td></td>
</tr>
<tr>
<td>DeleteDBInstance</td>
<td>Instance</td>
<td>rds:db-tag</td>
</tr>
<tr>
<td>rds:DeleteDBInstance</td>
<td>Instance</td>
<td></td>
</tr>
<tr>
<td>DeleteDBSubnetGroup</td>
<td>Subnet group</td>
<td></td>
</tr>
<tr>
<td>rds:DeleteDBSubnetGroup</td>
<td>Subnet group</td>
<td></td>
</tr>
<tr>
<td>DescribeDBClusterParameterGroup</td>
<td>Cluster parameter group</td>
<td>rds:cluster-pg-tag</td>
</tr>
<tr>
<td>rds:DescribeDBClusterParameterGroup</td>
<td>Cluster parameter group</td>
<td></td>
</tr>
<tr>
<td>DescribeDBClusterParameterGroup</td>
<td>Cluster parameter group</td>
<td></td>
</tr>
<tr>
<td>rds:DescribeDBClusterParameterGroup</td>
<td>Cluster parameter group</td>
<td></td>
</tr>
<tr>
<td>DescribeDBClusters</td>
<td>Cluster</td>
<td>rds:cluster-tag</td>
</tr>
<tr>
<td>rds:DescribeDBClusters</td>
<td>Cluster</td>
<td></td>
</tr>
<tr>
<td>rds:db-cluster-instance-name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Amazon DocumentDB API Operations and Actions</td>
<td>Resources</td>
<td>Condition Keys</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------</td>
<td>----------------</td>
</tr>
<tr>
<td>DescribeDBClusterSnapshotAttributes</td>
<td>Cluster snapshot</td>
<td>rds:cluster-snapshot-tag</td>
</tr>
<tr>
<td>rds:DescribeDBClusterSnapshotAttributes</td>
<td>cluster-snapshot-id:cluster-snapshot:cluster-snapshot-name</td>
<td></td>
</tr>
<tr>
<td>DescribeDBSubnetGroups</td>
<td>Subnet group</td>
<td>rds:subgrp-tag</td>
</tr>
<tr>
<td>rds:DescribeDBSubnetGroups</td>
<td>rds:region:account-id:subgrp:subnet-group-name</td>
<td></td>
</tr>
<tr>
<td>DescribePendingMaintenanceActions</td>
<td>Instance</td>
<td>rds:DatabaseClass</td>
</tr>
<tr>
<td>rds:DescribePendingMaintenanceActions</td>
<td>account-id:db:db-instance-name</td>
<td>rds:db-tag</td>
</tr>
<tr>
<td>FailoverDBCluster</td>
<td>Cluster</td>
<td>rds:cluster-tag</td>
</tr>
<tr>
<td>rds:FailoverDBCluster</td>
<td>rds:region:account-id:cluster:db-cluster-instance-name</td>
<td></td>
</tr>
<tr>
<td>ListTagsForResource</td>
<td>Instance</td>
<td>rds:db-tag</td>
</tr>
<tr>
<td>rds:ListTagsForResource</td>
<td>rds:region:account-id:db:db-instance-name</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Subnet group</td>
<td>rds:subgrp-tag</td>
</tr>
<tr>
<td></td>
<td>rds:region:account-id:subgrp:subnet-group-name</td>
<td></td>
</tr>
<tr>
<td>ModifyDBCluster</td>
<td>Cluster</td>
<td>rds:cluster-tag</td>
</tr>
<tr>
<td>rds:ModifyDBCluster</td>
<td>rds:region:account-id:cluster:db-cluster-name</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cluster parameter group</td>
<td>rds:cluster-pg-tag</td>
</tr>
<tr>
<td></td>
<td>rds:region:account-id:cluster-pg:cluster-parameter-group-name</td>
<td></td>
</tr>
<tr>
<td>ModifyDBClusterParameterGroup</td>
<td>Cluster parameter group</td>
<td>rds:cluster-pg-tag</td>
</tr>
<tr>
<td>rds:ModifyDBClusterParameterGroup</td>
<td>rds:region:account-id:cluster-pg:cluster-parameter-group-name</td>
<td></td>
</tr>
<tr>
<td>ModifyDBClusterSnapshotAttributes</td>
<td>Cluster snapshot</td>
<td>rds:cluster-snapshot-tag</td>
</tr>
<tr>
<td>rds:ModifyDBClusterSnapshotAttributes</td>
<td>rds:region:account-id:cluster-snapshot:cluster-snapshot-name</td>
<td></td>
</tr>
<tr>
<td>Amazon DocumentDB API Operations and Actions</td>
<td>Resources</td>
<td>Condition Keys</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------</td>
<td>----------------</td>
</tr>
<tr>
<td>ModifyDBInstance</td>
<td>Instance</td>
<td>rds:DatabaseClass, rds:db-tag</td>
</tr>
<tr>
<td>rds:ModifyDBInstance arn:aws:rds:region:account-id:db:db-instance-name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RebootDBInstance</td>
<td>Instance</td>
<td>rds:db-tag</td>
</tr>
<tr>
<td>rds:RebootDBInstance arn:aws:rds:region:account-id:db:db-instance-name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RemoveTagsFromResource</td>
<td>Instance</td>
<td>rds:db-tag</td>
</tr>
<tr>
<td>Subnet group</td>
<td></td>
<td>rds:subgrp-tag</td>
</tr>
<tr>
<td>arn:aws:rds:region:account-id:subgrp:subnet-group-name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ResetDBClusterParameterGroup</td>
<td>Cluster parameter group</td>
<td>rds:cluster-pg-tag</td>
</tr>
<tr>
<td>RestoreDBClusterFromSnapshot</td>
<td>Cluster</td>
<td>rds:cluster-tag</td>
</tr>
<tr>
<td>Cluster snapshot</td>
<td></td>
<td>rds:cluster-snapshot-tag</td>
</tr>
<tr>
<td>RestoreDBClusterToPointInTime</td>
<td>Cluster</td>
<td>rds:cluster-tag</td>
</tr>
<tr>
<td>rds:RestoreDBClusterToPointInTime arn:aws:rds:region:account-id:cluster:db-cluster-instance-name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subnet group</td>
<td></td>
<td>rds:subgrp-tag</td>
</tr>
<tr>
<td>arn:aws:rds:region:account-id:subgrp:subnet-group-name</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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 (Built-In Roles) (p. 125).

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.

```javascript
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. 131).

```
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:
      [{role: "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 (Built-In Roles)` (p. 125).

**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 (Built-In Roles)**

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 specified databases. Amazon DocumentDB currently supports built-in roles that are scoped at the database level, such as `read`, `readWrite`, `readAnyDatabase`, and `clusterAdmin`.

Common use cases for RBAC include enforcing least privileges by creating users with read-only access to the databases in a cluster, and multi-tenant application designs that enable a single user to access a given database 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. 126)
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. 123).

- **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": "sample-user",
    "db": "admin",
    "roles": [
        {
            "db": "sample-database",
            "role": "read"
        }
    ]
}
```

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.

```javascript
db.createUser({user: "sample-user", pwd: "abc123", roles: ["readWrite"]})
```

Output from this operation looks something like the following.

```javascript
{
  "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. 131).

### Getting Started with RBAC

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 Built-In Roles (p. 133).

**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. 123).

```javascript
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"
    }
  ]
}
```

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.

```javascript
db.dropUser("user1")
```
```
```javascript
db.dropUser("user2")
```

To ensure that all users have the appropriate roles, you can list all users with the following command.

```bash
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"
      }
   ]
}
```

**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.
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.

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.

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.

```javascript
db.updateUser("readInFooBar", {roles: [{role: "read", db: "foo"}, {role: "read", db: "baz"}]}))
```

Example 5: Revoke access to a database for a user.

```javascript
db.revokeRolesFromUser("readInFooBar", [{role: "read", db: "baz"}])
```

Example 6: Describe a built-in role.

```javascript
db.getRole("read", {showPriviliges:true})
```

Output from this operation looks something like the following.

```json
{
  "role": "read",
  "db": "sample-database-1",
  "isBuiltIn": true,
  "roles": [],
  "inheritedRoles": [],
  "privileges": [
    {
      "resource": {
        "db": "sample-database-1",
        "collection": ""
      },
      "actions": [
        "changeStream",
        "collStats",
        "dbStats",
        "find",
        "killCursors",
        "listCollections",
        "listIndexes"
      ]
    }
  ],
  "inheritedPrivileges": [
    {
      "resource": {
        "db": "sample-database-1",
        "collection": ""
      },
      "actions": [
        "changeStream",
        "collStats",
        "dbStats",
        "find",
        "killCursors",
        "listCollections",
        "listIndexes"
      ]
    }
  ]
}
```

Example 7: Drop a user from the cluster.

```javascript
db.dropUser("readInFooBar")
```
Output from this operation looks something like the following.

```
true
```

## 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 is applied at the database level.

## Limits

The following table contains the limits for built-in roles in Amazon DocumentDB. Please note that Amazon DocumentDB does not currently support RBAC custom roles.

<table>
<thead>
<tr>
<th>Description</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of users per cluster</td>
<td>1000</td>
</tr>
<tr>
<td>Number of roles associated with a user</td>
<td>1000</td>
</tr>
</tbody>
</table>

## Built-In Roles

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.

<table>
<thead>
<tr>
<th>Role Type</th>
<th>Role Name</th>
<th>Description</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database User</td>
<td>read</td>
<td>Grants a user read access to the specified database.</td>
<td><code>changeStreams</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>collStats</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>dbStats</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>find</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>killCursors</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>listIndexes</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>listCollections</code></td>
</tr>
<tr>
<td></td>
<td>readWrite</td>
<td>Grants the user read and write access to the specified database.</td>
<td><code>createCollection</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>dropCollection</code></td>
</tr>
<tr>
<td>Role Type</td>
<td>Role Name</td>
<td>Description</td>
<td>Actions</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------------------</td>
<td>--------------------------------------------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>Cluster User</td>
<td>readAnyDatabase</td>
<td>Grants a user read access to all databases in the cluster.</td>
<td>createIndex, dropIndex, insert, killCursors, listIndexes, listCollections, remove, update</td>
</tr>
<tr>
<td></td>
<td>readWriteAnyDatabase</td>
<td>Grants a user read and write access to all databases in the cluster.</td>
<td>All actions from readWrite permissions, listChangeStreams, listDatabases</td>
</tr>
<tr>
<td></td>
<td>userAdminAnyDatabase</td>
<td>Grants a user the ability to assign and modify the roles or privileges any user has to the specified database.</td>
<td>changeCustomData, changePassword, createUser, dropRole, dropUser, grantRole, listDatabases, revokeRole, viewRole, viewUser</td>
</tr>
<tr>
<td>Role Type</td>
<td>Role Name</td>
<td>Description</td>
<td>Actions</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>dbAdminAnyDatabase</td>
<td>Grants a user the ability to perform database administration roles on any specified database.</td>
<td>All actions from dbAdmin permissions. dropCollection listDatabases listChangeStreams modifyChangeStreams</td>
</tr>
<tr>
<td>Superuser</td>
<td>root</td>
<td>Grants a user access to the resources and operations of all the following roles combined: readWriteAnyDatabase, dbAdminAnyDatabase, userAdminAnyDatabase, clusterAdmin, restore, and backup.</td>
<td>All actions from readWriteAnyDatabase, dbAdminAnyDatabase, userAdminAnyDatabase, clusterAdmin, restore, and backup.</td>
</tr>
<tr>
<td>Database Administration</td>
<td>dbAdmin</td>
<td>Grants a user the ability to perform administrative tasks on the specified database.</td>
<td>collMod collStats createCollection createIndex dropCollection dropDatabase dropIndex dbStats find killCursors listIndexes listCollections modifyChangeStreams</td>
</tr>
<tr>
<td>Role Type</td>
<td>Role Name</td>
<td>Description</td>
<td>Actions</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>dbOwner</td>
<td>Grants a user the ability to perform any administrative tasks on the specified database by combining the roles dbAdmin and readWrite.</td>
<td>All actions from dbAdmin and readWrite.</td>
</tr>
<tr>
<td>Cluster Administration</td>
<td>clusterAdmin</td>
<td>Grants a user the greatest cluster management access by combining the clusterManager, clusterMonitor, and hostManager roles.</td>
<td>All actions from clusterManager, clusterMonitor, and hostManager. listChangeStreams dropDatabase modifyChangeStreams</td>
</tr>
<tr>
<td></td>
<td>clusterManager</td>
<td>Grants a user the ability to take management and monitoring actions on the specified cluster.</td>
<td>listChangeStreams listSessions modifyChangeStreams replSetGetConfig</td>
</tr>
<tr>
<td></td>
<td>clusterMonitor</td>
<td>Grants a user the ability to have read-only access to monitoring tools.</td>
<td>collStats dbStats find getParameter hostInfo indexStats killCursors listChangeStreams listCollections listDatabases listIndexes listSessions replSetGetConfig serverStatus serverStatus top</td>
</tr>
<tr>
<td>Role Type</td>
<td>Role Name</td>
<td>Description</td>
<td>Actions</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td></td>
<td>hostManager</td>
<td>Grants a user the ability to monitor and manage servers.</td>
<td>killCursors</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>killAnyCursor</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>killAnySession</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>killop</td>
</tr>
<tr>
<td>Backup Administration</td>
<td>backup</td>
<td>Grants a user the access needed to back up data.</td>
<td>getParameter</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>insert</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>find</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>listChangeStreams</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>listCollections</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>listDatabases</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>listIndexes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>update</td>
</tr>
</tbody>
</table>
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. 302).

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. 316).

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).

<table>
<thead>
<tr>
<th>Role Type</th>
<th>Role Name</th>
<th>Description</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>restore</td>
<td>Grants a user the access needed to restore data.</td>
<td>changeCustomData changePassword collMod createCollection createIndex createUser dropCollection dropRole dropUser getPassword grantRole find insert listCollections modifyChangeStreams revokeRole remove viewRole viewUser update</td>
</tr>
</tbody>
</table>
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. 315).

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. 149).

### 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. 139)
- **Step 2: Update the Server Certificate** (p. 140)

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.

```
```
Next, update your applications to use the new certificate bundle. The new CA bundle contains both the old CA certificate (`rds-ca-2015-root.pem`) 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. 144) 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. 104) and Connecting with TLS Enabled (p. 325).

**Note**
Currently, the MongoDB Go Driver 1.2.1 only accepts one CA server certificate in `sslcertificateauthorityfile`. Please see Connecting with TLS Enabled (p. 325) 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.

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. If your intention is to modify the server certificate immediately, use the **Apply Immediately** option.
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.

```bash
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.

```bash
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. 231).

- **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. 144).

- **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. 142).

- **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. 323).

- **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. 256).
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 March 5, 2020?

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 March 5, 2020. You must update your applications and clusters to use the new CA certificates before or after March 5, 2020.

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

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

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.

   ```bash
   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.

   ```bash
   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. 144)

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:

```bash
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:

```bash
shasum rds-combined-ca-bundle.pem
```

Output:

```
a4ded73667097aa2d97d28a469f1fec94912a166 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:

```
shai3sum rds-combined-ca-bundle.pem
```

Output:

```
a4ded73667097aa2d97d28a469f1fec94912a166 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:
   a4ded73667097aa2d97d28a469f1fec94912a166
```

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

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
```

146
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. 155).

**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. 166).

---

**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.

- 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), 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. 149)
• Enabling Auditing (p. 150)
• Disabling Auditing (p. 152)
• Accessing Your Audit Events (p. 154)

Supported Events

Amazon DocumentDB auditing supports the following event categories: connection, data definition language (DDL), user management, and authorization. The event types are as follows.

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>authenticate</td>
<td>Connection</td>
<td>Successful or failed authentication attempts on a new connection.</td>
</tr>
<tr>
<td>createDatabase</td>
<td>DDL</td>
<td>Creation of a new database.</td>
</tr>
</tbody>
</table>
Enabling Auditing

Enabling auditing on a cluster is a two-step process. Ensure that both steps are completed, or auditing logs will not be sent to CloudWatch Logs.

**Step 1. Enable the audit_logs Cluster Parameter**

To set the audit_logs parameter to enabled, create a new cluster parameter group or use an existing custom parameter group and modify it by setting audit_logs to enabled. You cannot modify a default parameter group.

For more information, see the following:

- Creating Amazon DocumentDB Cluster Parameter Groups (p. 261)

  After creating a custom parameter group, modify it by changing the audit_logs parameter value to enabled.

- Modifying Amazon DocumentDB Cluster Parameter Groups (p. 262)

**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. 191), see *Create a Cluster: Additional Configurations* (step 5, Log exports)
- **When modifying an existing cluster** — Using the AWS Management Console (p. 203)
- **When performing a cluster snapshot restore** — Restore from a Cluster Snapshot Using the AWS Management Console (p. 162), step 9.
- **When performing a point-in-time restore** — Using the AWS Management Console (p. 166) (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:

```bash
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:

```bash
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:

```bash
aws docdb modify-db-cluster \
  --db-cluster-identifier sample-cluster \
  --cloudwatch-logs-export-configuration '{"EnableLogTypes":["audit"]}'
```

For Windows:

```bash
aws docdb modify-db-cluster ^
  --db-cluster-identifier sample-cluster ^
  --cloudwatch-logs-export-configuration "{"EnableLogTypes":["audit"]}"
```
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": "ZNKXH85TT8WW7",
      "StorageEncrypted": false,
      "DBClusterParameterGroup": "default.docdb3.6",
      "MasterUsername": "<user-name>",
      "BackupRetentionPeriod": 1,
      "Port": 27017,
      "VpcSecurityGroups": [
         { "Status": "active",
           "VpcSecurityGroupId": "sg-77186ed0" }
      ],
      "Status": "creating",
      "Engine": "docdb",
      "EngineVersion": "3.6.0",
      "MultiAZ": false,
      "AvailabilityZones": [ "us-east-1a", "us-east-1c", "us-east-1f" ],
      "DBSubnetGroup": "default",
      "DBClusterMembers": [],
      "ReaderEndpoint": "sample-cluster.cluster-ro-corcjozrlsfc.us-east-1.docdb.amazonaws.com",
      "EnabledCloudwatchLogsExports": [ "audit" ],
      "PreferredMaintenanceWindow": "wed:03:08-wed:03:38",
      "AssociatedRoles": [],
      "ClusterCreateTime": "2019-02-13T16:35:04.756Z",
      "DbClusterResourceId": "cluster-YOS52CUXGDTNKDQ7DH72I4LED4",
      "Endpoint": "sample-cluster.cluster-corcjozrlsfc.us-east-1.docdb.amazonaws.com",
      "PreferredBackupWindow": "07:16-07:46",
      "DBClusterIdentifier": "sample-cluster"
   }
}
```

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

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:

```bash
aws docdb modify-db-cluster \
  --db-cluster-identifier sample-cluster \n  --cloudwatch-logs-export-configuration '{"DisableLogTypes":["audit"]}'
```

For Windows:

```bash
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).

```json
{
  "DBCluster": {
    "DBClusterParameterGroup": "default.docdb3.6",
    "HostedZoneId": "ZNKXH85TT8GW",
    "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",
    "Endpoint": "sample-cluster2.cluster-corcjozrlsfc.us-east-1.docdb.amazonaws.com",
    "ReaderEndpoint": "sample-cluster2.cluster-ro-corcjozrlsfc.us-east-1.docdb.amazonaws.com",
    "BackupRetentionPeriod": 1,
    "EngineVersion": "3.6.0",
  }
}
```
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. 203)
- Modifying Amazon DocumentDB Cluster Parameter Groups (p. 262)

Accessing Your Audit Events

Use following steps to access your audit events on Amazon 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.
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.

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. 155)
- Comparing Automatic and Manual Snapshots (p. 156)
- Cluster Snapshot Considerations (p. 157)
- Understanding Backup Storage Usage (p. 159)
- Creating a Manual Cluster Snapshot (p. 160)
- Restoring from a Cluster Snapshot (p. 162)
- Restoring to a Point in Time (p. 166)
- Copying a Cluster Snapshot (p. 171)
- Sharing Amazon DocumentDB Cluster Snapshots (p. 174)
- Deleting a Cluster Snapshot (p. 178)

## Back Up and Restore: Concepts

<table>
<thead>
<tr>
<th>Noun</th>
<th>Description</th>
<th>APIs (Verbs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backup retention period</td>
<td>A period of time between 1 and 35 days for which you can perform a point-in-time restore.</td>
<td>create-db-cluster</td>
</tr>
<tr>
<td></td>
<td></td>
<td>modify-db-cluster</td>
</tr>
<tr>
<td></td>
<td></td>
<td>restore-db-cluster-to-point-in-time</td>
</tr>
<tr>
<td>Amazon DocumentDB storage volume</td>
<td>Highly available, Multi-AZ storage for Amazon DocumentDB clusters.</td>
<td>create-db-cluster</td>
</tr>
<tr>
<td></td>
<td></td>
<td>delete-db-cluster</td>
</tr>
<tr>
<td>Backup window</td>
<td>Period of time in the</td>
<td>create-db-cluster</td>
</tr>
</tbody>
</table>
Comparing Automatic and Manual Snapshots

The following are key features of Amazon DocumentDB (with MongoDB compatibility) automatic and manual snapshots.

**Automatic snapshot features**

Amazon DocumentDB automatic snapshots have the following key features:

- **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.

- **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*.

<table>
<thead>
<tr>
<th>Noun</th>
<th>Description</th>
<th>APIs (Verbs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic snapshot</td>
<td>Daily snapshots that are full backups of cluster and are automatically created by the continuous backup process in Amazon DocumentDB.</td>
<td>restore-db-cluster-from-snapshot, describe-db-cluster-snapshot-attributes, describe-db-cluster-snapshots</td>
</tr>
<tr>
<td>Manual snapshot</td>
<td>Snapshots you create manually to retain full backups of a cluster beyond the backup period.</td>
<td>create-db-cluster-snapshot, copy-db-cluster-snapshot, delete-db-cluster-snapshot, describe-db-cluster-snapshot-attributes, describe-db-cluster-snapshots, modify-db-cluster-snapshot-attribute</td>
</tr>
</tbody>
</table>
• **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.

• **Deleting automatic snapshots**—Automatic snapshots are deleted when you delete the automatic snapshot's cluster. You can't manually delete an automatic snapshot.

• **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.

**Manual snapshot features**

Amazon DocumentDB manual snapshots have the following key features:

• **Created on demand**—Amazon DocumentDB manual snapshots are created on demand using the Amazon DocumentDB console or AWS CLI.

• **Full backups**—When a manual snapshot is taken, a full backup of your cluster's data is created and stored.

• **Restoring from a manual snapshot**—You can restore from a 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.

• **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.

• **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.

• **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.

• **Service Quotas**—You are limited to a maximum of 100 manual snapshots per AWS Region.

**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 follow these rules:

• Your cluster must be in the available state for an automatic snapshot to be taken.

• 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. For more information about manually creating a cluster snapshot, see Creating a Manual Cluster Snapshot (p. 160).
You can perform the following operations with your snapshots:

- Copy both automatic and manual snapshots. For more information, see Copying a Cluster Snapshot (p. 171).

## Snapshot Storage

Your Amazon DocumentDB snapshot storage for each AWS Region is composed of the automated cluster snapshots and manual cluster snapshots for that Region. Your snapshot storage is equivalent to the sum of the database storage for all instances in that Region.

For more information about backup storage costs, see Amazon DocumentDB Pricing.

When you delete a cluster, all of its automatic snapshots are deleted and cannot be recovered. However, if you choose to have Amazon DocumentDB create a final snapshot before your cluster is deleted, you can use the final snapshot to recover your cluster.

Manual snapshots are not deleted when you delete a cluster.

For more information on snapshots and storage, see Understanding Backup Storage Usage (p. 159).

## 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. The following table lists the time blocks for each Region from which the default backups windows are assigned.

<table>
<thead>
<tr>
<th>Region</th>
<th>UTC Time Block</th>
</tr>
</thead>
<tbody>
<tr>
<td>US East (Ohio)</td>
<td>03:00-11:00</td>
</tr>
<tr>
<td>US East (N. Virginia)</td>
<td>03:00-11:00</td>
</tr>
<tr>
<td>US West (Oregon)</td>
<td>06:00-14:00</td>
</tr>
<tr>
<td>Asia Pacific (Mumbai)</td>
<td>17:30–01:30</td>
</tr>
<tr>
<td>Asia Pacific (Seoul)</td>
<td>13:00-21:00</td>
</tr>
<tr>
<td>Asia Pacific (Singapore)</td>
<td>14:00–22:00</td>
</tr>
<tr>
<td>Asia Pacific (Sydney)</td>
<td>12:00–20:00</td>
</tr>
<tr>
<td>Asia Pacific (Tokyo)</td>
<td>13:00-21:00</td>
</tr>
<tr>
<td>Canada (Central)</td>
<td>22:00-06:00</td>
</tr>
<tr>
<td>Europe (Frankfurt)</td>
<td>23:00-07:00</td>
</tr>
<tr>
<td>Europe (Ireland)</td>
<td>22:00-06:00</td>
</tr>
<tr>
<td>Europe (Paris)</td>
<td>22:00-06:00</td>
</tr>
</tbody>
</table>
You can change your preferred backup window by modifying the cluster. Use the AWS Management Console or AWS CLI to change the backup window. For more information, see Modifying an Amazon DocumentDB Cluster (p. 203).

**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 console or the AWS CLI. For more information, see Modifying an Amazon DocumentDB Cluster (p. 203).

**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. 155). 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 you 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).
Creating a Manual Cluster Snapshot

The amount of time it takes to create a snapshot varies with the size 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.

You can create a manual snapshot using either the AWS Management Console or AWS CLI.

Using the AWS Management Console

There are two different ways to create a manual snapshot using the console.

To manually create a snapshot (method 1)

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. In the list of clusters, choose the cluster that you want to create a snapshot of.
   b. In the Snapshot name box, enter a name for your snapshot.
   c. Choose Create.

To manually create a snapshot (method 2)

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. On the Actions menu, choose Take snapshot.
5. On the Create cluster snapshot page:
   a. In the Snapshot name box, enter a name for your snapshot.
b. Choose Create.

## 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.

In the following example, you create a snapshot named `sample-cluster-snapshot` for a cluster named `sample-cluster`.

For Linux, macOS, or Unix:

```bash
aws docdb create-db-cluster-snapshot \
  --db-cluster-identifier sample-cluster \
  --db-cluster-snapshot-identifier sample-cluster-snapshot
```

For Windows:

```bash
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 (JSON format).

```json
{
  "DBClusterSnapshot": {
    "VpcId": "vpc-91280df6",
    "Port": 0,
    "AvailabilityZones": [
      "us-east-1a",
      "us-east-1b",
      "us-east-1c"
    ],
    "IAMDatabaseAuthenticationEnabled": false,
    "AllocatedStorage": 1,
    "PercentProgress": 0,
    "ClusterCreateTime": "2018-06-26T17:32:07.729Z",
    "StorageEncrypted": false,
    "Engine": "docdb",
    "SnapshotType": "manual",
    "Status": "creating",
    "MasterUsername": "<user-name>",
    "LicenseModel": "n/a",
    "DBClusterSnapshotIdentifier": "sample-cluster-snapshot",
    "DBClusterIdentifier": "sample-cluster",
    "EngineVersion": "3.6.0"
  }
}
```
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.

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.

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 DML logging to Amazon CloudWatch Logs.
  - **Disabled** — Prevents your cluster from exporting DML 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**.

### 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. 474)`.
• --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.

```json
{
    "DBCluster": {
        "AvailabilityZones": [
            "us-east-1c",
            "us-east-1b",
            "us-east-1a"
        ],
        "BackupRetentionPeriod": 1,
        "DBClusterIdentifier": "sample-cluster-restore",
        "DBClusterParameterGroup": "default.docdb3.6",
        "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": "3.6.0",
"Port": 27017,
"MasterUsername": "<master-user>",
"PreferredBackupWindow": "02:00-02:30",
"PreferredMaintenanceWindow": "tue:09:50-tue:10:20",
"DBClusterMembers": [],
"VpcSecurityGroups": [
    {
      "VpcSecurityGroupId": "sg-abcdefgh",
      "Status": "active"
    }
],
"HostedZoneId": "ABCDEFGHIJKLM",
"StorageEncrypted": true,
"KmsKeyId": "arn:aws:kms:us-east-1:<accountID>:key/<sample-key-id>",
"DbClusterResourceId": "cluster-ABCDEFGHIJKLMNOPQRSTUVWXYZ",
"DBClusterArn": "arn:aws:rds:us-east-1:<accountID>:cluster:sample-cluster-restore",
"AssociatedRoles": [],
"ClusterCreateTime": "2020-04-01T01:43:40.871Z",
"DeletionProtection": true
}

After the cluster status is available, create at least one instance for the cluster.

For Linux, macOS, or Unix:

```bash
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:

```bash
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.

```json
{
  "DBInstance": {
    "DBInstanceIdentifier": "sample-cluster-restore-instance",
    "DBInstanceClass": "db.r5.large",
    "Engine": "docdb",
    "DBInstanceStatus": "creating",
    "PreferredBackupWindow": "02:00-02:30",
    "BackupRetentionPeriod": 1,
    "VpcSecurityGroups": [
      {
        "VpcSecurityGroupId": "sg-abcdefg",
        "Status": "active"
      }
    ]
  }
}
```
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).

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. 203).

Using the AWS Management Console

You can restore a cluster to a point-in-time within its backup retention period using the console.

To restore a cluster to a point-in-time

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.
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.

   ![ Restore time section diagram ]

   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.

   ![ Configuration section diagram ]

   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.
b. **Instance class**—In the list, choose the instance class that you want for the cluster's instances.

c. **Number of instances**—In the 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 those sections. Otherwise continue with the next step.

- **Network settings**

1. **Virtual Private Cloud (VPC)**—In the list, choose the VPC that you want to use for this cluster.

2. **Subnet group**—In the list, choose the subnet group for this cluster.

3. **VPC security groups**—In the list, choose the VPC security groups for the 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. **Export auditing logs to Amazon CloudWatch**—Choose one of the following:
   - **Enabled**—Enables your cluster to export DML logging to Amazon CloudWatch Logs.
   - **Disabled**—Prevents your cluster from exporting DML logs to Amazon CloudWatch Logs. **Disabled** is the default.
2. **IAM role**—From the list, choose *RDS Service Linked Role*.

7. Complete the process.
   - To restore the cluster, choose **Create cluster**.
   - To cancel the operation, choose **Cancel**.

## 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.

### 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, Amazon Neptune, and Amazon DocumentDB) per AWS account, per Region.

- **--restore-to-time**—The UTC date and time to restore the cluster to.
  
  **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-20181102182833-snapshot** to the latest restorable time.

For Linux, macOS, or Unix:
For Windows:

```bash
aws docdb restore-db-cluster-to-point-in-time
  --db-cluster-identifier sample-cluster-restored
  --source-db-cluster-identifier sample-cluster-20181102182833-snapshot
  --use-latest-restorable-time
```

Example

The following example creates `sample-cluster-restored` from the snapshot `sample-cluster-snapshot` to the 03:15 on December 11, 2018 (UTC), which is within the backup retention period of `sample-cluster-2018-12-02-00-07`.

For Linux, macOS, or Unix:

```bash
aws docdb restore-db-cluster-to-point-in-time
  --db-cluster-identifier sample-cluster-restored
  --source-db-cluster-identifier sample-cluster-2018-12-02-00-07
  --restore-type full-copy
  --restore-to-time 2018-12-11T03:15:00Z
```

For Windows:

```bash
aws docdb restore-db-cluster-to-point-in-time
  --db-cluster-identifier sample-cluster-restored
  --source-db-cluster-identifier sample-cluster-2018-12-02-00-07
  --restore-type full-copy
  --restore-to-time 2018-12-11T03:15:00Z
```

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

```json
{
  "DBCluster": {
    "PreferredMaintenanceWindow": "wed:06:28-wed:06:58",
    "ClusterCreateTime": "2018-12-11T18:21:32.473Z",
    "DBSubnetGroup": "default",
    "ReaderEndpoint": "sample-cluster-restored.cluster-ro-dvuterjxdl57.us-east-1.docdb.amazonaws.com",
    "Status": "creating",
    "MasterUsername": "<user-name>",
    "IAMDatabaseAuthenticationEnabled": false,
    "Engine": "docdb",
    "HostedZoneId": "Z2SUY0A1719RZT",
    "EngineVersion": "3.6.0",
    "VpcSecurityGroups": [
      {
        "VpcSecurityGroupId": "sg-77186e0d",
        "Status": "active"
      }
    ],
    "ReadReplicaIdentifiers": []
  }
}```
Copying a Cluster Snapshot

You can copy an Amazon DocumentDB (with MongoDB compatibility) automatic or manual snapshot to another account in the same region using the AWS Management Console or AWS CLI. Amazon DocumentDB does not currently support the ability to copy snapshots across regions.

Using the AWS Management Console

To make a copy of a snapshot

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. Locate the snapshot that you want to copy, and then choose the button to the left of its name.
4. On the Actions menu, choose Copy.
5. Complete the Settings section.
a. **New snapshot identifier**—Enter a name for the new snapshot.

<table>
<thead>
<tr>
<th>Target Snapshot Naming Constraints:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Cannot be the name of an existing snapshot.</td>
</tr>
<tr>
<td>• The target snapshot must be in the same AWS Region as the source and be a valid snapshot identifier.</td>
</tr>
<tr>
<td>• Length is [1–63] letters, numbers, or hyphens.</td>
</tr>
<tr>
<td>• First character must be a letter.</td>
</tr>
<tr>
<td>• Cannot end with a hyphen or contain two consecutive hyphens.</td>
</tr>
<tr>
<td>• Must be unique for all clusters (across Amazon RDS, Amazon Neptune, and Amazon DocumentDB) per AWS account, per Region.</td>
</tr>
</tbody>
</table>

b. **Copy tags**—To copy any tags you have on your source snapshot to your snapshot copy, choose Copy tags.

6. 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**—In the 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>`.

7. Choose one of the following:

   • To make a copy of the selected snapshot, choose **Copy snapshot**.

   • To not make a copy of the snapshot, choose **Cancel**.
Using the AWS CLI

To make a copy of a cluster snapshot, use the `copy-db-cluster-snapshot` operation with the following parameters.

**Parameters**

- **--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. This parameter is not case sensitive.

  **Source Snapshot Naming Constraints:**
  
  - Must specify a valid system snapshot in the *available* state.
  - The source snapshot must be in the same AWS Region as the copy and be a valid snapshot identifier.
    - 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.

- **--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.
  - The target snapshot must be in the same AWS Region as the source and be a valid snapshot identifier.
    - 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.

**Example**

The following AWS CLI example makes a copy of `sample-cluster-snapshot` named `sample-cluster-snapshot-copy`.

For Linux, macOS, or Unix:

```bash
aws docdb copy-db-cluster-snapshot \
    --source-db-cluster-snapshot-identifier sample-cluster-snapshot \
    --target-db-cluster-snapshot-identifier sample-cluster-snapshot-copy
```

For Windows:

```bash
aws docdb copy-db-cluster-snapshot ^
    --source-db-cluster-snapshot-identifier sample-cluster-snapshot ^
    --target-db-cluster-snapshot-identifier sample-cluster-snapshot-copy
```
Output from this operation looks something like the following (JSON format).

```json
{
  "DBClusterSnapshot": {
    "AvailabilityZones": [
      "us-east-1a",
      "us-east-1b",
      "us-east-1c",
      "us-east-1d",
      "us-east-1e",
      "us-east-1f"
    ],
    "VpcId": "vpc-91280df6",
    "ClusterCreateTime": "2018-11-05T21:01:18.946Z",
    "StorageEncrypted": true,
    "MasterUsername": "user-name",
    "Port": 0,
    "EngineVersion": "3.6.0",
    "Engine": "docdb",
    "SnapshotCreateTime": "2018-11-05T21:25:34.841Z",
    "DBClusterSnapshotIdentifier": "sample-cluster-snapshot-copy",
    "DBClusterIdentifier": "sample-cluster",
    "PercentProgress": 0,
    "KmsKeyId": "arn:aws:kms:us-east-1:012345678901:key/EXAMPLE-KMS-KEY-ID",
    "Status": "copying",
    "SnapshotType": "manual"
  }
}
```

### Sharing Amazon DocumentDB Cluster Snapshots

Sharing a manual snapshot using Amazon DocumentDB enables the following actions:

- When you share a manual snapshot, whether encrypted or unencrypted, authorized AWS accounts can then copy the snapshot.

- When you share 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 the copy.

**Note**

To share an automated snapshot, create a manual snapshot by copying the automated snapshot, and then share that copy. In this topic, the terms *manual snapshot* and *snapshot* are used interchangeably unless otherwise noted.

For more information about copying a snapshot, see [Copying a Cluster Snapshot (p. 171)](#). For more information about restoring a cluster from a snapshot, see [Restoring from a Cluster Snapshot (p. 162)](#).

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. Take care when sharing a snapshot as public so that none of your private information is included in any of your public snapshots.
The following limitation applies when sharing manual snapshots with other AWS accounts:

- When 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

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 account by adding the other account 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 Allowing Access to an AWS KMS Encryption Key (p. 175) later in this topic.

2. Use the AWS CLI to share the encrypted snapshot with the other accounts.

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.

## 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 \texttt{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 \texttt{kms:CreateGrant} action.

```json
{
    "Id": "key-policy-1",
    "Version": "2012-10-17",
    "Statement": [
```
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.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowUseOfTheKey",
            "Effect": "Allow",
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey",
                "kms:CreateGrant",
                "kms:RetireGrant"
            ],
            "Resource": ["arn:aws:kms:us-west-2:123451234512:key/c989c1dd-a3f2-4a5d-8d96-e793d082ab26"]
        },
        {
            "Sid": "AllowAttachmentOfPersistentResources",
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": "*",
            "Condition": {"Bool": {"kms:GrantIsForAWSResource": true}}
        }
    ]
}
```
Sharing a Snapshot

You can share a snapshot using the AWS CLI.

Using the AWS CLI

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:

```bash
aws docdb modify-db-snapshot-attribute \
   --db-snapshot-identifier manual-snapshot1 \
   --attribute-name restore \
   --values-to-add '["123451234512","123456789012"]'
```

For Windows:

```bash
aws docdb modify-db-snapshot-attribute ^
   --db-snapshot-identifier manual-snapshot1 ^
   --attribute-name restore ^
   --values-to-add '["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:

```bash
aws docdb modify-db-snapshot-attribute \
   --db-snapshot-identifier manual-snapshot1 \
   --attribute-name restore \
   --values-to-remove '["123456789012"]'
```

For Windows:

```bash
aws docdb modify-db-snapshot-attribute ^
```
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.

Using the AWS Management Console

To delete a manual cluster snapshot using the console

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.
4. On 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.

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

```json
{
    "DBClusterSnapshot": {
        "DBClusterIdentifier": "sample-cluster",
        "Port": 0,
        "KmsKeyId": "arn:aws:kms:us-east-1:012345678901:key/EXAMPLE-KMS-KEY-ID",
        "Status": "available",
        "SnapshotCreateTime": "2018-11-05T21:25:34.841Z",
        "MasterUsername": "user-name",
        "PercentProgress": 100,
        "ClusterCreateTime": "2018-11-05T21:01:18.946Z",
        "DBClusterSnapshotIdentifier": "sample-cluster-snapshot",
        "Engine": "docdb",
        "EngineVersion": "3.6.0",
        "VpcId": "vpc-91280df6",
        "SnapshotType": "manual",
        "AvailabilityZones": ["us-east-1a", "us-east-1b", "us-east-1c", "us-east-1d", "us-east-1e", "us-east-1f"],
        "StorageEncrypted": true
    }
}
```
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. 180)
- Managing Amazon DocumentDB Clusters (p. 185)
- Managing Amazon DocumentDB Instances (p. 217)
- Managing Amazon DocumentDB Subnet Groups (p. 236)
- Amazon DocumentDB High Availability and Replication (p. 245)
- Managing Amazon DocumentDB Events (p. 250)
- Choosing Regions and Availability Zones (p. 254)
- Managing Amazon DocumentDB Cluster Parameter Groups (p. 256)
- Understanding Amazon DocumentDB Cluster Parameter Groups (p. 256)
- Understanding Amazon DocumentDB Endpoints (p. 279)
- Understanding Amazon DocumentDB Amazon Resource Names (ARNs) (p. 285)
- Tagging Amazon DocumentDB Resources (p. 288)
- Maintaining Amazon DocumentDB (p. 292)
- Understanding Service-Linked Roles (p. 298)

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. 180)
- Describing Clusters and Instances (p. 181)
- Creating a Cluster Snapshot (p. 182)
- Restoring from a Snapshot (p. 183)
- Removing an Instance from a Cluster (p. 184)
- Deleting a Cluster (p. 184)

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.

```bash
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 ???(p. 189).

```bash
aws docdb describe-db-clusters --filter Name=engine,Values=docdb
```

Output from this operation looks something like the following.

```json
{
  "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": [
```
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):
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.

```bash
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.

```bash
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`.

```bash
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.

```json
[
    "sample-cluster-snapshot",
    "manual",
    "available"
],
[
    "rds:sample-cluster",
    "automated",
    "available"
]
```

The following example shows how to restore an Amazon DocumentDB cluster from a snapshot.

```bash
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[*,DBInstanceStatus]'
```

### 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. 184).

```
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:
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. 185)
- Amazon DocumentDB Cluster Settings (p. 187)
- Determining a Cluster's Status (p. 188)
- Amazon DocumentDB Cluster Lifecycle (p. 189)
- Managing Performance and Scaling for Amazon DocumentDB Clusters (p. 216)
- Understanding Amazon DocumentDB Cluster Fault Tolerance (p. 216)

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.

<table>
<thead>
<tr>
<th>Noun</th>
<th>Description</th>
<th>API Operations (Verbs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster</td>
<td>Consists of one or more instances and a cluster storage volume that manages the data for those instances.</td>
<td>create-db-cluster</td>
</tr>
<tr>
<td></td>
<td></td>
<td>delete-db-cluster</td>
</tr>
<tr>
<td></td>
<td></td>
<td>describe-db-clusters</td>
</tr>
<tr>
<td></td>
<td></td>
<td>modify-db-cluster</td>
</tr>
<tr>
<td>Instance</td>
<td>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.</td>
<td>create-db-instance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>delete-db-instance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>describe-db-instances</td>
</tr>
<tr>
<td></td>
<td></td>
<td>modify-db-instance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>describe-orderable-db-instance-options</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reboot-db-instance</td>
</tr>
</tbody>
</table>

aws docdb delete-db-cluster \
--db-cluster-identifier sample-cluster \
--skip-final-snapshot
<table>
<thead>
<tr>
<th>Noun</th>
<th>Description</th>
<th>API Operations (Verbs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster volume</td>
<td>A virtual database storage volume that spans three Availability Zones, with each Availability Zone having two copies of the cluster data.</td>
<td>N/A</td>
</tr>
<tr>
<td>Primary instance</td>
<td>Supports both read and write operations, and performs all data modifications to the cluster volume. Each cluster has one primary instance.</td>
<td>N/A</td>
</tr>
<tr>
<td>Replica instance</td>
<td>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.</td>
<td>N/A</td>
</tr>
<tr>
<td>Cluster endpoint</td>
<td>An endpoint for an Amazon DocumentDB cluster that connects to the current primary instance for the cluster. Each Amazon DocumentDB cluster has a cluster endpoint and one primary instance.</td>
<td>N/A</td>
</tr>
<tr>
<td>Reader endpoint</td>
<td>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.</td>
<td>N/A</td>
</tr>
<tr>
<td>Instance endpoint</td>
<td>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.</td>
<td>N/A</td>
</tr>
</tbody>
</table>
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. 256).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Modifiable</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DBClusterIdentifier</strong></td>
<td>Yes</td>
<td>Naming constraints:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Length is [1—63] letters, numbers, or hyphens.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• First character must be a letter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Cannot end with a hyphen or contain two consecutive hyphens.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Must be unique for all clusters across Amazon RDS, Amazon Neptune, and Amazon DocumentDB per AWS account, per Region.</td>
</tr>
<tr>
<td><strong>Engine</strong></td>
<td>No</td>
<td>Must be docdb.</td>
</tr>
<tr>
<td><strong>BackupRetentionPeriod</strong></td>
<td>Yes</td>
<td>Must be between [1-35] days.</td>
</tr>
<tr>
<td><strong>DBClusterParameterGroupName</strong></td>
<td>Yes</td>
<td>Naming constraints:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Length is [1—255] alphanumeric characters.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• First character must be a letter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Cannot end with a hyphen or contain two consecutive hyphens.</td>
</tr>
<tr>
<td><strong>DBSubnetGroupName</strong></td>
<td>No</td>
<td>After a cluster has been created, you cannot modify the cluster's subnet.</td>
</tr>
<tr>
<td><strong>EngineVersion</strong></td>
<td>No</td>
<td>Must be 3.6.0.</td>
</tr>
<tr>
<td><strong>KmsKeyId</strong></td>
<td>No</td>
<td>If you choose to encrypt your cluster, you cannot change the AWS KMS key that you used to encrypt your cluster.</td>
</tr>
<tr>
<td><strong>MasterUsername</strong></td>
<td>No</td>
<td>After a cluster has been created, you cannot modify the MasterUsername.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Naming constraints:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Length is [1—63] alphanumeric characters.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• First character must be a letter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Cannot be a word reserved by the database engine.</td>
</tr>
<tr>
<td><strong>MasterUserPassword</strong></td>
<td>Yes</td>
<td>Constraints:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Length is [8—100] printable ASCII characters.</td>
</tr>
</tbody>
</table>
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:

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.

Using the AWS CLI

Use the describe-db-clusters operation to see 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 \
```
Amazon DocumentDB Developer Guide
Cluster Lifecycle

```
--db-cluster-identifier sample-cluster \n--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. 189)
- Describing Amazon DocumentDB Clusters (p. 199)
- Modifying an Amazon DocumentDB Cluster (p. 203)
- Determining Pending Maintenance (p. 206)
- Upgrading a Cluster's Engine Version (p. 207)
- Stopping and Starting an Amazon DocumentDB Cluster (p. 209)
- Deleting an Amazon DocumentDB Cluster (p. 212)

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.

For instructions on connecting to your Amazon DocumentDB cluster, see the following Getting Started topics:
- Step 2: Launch an Amazon EC2 Instance (p. 40)
- Step 3: Access and Use Your Amazon DocumentDB Cluster Using the mongo Shell (p. 42)

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**

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.

**VPC Prerequisites**

You can only create an Amazon DocumentDB cluster in an Amazon Virtual Private Cloud (Amazon VPC) that spans three Availability Zones. Each Availability Zone must contain at least one subnet. By distributing your cluster instances across at least three Availability Zones, you ensure that instances are available in your cluster in the unlikely case of an Availability Zone failure. The cluster volume for your Amazon DocumentDB cluster always spans three Availability Zones to provide durable storage with minimal possibility of data loss.

If you are using the console to create your Amazon DocumentDB cluster, by default the console will create the cluster in your account's default Amazon VPC. Alternatively, you can choose any VPC in which to create Amazon DocumentDB cluster. Your VPC must have at least one subnet in each of at least three Availability Zones in order for you to use it with an Amazon DocumentDB cluster.

**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](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. 256).

• 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


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. **Instance class**—Accept the default *db.r5.large*, or choose the instance class that you want from the list.

c. **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.

![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. 194).
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 is 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**

a. Sign in to the AWS Management Console and open the Amazon EC2 console at [https://console.aws.amazon.com/ec2/](https://console.aws.amazon.com/ec2/).

b. In the **Resources** section of the main window, choose **Security groups**.

c. 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.

d. From the **Actions** menu, choose **Edit inbound rules** then choose or enter the rule constraints.

   i. **Type**—From the list, choose the protocol to open to network traffic.

   ii. **Protocol**—From the list, choose the type of protocol.

   iii. **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).
iv. **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).

v. **Description**—Enter a description for this rule.

vi. 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 pane](image)

   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 pane](image)

   a. **Database 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.
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.

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.
6. Complete the Maintenance pane.

   • 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.

       a. Start day—In the list, choose the day of the week to start cluster maintenance.

       b. Start time—In the lists, choose the hour and minute (UTC) to start maintenance.

       c. 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.
For each tag you want to add to the cluster, repeat the following steps. You may have up to 10 on a cluster.

a. Choose Add tags.
b. Type the tag’s Key.
c. 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.

![Deletion Protection](image)

For more information about deletion protection, see Deleting an Amazon DocumentDB Cluster (p. 212).

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. 212).

- **--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.
Cluster Lifecycle

- 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. 392).

**To launch an Amazon DocumentDB cluster using the AWS CLI**

1. Identify the subnet group and Amazon VPC security group ID for your new cluster.
2. Then call the `create-db-cluster` AWS CLI operation to create the Amazon DocumentDB cluster.

**Example**

The following AWS CLI code creates an Amazon DocumentDB cluster named `sample-cluster` with deletion protection enabled.

For Linux, macOS, or Unix:

```bash
code
aws docdb create-db-cluster \
  --db-cluster-identifier sample-cluster \
  --engine docdb \
  --deletion-protection \
  --master-username master-user \
  --master-user-password password
```

For Windows:

```bash
code
aws docdb create-db-cluster ^
  --db-cluster-identifier sample-cluster ^
  --engine docdb ^
  --deletion-protection ^
  --master-username master-user ^
  --master-user-password password
```

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

```json
code
{
  "DBCluster": {
    "StorageEncrypted": false,
    "DBClusterMembers": [],
    "Engine": "docdb",
    "DeletionProtection": "enabled",
    "ClusterCreateTime": "2018-11-26T17:15:19.885Z",
    "DBSubnetGroup": "default",
    "EngineVersion": "3.6.0",
    "MasterUsername": "master-user",
    "BackupRetentionPeriod": 1,
    "DBClusterArn": "arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster",
  }
}
```
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. 303).

**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. 221).

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. 303)
- Finding a Cluster's Endpoints (p. 280)

**Using the AWS Management Console**

Use the following procedure to view the details of a specified Amazon DocumentDB cluster using the console.

2. In the navigation pane, choose **Clusters**.
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. 308).
- **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.

```sh
aws docdb describe-db-clusters --filter Name=engine,Values=docdb
```

Output from this operation looks something like the following.

```json
{
  "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 \n  --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.

```json
{
  "DBClusters": [ 
    
  ]
}
```
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.

Example
For Linux, macOS, or Unix:

```bash
aws docdb describe-db-clusters \
  --filter Name=engine,Values=docdb \
  --db-cluster-identifier sample-cluster \
  --query 'DBClusters[*].[DBClusterMembers]'
```

For Windows:

```bash
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. 209).

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**

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. 303).

### 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. 212).

• **--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:

```bash
aws docdb modify-db-cluster \
  --db-cluster-identifier sample-cluster \
  --apply-immediately \
  --backup-retention-period 7
```

For Windows:

```bash
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-VDP53QEWST7YHM36TTXOJPJT5YE",
    "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,
  }
}
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. 303).

## 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.

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.

   ![Maintenance column in AWS Management Console]

   *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 Upgrading a Cluster's Engine Version (p. 207).
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 ^
  --region us-east-1
```

Output from this operation looks something like the following.

```
{
  "PendingMaintenanceActions": [
    {
      "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 Upgrading a Cluster's Engine Version (p. 207).

**Upgrading a Cluster's Engine Version**

If you are running an earlier engine version, you can upgrade your cluster's engine version using the AWS Management Console or the AWS CLI. You can upgrade it immediately or during your cluster's next maintenance window. When you upgrade the engine version, your cluster will experience some downtime. To determine whether your engine needs an upgrade, see Determining Pending Maintenance (p. 206).
To upgrade 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 upgrades your cluster's engine version to the latest version using the console. You have the option to upgrade immediately or during your cluster's next maintenance window.

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 upgrade your cluster to a newer engine version 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`. 

208
Cluster Lifecycle

• `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 upgrades the engine version of `sample-cluster` to version 3.6.0.

For Linux, macOS, or Unix:

```bash
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:

```bash
aws docdb apply-pending-maintenance-action ^
  --apply-action db-upgrade ^
  --opt-in-type immediate
```

Output from this operation looks like the following.

```json
{
  "ResourcePendingMaintenanceActions": {
    "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. 210)
• Using the AWS Management Console (p. 210)
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

2. In the navigation pane, choose Clusters.
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.

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. 188).

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:

```bash
aws docdb stop-db-cluster
   --db-cluster-identifier sample-cluster
```
Cluster Lifecycle

For Windows:

```bash
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:

```bash
aws docdb start-db-cluster \
   --db-cluster-identifier sample-cluster
```

For Windows:

```bash
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. 166).

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. 209).

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 enabled, you can’t delete a cluster. To delete a cluster that has deletion projection 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. 189)
- Modifying an Amazon DocumentDB Cluster (p. 203)

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. 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. 203).
   
   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. 234).
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. 234).

   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. 303).

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",  
      "sample-instance2"  
    ]  
  ]  
]
```

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 ^
```
Managing Performance and Scaling for Amazon DocumentDB Clusters

You can use storage scaling, instance scaling, and read scaling to manage performance and scaling for your Amazon DocumentDB clusters and instances.

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.

The size of your cluster volume is checked on an hourly basis to determine your storage costs.

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. 229).

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. 221).

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. 347)

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 fails over to a new primary instance in one of two ways:
• By promoting an existing Amazon DocumentDB replica to the new primary instance and then creating a replacement for the promoted replica.
• By creating a new primary instance.

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. 247).

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. 217)
• Determining an Instance's Status (p. 221)
• Amazon DocumentDB Instance Lifecycle (p. 221)

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 and R5 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. 220).

Topics
• Determining an Instance's Class (p. 218)
Determining an Instance's 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

You can use the console to determine what the instance class is for your cluster's instances.

**To determine the instance classes for your cluster**

2. To find the instance that you're interested in, choose **Instances** in the navigation pane.
3. In the list of instances, find the instance that you want. Then look at the **Class** column of the instance's row.

In the following image, the instance class for instance `docdb-2019-05-13-20-51-52` is `db.r5.4xlarge`.

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 up to 100 instances.
For Linux, macOS, or Unix:

```
aws docdb describe-db-instances \
   --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]'  
```

For Windows:

```
aws docdb describe-db-instances ^
   --query 'DBInstances[*].[DBInstanceIdentifier,DBInstanceClass]'  
```

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

```
[  
  [  
    "db.r5.large"  
  ],  
  [  
    "db.r5.large"  
  ],  
  [  
    "docdb-2019-05-13-20-51-52",  
    "db.r5.4xlarge"  
  ],  
  [  
    "db.r5.4xlarge"  
  ]  
]
```

For more information, see Describing Amazon DocumentDB Instances (p. 226).

### 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. 229).

### Supported Instance Classes by Region

Amazon DocumentDB supports the following instance classes:

- R5—Current generation memory-optimized
- R4—Current generation memory-optimized

For detailed specifications on the instance classes, see Instance Class Specifications (p. 220).

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.

<table>
<thead>
<tr>
<th>Region</th>
<th>R5</th>
<th>R4</th>
</tr>
</thead>
<tbody>
<tr>
<td>US East (Ohio)</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>US East (N. Virginia)</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

219
## 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**

<table>
<thead>
<tr>
<th>Instance class</th>
<th>vCPU</th>
<th>ECU</th>
<th>Memory (GiB)</th>
<th>Max. bandwidth (Mbps)</th>
<th>Network performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>db.r5.large</td>
<td>2</td>
<td>10</td>
<td>16</td>
<td>Up to 3,500</td>
<td>Up to 10 Gbps</td>
</tr>
<tr>
<td>db.r5.xlarge</td>
<td>4</td>
<td>19</td>
<td>32</td>
<td>Up to 3,500</td>
<td>Up to 10 Gbps</td>
</tr>
<tr>
<td>db.r5.2xlarge</td>
<td>8</td>
<td>38</td>
<td>64</td>
<td>Up to 3,500</td>
<td>Up to 10 Gbps</td>
</tr>
<tr>
<td>db.r5.4xlarge</td>
<td>16</td>
<td>71</td>
<td>128</td>
<td>3,500</td>
<td>Up to 10 Gbps</td>
</tr>
<tr>
<td>db.r5.12xlarge</td>
<td>48</td>
<td>173</td>
<td>384</td>
<td>7,000</td>
<td>10 Gbps</td>
</tr>
<tr>
<td>db.r5.24xlarge</td>
<td>96</td>
<td>347</td>
<td>768</td>
<td>14,000</td>
<td>25 Gbps</td>
</tr>
<tr>
<td>db.r4.large</td>
<td>2</td>
<td>7</td>
<td>15.25</td>
<td>437</td>
<td>Up to 10 Gbps</td>
</tr>
<tr>
<td>db.r4.xlarge</td>
<td>4</td>
<td>13.5</td>
<td>30.5</td>
<td>875</td>
<td>Up to 10 Gbps</td>
</tr>
<tr>
<td>db.r4.2xlarge</td>
<td>8</td>
<td>27</td>
<td>61</td>
<td>875</td>
<td>Up to 10 Gbps</td>
</tr>
<tr>
<td>db.r4.4xlarge</td>
<td>16</td>
<td>53</td>
<td>122</td>
<td>875</td>
<td>Up to 10 Gbps</td>
</tr>
<tr>
<td>db.r4.8xlarge</td>
<td>32</td>
<td>99</td>
<td>244</td>
<td>875</td>
<td>10 Gbps</td>
</tr>
<tr>
<td>db.r4.16xlarge</td>
<td>64</td>
<td>195</td>
<td>488</td>
<td>14,000</td>
<td>25 Gbps</td>
</tr>
</tbody>
</table>
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. 305).

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. 221)
- Describing Amazon DocumentDB Instances (p. 226)
- Modifying an Amazon DocumentDB Instance (p. 229)
- Rebooting an Amazon DocumentDB Instance (p. 231)
- Deleting an Amazon DocumentDB Instance (p. 234)

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. 209).

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.

<table>
<thead>
<tr>
<th>Instance class</th>
<th>vCPU</th>
<th>ECU</th>
<th>Memory (GiB)</th>
<th>Max. bandwidth (Mbps)</th>
<th>Network performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. 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.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. ECU—The relative measure of the integer processing power of an Amazon Elastic Compute Cloud (Amazon EC2) instance. To make it easier for developers to compare the CPU capacity between different instance classes, we defined an Amazon EC2 compute unit. The amount of CPU that is allocated to a particular instance is expressed in terms of these EC2 compute units (ECUs). One ECU currently provides CPU capacity equivalent to a 1.0–1.2 GHz 2007 Opteron or 2007 Xeon processor.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Memory (GiB)—The RAM, in gigabytes, that is allocated to the instance. There is often a consistent ratio between memory and vCPU.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Max. bandwidth (Mbps) - The maximum bandwidth in megabits per second. Divide by 8 to get the expected throughput in megabytes per second.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. Network performance—The network speed relative to other instance classes.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
To create an instance

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 upon 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, Amazon Neptune, and Amazon DocumentDB per AWS account, per Region.

   b. **Instance class**—From the list, choose the instance type you want for this instance.

   c. **Promotion tier**—From the list, choose No preference to allow Amazon DocumentDB to set the promotion tier for the instance, or choose the promotion tier for the instance. Lower numbers mean higher priority. For more information, see Controlling the Failover Target (p. 247).

   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. 305).

**Adding an Instance Using the AWS CLI**

Use the `create-db-instance` AWS CLI operation with the below 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 lowercase 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, Amazon 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. 245).
- **--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. 247).

**Example**

If you want to specify the availability zone before you create your instance, run the following command to determine which Availability Zones you can create your instance in.

**For Linux, macOS, or Unix:**

```bash
aws docdb describe-db-clusters \
  --query 'DBClusters[*].[DBClusterIdentifier,AvailabilityZones[*]]'
```

**For Windows:**

```bash
aws docdb describe-db-clusters ^
  --query 'DBClusters[*].[DBClusterIdentifier,AvailabilityZones[*]]'
```

Output from this operation looks something like the following.

```
[
  [
    "sample-cluster",
    [
      "us-east-1b",
      "us-east-1d",
      "us-east-1e"
    ]
  ]
]
```
To determine which instance classes are available to you in your Region, run the following code. From the output, choose an instance class.

For Linux, macOS, or Unix:

```bash
aws docdb describe-orderable-db-instance-options
   --engine docdb
   --query 'OrderableDBInstanceOptions[*].DBInstanceClass'
```

For Windows:

```bash
aws docdb describe-orderable-db-instance-options
   --engine docdb
   --query 'OrderableDBInstanceOptions[*].DBInstanceClass'
```

Output from this operation looks something like the following:

```json
[
  "db.r5.16xlarge",
  "db.r5.2xlarge",
  "db.r5.4xlarge",
  "db.r5.8xlarge",
  "db.r5.large",
  "db.r5.xlarge"
]
```

Finally, run the following code to add an instance to your `sample-cluster`.

For Linux, macOS, or Unix:

```bash
aws docdb create-db-instance
   --db-cluster-identifier sample-cluster
   --db-instance-identifier sample-cluster-instance-00
   --availability-zone us-east-1b
   --promotion-tier 2
   --db-instance-class db.r5.xlarge
   --engine docdb
```

For Windows:

```bash
aws docdb create-db-instance
   --db-cluster-identifier sample-cluster
   --db-instance-identifier sample-cluster-instance-00
   --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:

```json
{
   "DBInstance": {
      "DbiResourceId": "db-SAMPLERESOURCEID",
      "DBClusterIdentifier": "sample-cluster",
   }
}
```
"DBInstanceIdentifier": "sample-cluster-instance-00",
"PendingModifiedValues": {},
"PromotionTier": 2,
"DBInstanceArn": "arn:aws:rds:us-east-1:012345678901:db:sample-cluster-instance-00",
"VpcSecurityGroups": [
  {
    "VpcSecurityGroupId": "sg-77186e0d",
    "Status": "active"
  }
],
"StorageEncrypted": true,
"DBInstanceClass": "db.r5.large",
"DBSubnetGroup": {
  "SubnetGroupStatus": "Complete",
  "DBSubnetGroupName": "default",
  "VpcId": "vpc-91280df6",
  "DBSubnetGroupDescription": "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"
    }
  ]
},
"BackupRetentionPeriod": 1,
"KmsKeyId": "arn:aws:kms:us-east-1:012345678901:key/0961325d-a50b-44d4-b6a0-a177d5ff730b"
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. 305).

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.

2. In the navigation pane, choose Instances.
   
   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 instances, choose the name of the instance that you want to see the details of. 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. 203).
   
   • Monitoring—The CloudWatch Logs metrics for this instance. For more information, see Monitoring Amazon DocumentDB with CloudWatch (p. 308).
   
   • 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
<guillemot>Tags</guillemot> section lists the tags attached to this cluster. For more information, see Tagging Amazon DocumentDB Resources (p. 288).

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.

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:

   ```bash
   aws docdb describe-db-instances \
   --filter Name=engine,Values=docdb
   ```

   For Windows:

   ```bash
   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:

   ```bash
   aws docdb describe-db-instances \
   --db-instance-identifier sample-cluster-instance
   ```

   For Windows:

   ```bash
   aws docdb describe-db-instances \
   --db-instance-identifier sample-cluster-instance
   ```

   Output from this operation looks like the following.

   ```json
   { 
   "DBInstances": [ 
   { 
   "DbiResourceId": "db-BJKKB54PIDV5QFKGVRX5T3S6GM", 
   "DBInstanceArn": "arn:aws:rds:us-east-1:012345678901:db:sample-cluster-instance-00", 
   "VpcSecurityGroups": [ 
   { 
   "VpcSecurityGroupId": "sg-77186e0d", 
   "Status": "active" 
   } 
   ], 
   "DBInstanceClass": "db.r5.large", 
   "DBInstanceStatus": "creating", 
   "AutoMinorVersionUpgrade": true, 
   "PreferredMaintenanceWindow": "fri:09:32-fri:10:02",
   ```
"BackupRetentionPeriod": 1,
"StorageEncrypted": true,
"DBClusterIdentifier": "sample-cluster",
"EngineVersion": "3.6.0",
"AvailabilityZone": "us-east-1a",
"Engine": "docdb",
"PromotionTier": 2,
"DBInstanceIdentifier": "sample-cluster-instance",
"PreferredBackupWindow": "00:00-00:30",
"PubliclyAccessible": false,
"DBSubnetGroup": {
  "DBSubnetGroupName": "default",
  "Subnets": [
    {
      "SubnetIdentifier": "subnet-4e26d263",
      "SubnetAvailabilityZone": {
        "Name": "us-east-1a"
      },
      "SubnetStatus": "Active"
    },
    {
      "SubnetIdentifier": "subnet-afc329f4",
      "SubnetAvailabilityZone": {
        "Name": "us-east-1c"
      },
      "SubnetStatus": "Active"
    },
    {
      "SubnetIdentifier": "subnet-b3806e8f",
      "SubnetAvailabilityZone": {
        "Name": "us-east-1e"
      },
      "SubnetStatus": "Active"
    },
    {
      "SubnetIdentifier": "subnet-53ab3636",
      "SubnetAvailabilityZone": {
        "Name": "us-east-1d"
      },
      "SubnetStatus": "Active"
    },
    {
      "SubnetIdentifier": "subnet-991cb8d0",
      "SubnetAvailabilityZone": {
        "Name": "us-east-1b"
      },
      "SubnetStatus": "Active"
    },
    {
      "SubnetIdentifier": "subnet-29ab1025",
      "SubnetAvailabilityZone": {
        "Name": "us-east-1f"
      },
      "SubnetStatus": "Active"
    }
  ],
  "VpcId": "vpc-91280df6",
  "DBSubnetGroupDescription": "default",
  "SubnetGroupStatus": "Complete"
},
"PendingModifiedValues": {},
"KmsKeyId": "arn:aws:kms:us-east-1:012345678901:key/0961325d-a50b-44d4-b6a0-a177d5ff730b"}
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. 209).

Using the AWS Management Console

Follow these steps to modify a specific Amazon DocumentDB instance using the console.

2. In the navigation pane, choose Instances.
   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 instance that you want to modify by choosing the button to the left of the instance name.
4. Choose Actions, and then choose Modify.
5. 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:
     • Must contain [1—63] alphanumeric characters or hyphens.
     • First character must be a letter.
     • Cannot end with a hyphen or contain two consecutive hyphens.
   • Certificate authority — Server certificate for this instance. For more information, see Updating Your Amazon DocumentDB TLS Certificates (p. 139).
   • Failover — During failover, the instance with the highest promotion tier will be promoted to primary. For more information, see Amazon DocumentDB Failover (p. 247).
   • Maintenance — The maintenance window in which pending modifications or patches are applied to instances in the cluster.
6. When you have finished, choose Continue to see a summary of your changes.
7. 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. 305).

Using the AWS CLI

The following code modifies the instance class to db.r5.xlarge for the instance sample-instance.

Parameters

• --db-instance-identifier — Required. The identifier for the instance to be modified.
• **--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:

```bash
aws docdb modify-db-instance \
   --db-instance-identifier sample-instance \
   --db-instance-class db.r5.4xlarge \
   --apply-immediately
```

For Windows:

```bash
aws docdb modify-db-instance ^
   --db-instance-identifier sample-instance ^
   --db-instance-class db.r5.4xlarge ^
   --apply-immediately
```

Output from this operation looks something like the following.

```json
{
   "DBInstances": [
      {
         "DBInstanceIdentifier": "sample-instance-1",
         "DBInstanceClass": "db.r5.large",
         "Engine": "docdb",
         "DBInstanceStatus": "modifying",
         "Endpoint": {
            "Address": "sample-instance-1.node.us-east-1.docdb.amazonaws.com",
            "Port": 27017,
            "HostedZoneId": "ABCDEFGHIJKLM"
         },
         "InstanceCreateTime": "2020-01-10T22:18:55.921Z",
         "PreferredBackupWindow": "02:00-02:30",
         "BackupRetentionPeriod": 1,
         "VpcSecurityGroups": [
            {
               "VpcSecurityGroupId": "sg-abcd0123",
               "Status": "active"
            }
         ],
         "AvailabilityZone": "us-east-1a",
         "DBSubnetGroup": {
            "DBSubnetGroupName": "default",
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-abcd0123",
            "SubnetGroupStatus": "Complete",
            "Subnets": [
               {
                  "SubnetIdentifier": "subnet-abcd0123",
                  "SubnetAvailabilityZone": {
                     "Name": "us-east-1a"
                  },
                  "SubnetStatus": "Active"
               },
               {
                  "SubnetIdentifier": "subnet-abcd0123",
                  "SubnetAvailabilityZone": {
                     "Name": "us-east-1b"
                  },
                  "SubnetStatus": "Active"
               }
            ]
         }
      }
   ]
}```
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. 305).

Rebooting an Amazon DocumentDB Instance

Occasionally, you might need to reboot your Amazon DocumentDB (with MongoDB compatibility) instance, usually for maintenance reasons. If you make certain changes, such as changing the 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 results in a momentary outage, during which the instance status is set to rebooting. If the Amazon DocumentDB instance is configured for Multi-AZ, the reboot can be conducted with a failover. An Amazon DocumentDB event is created when the reboot is completed.

If your instance is a Multi-AZ deployment, you can force a failover from one Availability Zone to another when you reboot. When you force a failover of your instance, Amazon DocumentDB automatically switches to a standby replica in another Availability Zone. It then updates the DNS record for the instance to point to the standby instance. As a result, you need to clean up and re-establish any existing connections to your instance. Rebooting with failover is beneficial when you want to simulate a failure of an instance for testing, or restore operations to the original Availability Zone after a failover occurs.

When you reboot the primary instance of an Amazon DocumentDB cluster, Amazon DocumentDB also automatically reboots all of the Amazon DocumentDB replicas in that cluster. When you reboot an Amazon DocumentDB replica, no failover occurs. To fail over an Amazon DocumentDB cluster, call the AWS CLI operation failover-db-cluster.

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.

You can reboot a specified instance using the AWS Management Console or the AWS CLI.

Using the AWS Management Console

The following procedure reboots the instance that you specify using the console.
To reboot an instance using the console

2. In the navigation pane, choose **Instances**.
   
   **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 specify an instance to reboot, locate the instance in the list of instances, and choose the button to the left of its name.
4. Choose **Actions**, choose **Reboot**, and then choose **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. 305).

Using the AWS CLI

To reboot an Amazon DocumentDB instance, use the `reboot-db-instance` operation with the following parameter.

**Parameters**

- **--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:

```bash
aws docdb reboot-db-instance \
  --db-instance-identifier sample-db-instance
```

For Windows:

```bash
aws docdb reboot-db-instance ^
  --db-instance-identifier sample-db-instance
```

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

```json
{
  "DBInstance": {
    "AutoMinorVersionUpgrade": true,
    "PubliclyAccessible": false,
    "PreferredMaintenanceWindow": "fri:09:32-fri:10:02",
    "PendingModifiedValues": {},
    "DBInstanceStatus": "rebooting",
    "DBSubnetGroup": {
      "Subnets": [
        {
          "SubnetStatus": "Active",
          "SubnetAvailabilityZone": {
            "Name": "us-east-1a"
          }
        }
      ]
    }
  }
}
```
"SubnetIdentifier": "subnet-4e26d263"
},

"SubnetStatus": "Active",
"SubnetAvailabilityZone": {
  "Name": "us-east-1c"
},

"SubnetIdentifier": "subnet-a4f329f4"
},

"SubnetStatus": "Active",
"SubnetAvailabilityZone": {
  "Name": "us-east-1e"
},

"SubnetIdentifier": "subnet-b3806e8f"
},

"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-
3f730b",
"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.corcjozrlsfc.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. 305).

Deleting an Amazon DocumentDB Instance

You can delete your Amazon DocumentDB instance using either the AWS Management Console or the AWS CLI.

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.)

Important
To delete an instance, the instance must be in the available state. You cannot delete an instance that is stopped. If the cluster 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. 209).

You can delete an Amazon DocumentDB instance using the AWS Management Console or the AWS CLI.

Using the AWS Management Console

The following procedure deletes the instance of your choice using the console.

To delete an instance using the console

2. In the navigation pane, choose Instances.
   
   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 the button to the left of the name of the instance that you want to delete.
4. Choose Actions, and then choose Delete.

It takes several minutes for the instance to be deleted. To monitor the status of the instance, see Monitoring an Amazon DocumentDB Instance's Status (p. 305).

Using the AWS CLI

Use the following AWS CLI code to delete the cluster sample-cluster-instance.

Required parameters

• `--db-instance-identifier`—Required. The identifier of the instance that you want to delete.
Example

For Linux, macOS, or Unix:

```bash
aws docdb delete-db-instance \
  --db-instance-identifier sample-cluster-instance-00
```

For Windows:

```bash
aws docdb delete-db-instance ^
  --db-instance-identifier sample-cluster-instance-00
```

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

```json
{
  "DBInstance": {
    "DBClusterIdentifier": "sample-cluster",
    "StorageEncrypted": true,
    "DBInstanceStatus": "deleting",
    "DBInstanceArn": "arn:aws:rds:us-east-1:012345678901:db:sample-cluster-instance-00",
    "PendingModifiedValues": {},
    "DBSubnetGroup": {
      "DBSubnetGroupDescription": "default",
      "VpcId": "vpc-91280df6",
      "SubnetGroupStatus": "Complete",
      "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"
        },
        {
          "SubnetAvailabilityZone": {
            "Name": "us-east-1d"
          },
          "SubnetIdentifier": "subnet-53ab3636",
          "SubnetStatus": "Active"
        },
        {
          "SubnetAvailabilityZone": {
            "Name": "us-east-1b"
          },
          "SubnetIdentifier": "subnet-991cb8d0",
          "SubnetStatus": "Active"
        }
      ]
    }
  }
}
```
It takes several minutes for the instance to be deleted. To monitor the status of the instance, see Monitoring an Amazon DocumentDB Instance's Status (p. 305).

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
Creating a Subnet Group

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. 7).

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. 237)
- Describing an Amazon DocumentDB Subnet Group (p. 240)
- Modifying an Amazon DocumentDB Subnet Group (p. 242)
- Deleting an Amazon DocumentDB Subnet Group (p. 244)

Creating an Amazon DocumentDB 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. 237)
- Using the AWS CLI (p. 238)

Using the AWS Management Console

Use the following steps to create an Amazon DocumentDB subnet group.

To create an Amazon DocumentDB subnet group

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.
Creating a Subnet Group

- 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.

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:

```bash
aws docdb describe-db-subnet-groups
   --db-subnet-group-name default
   --query 'DBSubnetGroups[*].[DBSubnetGroupName,Subnets[*].[SubnetAvailabilityZone.Name,SubnetIdentifier]]'
```

For Windows:

```bash
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).

```json
[
  [
    "default",
    [
      ["us-east-1a",
       "subnet-4e26d263"],
      ["us-east-1c",
```
Creating a Subnet Group

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:

```bash
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:

```bash
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).

```json
{
  "DBSubnetGroup": {
    "DBSubnetGroupDescription": "A sample subnet group",
    "VpcId": "vpc-03030303",
    "Subnets": [
      "subnet-4e26d263",
      "subnet-afc329f4",
      "subnet-b3806e8f"
    ]
  }
}
```
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**

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.
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. 237) section.

For Linux, macOS, or Unix:

```bash
aws docdb describe-db-subnet-groups \
   --db-subnet-group-name sample-subnet-group
```

For Windows:

```sql
aws docdb describe-db-subnet-groups ^
   --db-subnet-group-name sample-subnet-group
```

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

```json
{
   "DBSubnetGroup": {
      "DBSubnetGroupArn": "arn:aws:rds:us-east-1:123SAMPLE012:subgrp:sample-subnet-group",
      "VpcId": "vpc-91280df6",
      "SubnetGroupStatus": "Complete",
      "DBSubnetGroupName": "sample-subnet-group",
      "Subnets": [
```
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. 242)
- Using the AWS CLI (p. 243)

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

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:

   a. To make these changes to the subnet group, choose **Modify**.
   
   b. 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:

```bash
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:

```bash
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. 237) section. However, the subnets in the subnet group are replaced with those listed in the modify-db-subnet-group operation.

```json
{
  "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. 244)
- Using the AWS CLI (p. 245)

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

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:

```bash
aws docdb delete-db-subnet-group
   --db-subnet-group-name sample-subnet-group
```

For Windows:

```bash
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 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. 216)
- Amazon DocumentDB Failover (p. 247)
  - Controlling the Failover Target (p. 247)
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. 189). To add more replicas to an Amazon DocumentDB cluster, see Adding an Amazon DocumentDB Instance to a Cluster (p. 221).

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. 189)
- Adding an Amazon DocumentDB Instance to a Cluster (p. 221)

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 lowest numbered priority 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 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. 221)
- Modifying an Amazon DocumentDB Instance (p. 229)

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.
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 priority.

For Linux, macOS, or Unix:

```bash
aws docdb failover-db-cluster \
    --db-cluster-identifier sample-cluster
```

For Windows:

```bash
aws docdb failover-db-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:

```bash
aws docdb failover-db-cluster 
--db-cluster-identifier sample-cluster 
--target-db-instance-identifier sample-cluster-instance
```

For Windows:

```bash
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).

```json
{
    "DBCluster": {
        "HostedZoneId": "Z2SUY0A1719RZT",
        "Port": 27017,
        "EngineVersion": "3.6.0",
        "PreferredMaintenanceWindow": "thu:04:05-thu:04:35",
        "BackupRetentionPeriod": 1,
        "ClusterCreateTime": "2018-06-28T18:53:29.455Z",
        "AssociatedRoles": [],
        "DBSubnetGroup": "default",
        "MasterUsername": "master-user",
        "Engine": "docdb",
        "ReadReplicaIdentifiers": [],
        "EarliestRestorableTime": "2018-08-21T00:04:10.546Z",
        "DBClusterIdentifier": "sample-cluster",
        "ReaderEndpoint": "sample-cluster.node.us-east-1.docdb.amazonaws.com",
        "DBClusterMembers": [
            {
                "DBInstanceIdentifier": "sample-cluster-instance",
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1,
                "IsClusterWriter": true
            },
            {
                "DBInstanceIdentifier": "sample-cluster-instance-00",
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1,
                "IsClusterWriter": false
            },
            {
                "DBInstanceIdentifier": "sample-cluster-instance-01",
                "DBClusterParameterGroupStatus": "in-sync",
                "PromotionTier": 1,
                "IsClusterWriter": false
            }
        ],
        "AvailabilityZones": [
            "us-east-1b",
            "us-east-1c",
            "us-east-1a"
        ],
        "DBClusterParameterGroup": "default.docdb3.6",
    }
}
```
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. 354).

Topics
• Viewing Amazon DocumentDB Event Categories (p. 250)
• Viewing Amazon DocumentDB Events (p. 252)

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-snapshot`
  - `db-cluster-snapshot`
- `--filters`—Optional. To view the event categories for just Amazon DocumentDB, use the filter `--filter Name=engine,Values=docdb`. 

```json
"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"
}
Example

The following code lists the event categories associated with clusters.

For Linux, macOS, or Unix:

```bash
aws docdb describe-event-categories
   --filter Name=engine,Values=docdb \
   --source-type db-cluster
```

For Windows:

```bash
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.

```bash
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 replicas",
            "backtrack",
            "low storage",
            "backup",
            "availability",
            "failover"
         ]
      }
   ]
}
```
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. 252)
- Using the AWS CLI (p. 253)

**Using the AWS Management Console**

**To view all Amazon DocumentDB instance events for the past 24 hours**

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 Screenshot]

### 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).

```bash
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-snapshot`
  - `db-cluster-snapshot`

The following example lists all your Amazon DocumentDB events.

```bash
aws docdb describe-events --filters Name=engine,Values=docdb
```

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

```json
{
    "Events": [
```
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.

For more information, see Auditing Amazon DocumentDB Events (p. 149).
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

<table>
<thead>
<tr>
<th>Region Name</th>
<th>Region</th>
<th>Availability Zones (compute)</th>
</tr>
</thead>
<tbody>
<tr>
<td>US East (Ohio)</td>
<td>us-east-2</td>
<td>3</td>
</tr>
<tr>
<td>US East (N. Virginia)</td>
<td>us-east-1</td>
<td>6</td>
</tr>
<tr>
<td>US West (Oregon)</td>
<td>us-west-2</td>
<td>4</td>
</tr>
<tr>
<td>Asia Pacific (Mumbai)</td>
<td>ap-south-1</td>
<td>3</td>
</tr>
<tr>
<td>Asia Pacific (Seoul)</td>
<td>ap-northeast-2</td>
<td>3</td>
</tr>
<tr>
<td>Asia Pacific (Singapore)</td>
<td>ap-southeast-1</td>
<td>3</td>
</tr>
<tr>
<td>Asia Pacific (Sydney)</td>
<td>ap-southeast-2</td>
<td>3</td>
</tr>
<tr>
<td>Asia Pacific (Tokyo)</td>
<td>ap-northeast-1</td>
<td>3</td>
</tr>
</tbody>
</table>
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. 279).

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. 256)
- Creating Amazon DocumentDB Cluster Parameter Groups (p. 261)
- Modifying Amazon DocumentDB Cluster Parameter Groups (p. 262)
- Modifying Amazon DocumentDB Clusters to Use Customized Cluster Parameter Groups (p. 266)
- Copying Amazon DocumentDB Cluster Parameter Groups (p. 267)
- Resetting Amazon DocumentDB Cluster Parameter Groups (p. 268)
- Deleting Amazon DocumentDB Cluster Parameter Groups (p. 270)
- Amazon DocumentDB Cluster Parameters Reference (p. 272)

Describing Amazon DocumentDB Cluster Parameter Groups

When you create an Amazon DocumentDB cluster, a default.docdb3.6 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. 257)
- Determining an Amazon DocumentDB Cluster's Parameter Group (p. 259)
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

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. 275).
   
   • 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. 250).
   
   • 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. 288).

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.
Describing Cluster Parameter Groups

aws docdb describe-db-cluster-parameter-groups

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

```
{
   "DBClusterParameterGroups": [
   {
      "DBParameterGroupFamily": "docdb3.6",
      "Description": "Default cluster parameter group for docdb3.6",
      "DBClusterParameterGroupName": "default.docdb3.6"
   },
   {
      "DBParameterGroupFamily": "docdb3.6",
      "Description": "Custom docdb3.6 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": [
   {
      "Description": "Custom docdb3.6 parameter group",
      "DBParameterGroupFamily": "docdb3.6",
      "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**

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

Cluster identifier
sample-cluster ( available )

Cluster creation time
1/10/2020, 2:13:38 PM UTC-8

Cluster endpoint
sample-cluster.<region>.docdb.amazonaws.com

Reader endpoint
sample-cluster.<region>.docdb.amazonaws.com

Master username

Port
27017

Status
available

Cluster parameter group
sample-parameter-group
Creating Amazon DocumentDB Cluster Parameter Groups

Amazon DocumentDB provides a default cluster parameter group based on the docdb3.6 family named default.docdb3.6. You cannot modify the 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 associate that parameter group to your Amazon DocumentDB cluster.

The following procedure guides you through creating a custom cluster parameter group based on the docdb3.6 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. 256).

Using the AWS Management Console

To create a cluster parameter group

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, 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 `docdb3.6`.
- **--description** — The user-provided description for this cluster parameter group. The following example uses "Custom docdb3.6 parameter group".

For Linux, macOS, or Unix:

**Example**

```bash
aws docdb create-db-cluster-parameter-group \
  --db-cluster-parameter-group-name sample-parameter-group \
  --db-parameter-group-family docdb3.6 \
  --description "Custom docdb3.6 parameter group"
```

For Windows:

```bash
aws docdb create-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name sample-parameter-group ^
  --db-parameter-group-family docdb3.6 ^
  --description "Custom docdb3.6 parameter group"
```

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

```
{
  "DBClusterParameterGroup": {
    "DBClusterParameterGroupName": "sample-parameter-group",
    "DBParameterGroupFamily": "docdb3.6",
    "Description": "Custom docdb3.6 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 `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.

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. 256).

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. 203).

To modify a custom cluster parameter group

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 (Ξ) 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. 272). 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:

   ```bash
   aws docdb describe-db-cluster-parameters \
     --db-cluster-parameter-group-name sample-parameter-group
   ```

   For Windows:

   ```bash
   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).

   ```json
   
   { 
       "Parameters": [ 
           { 
               "Source": "system", 
               "ApplyType": "static", 
               "AllowedValues": "disabled,enabled", 
               "ParameterValue": "enabled", 
               "ApplyMethod": "pending-reboot", 
               "DataType": "string", 
               "ParameterName": "tls", 
               "IsModifiable": true, 
               "Description": "Config to enable/disable TLS"
           }, 
           { 
               "Source": "user", 
               "ApplyType": "dynamic", 
               "AllowedValues": "disabled,enabled", 
               "ParameterValue": "enabled", 
               "ApplyMethod": "pending-reboot", 
               "DataType": "string", 
               "ParameterName": "ttl_monitor", 
               "IsModifiable": true, 
               "Description": "Enables TTL Monitoring"
           }
       ]
   }
   ```

2. 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:

   ```bash
   aws docdb modify-db-cluster-parameter-group \
     --db-cluster-parameter-group-name sample-parameter-group \
   ```
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": "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.docdb3.6 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. 189) and Creating Amazon DocumentDB Cluster Parameter Groups (p. 261) 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).
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

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.
Resetting Cluster Parameter Groups

- 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:

```bash
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:

```bash
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).

```json
{
  "DBClusterParameterGroup": {
    "DBClusterParameterGroupName": "sample-parameter-group-copy",
    "DBParameterGroupFamily": "docdb3.6",
    "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**

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` their default values.

For Linux, macOS, or Unix:

```bash
aws docdb reset-db-cluster-parameter-group \
    --db-cluster-parameter-group-name sample-parameter-group \
    --reset-all-parameters
```

For Windows:

```bash
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:

```bash
aws docdb reset-db-cluster-parameter-group \
    --db-cluster-parameter-group-name sample-parameter-group \
```

For Windows:

```bash
aws docdb reset-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name sample-parameter-group ^
```
Deleting Cluster Parameter Groups

---no-reset-all-parameters \
--parameters ParameterName=tls,ApplyMethod=pending-reboot

For Windows:

```bash
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).

```json
{
   "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:

```bash
aws docdb reboot-db-instance ^
   --db-instance-identifier sample-cluster-instance
```

For Windows:

```bash
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.docdb3.6 cluster parameter group.

Using the AWS Management Console

**To delete a cluster parameter group**

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.docdb3.6` 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 example lists the names of all cluster parameter groups:

   For **Linux, macOS, or Unix**:

   ```bash
   aws docdb describe-db-cluster-parameter-groups --query 'DBClusterParameterGroups[*].[DBClusterParameterGroupName]'
   ```

   For **Windows**:

   ```bash
   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).

   ```json
   [
     [
       "default.docdb3.6",
     ],
     [
       "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**:

   ```bash
   aws docdb delete-db-cluster-parameter-group --db-cluster-parameter-group-name sample-parameter-group-copy
   ```

   For **Windows**:

   ```bash
   aws docdb delete-db-cluster-parameter-group --db-cluster-parameter-group-name sample-parameter-group-copy
   ```
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.docdb3.6"
  ],
  [
    "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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default Value</th>
<th>Valid Values</th>
<th>Modifiable</th>
<th>Apply Type</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>audit_logs</td>
<td>disabled</td>
<td>enabled, disabled</td>
<td>Yes</td>
<td>Dynamic</td>
<td>String</td>
<td>Defines whether AWS CloudTrail audit logs are enabled.</td>
</tr>
</tbody>
</table>

- **enabled**—CloudTrail audit
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default Value</th>
<th>Valid Values</th>
<th>Modifiable</th>
<th>Apply Type</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>logs_enabled</td>
<td></td>
<td>Disabled, enabled</td>
<td>Yes</td>
<td>Dynamic</td>
<td>String</td>
<td>Defines the state of CloudTrail audit logs.</td>
</tr>
<tr>
<td>change_stream_log_retention_duration</td>
<td>10800</td>
<td>3600-86400</td>
<td>Yes</td>
<td>Dynamic</td>
<td>Integer</td>
<td>Defines the duration of time (in seconds) that the change stream log is retained and can be consumed.</td>
</tr>
<tr>
<td>profiler</td>
<td>disabled</td>
<td>enabled, disabled</td>
<td>Yes</td>
<td>Dynamic</td>
<td>String</td>
<td>Enables profiling for slow operations.</td>
</tr>
<tr>
<td>profiler_sampling_rate</td>
<td>0.0-1.0</td>
<td>Yes</td>
<td>Dynamic</td>
<td>Float</td>
<td></td>
<td>Defines the sampling rate for logged operations.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Default Value</td>
<td>Valid Values</td>
<td>Modifiable</td>
<td>Apply Type</td>
<td>Data Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>---------------</td>
<td>-------------------------------</td>
<td>------------</td>
<td>------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>profiler_threshold_ms</td>
<td>100</td>
<td>50-2147483646</td>
<td>Yes</td>
<td>Dynamic</td>
<td>Integer</td>
<td>Defines the threshold for profiler.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• All operations greater than profiler_threshold_ms are logged to CloudWatch Logs.</td>
</tr>
<tr>
<td>tls</td>
<td>enabled</td>
<td>enabled, disabled</td>
<td>Yes</td>
<td>Static</td>
<td>String</td>
<td>Defines whether Transport Layer Security (TLS) connections are required.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• enabled—TLS connections are required to connect.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• disabled—TLS connections cannot be used to connect.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Default Value</td>
<td>Valid Values</td>
<td>Modifiable</td>
<td>Apply Type</td>
<td>Data Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------</td>
<td>-----------------------</td>
<td>------------</td>
<td>------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ttl_monitor</td>
<td>enabled</td>
<td>enabled, disabled</td>
<td>Yes</td>
<td>Dynamic</td>
<td>String</td>
<td>Defines whether Time to Live (TTL) monitoring is enabled for the cluster.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• enabled—TTL monitoring is enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• disabled—TTL monitoring is disabled.</td>
</tr>
</tbody>
</table>

**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**

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.
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**

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. 272).
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:
   ```bash
   aws docdb describe-db-cluster-parameters \
   --db-cluster-parameter-group-name sample-parameter-group
   ```

   For Windows:
   ```bash
   aws docdb describe-db-cluster-parameters ^
   ```
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).

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

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
```
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. 280)
- Finding an Instance’s Endpoint (p. 281)
- Connecting to Endpoints (p. 283)

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**

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.

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.
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:

```bash
aws docdb describe-db-clusters \
  --region us-east-1 \
  --db-cluster-identifier sample-cluster \
  --query 'DBClusters[*].[DBClusterIdentifier,Port,Endpoint,ReaderEndpoint]'s
```

For Windows:

```bash
aws docdb describe-db-clusters ^
  --region us-east-1 ^
  --db-cluster-identifier sample-cluster ^
  --query 'DBClusters[*].[DBClusterIdentifier,Port,Endpoint,ReaderEndpoint]'\n```

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

```json
[
  {
    "sample-cluster",
    27017,
    "sample-cluster.cluster-corlsfccjozr.us-east-1.docdb.amazonaws.com",
    "sample-cluster.cluster-ro-corlsfccjozr.us-east-1.docdb.amazonaws.com"
  }
]
```

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. 283).

## 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

2. In the navigation pane, choose **instances**.
3. From the list of instances, choose the name of the instance that you are interested in.
4. Scroll down to the **Details** section then locate the instance endpoint.

![Instance Details](image)

5. 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 Details](image)

**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
```
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.

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. 283).
Important
The certificate authority (CA) certificate for Amazon DocumentDB 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 and certificates, see Updating Your Amazon DocumentDB TLS Certificates (p. 139).

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:

```bash
mongo \
  --ssl \ 
  --host Endpoint:Port \ 
  --sslCAFile rds-combined-ca-bundle.pem \ 
  --username UserName \ 
  --password Password
```

**mongo shell examples**

Connect to a cluster:

```bash
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:

```bash
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.

```bash
mongodb://UserName:Password@endpoint:port?replicaSet=rs0&ssl_ca_certs=rds-combined-ca-bundle.pem
```

**Connection string examples**

Connect to a cluster:

```bash
mongodb://UserName:Password@sample-cluster.cluster-corlsfccjozr.us-east-1.docdb.amazonaws.com:27017?replicaSet=rs0&ssl_ca_certs=rds-combined-ca-bundle.pem
```

Connect to an instance:
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. 285)
- Finding an Amazon DocumentDB Resource ARN (p. 286)

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
```

<table>
<thead>
<tr>
<th>Region Name</th>
<th>Region</th>
<th>Availability Zones (compute)</th>
</tr>
</thead>
<tbody>
<tr>
<td>US East (Ohio)</td>
<td>us-east-2</td>
<td>3</td>
</tr>
<tr>
<td>US East (N. Virginia)</td>
<td>us-east-1</td>
<td>6</td>
</tr>
<tr>
<td>US West (Oregon)</td>
<td>us-west-2</td>
<td>4</td>
</tr>
<tr>
<td>Asia Pacific (Mumbai)</td>
<td>ap-south-1</td>
<td>3</td>
</tr>
<tr>
<td>Asia Pacific (Seoul)</td>
<td>ap-northeast-2</td>
<td>3</td>
</tr>
<tr>
<td>Asia Pacific (Singapore)</td>
<td>ap-southeast-1</td>
<td>3</td>
</tr>
<tr>
<td>Asia Pacific (Sydney)</td>
<td>ap-southeast-2</td>
<td>3</td>
</tr>
<tr>
<td>Asia Pacific (Tokyo)</td>
<td>ap-northeast-1</td>
<td>3</td>
</tr>
<tr>
<td>Canada (Central)</td>
<td>ca-central-1</td>
<td>3</td>
</tr>
<tr>
<td>Europe (Frankfurt)</td>
<td>eu-central-1</td>
<td>3</td>
</tr>
<tr>
<td>Europe (Ireland)</td>
<td>eu-west-1</td>
<td>3</td>
</tr>
<tr>
<td>Europe (London)</td>
<td>eu-west-2</td>
<td>3</td>
</tr>
<tr>
<td>Europe (Paris)</td>
<td>eu-west-3</td>
<td>3</td>
</tr>
</tbody>
</table>
### Finding an ARN

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.

<table>
<thead>
<tr>
<th>Resource Type</th>
<th>ARN Format / Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance (db)</td>
<td>arn:aws:rds:region:account_number:db:resource_id</td>
</tr>
<tr>
<td></td>
<td>arn:aws:rds:us-east-1:1234567890:db:sample-db-instance</td>
</tr>
<tr>
<td>Cluster (cluster)</td>
<td>arn:aws:rds:region:account_number:cluster:resource_id</td>
</tr>
<tr>
<td></td>
<td>arn:aws:rds:us-east-1:1234567890:cluster:sample-db-cluster</td>
</tr>
<tr>
<td>Cluster parameter group</td>
<td>arn:aws:rds:region:account_number:cluster-pg:resource_id</td>
</tr>
<tr>
<td>Security group (secgrp)</td>
<td>arn:aws:rds:region:account_number:secgrp:resource_id</td>
</tr>
<tr>
<td>Subnet group (subgrp)</td>
<td>arn:aws:rds:region:account_number:subgrp:resource_id</td>
</tr>
<tr>
<td></td>
<td>arn:aws:rds:us-east-1:1234567890:subgrp:sample-subnet-10</td>
</tr>
</tbody>
</table>

### 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.

![Configurations and status](image)

### 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.

<table>
<thead>
<tr>
<th>AWS CLI Command</th>
<th>ARN Property</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>describe-db-instances</code></td>
<td><code>DBInstanceArn</code></td>
</tr>
<tr>
<td><code>describe-db-clusters</code></td>
<td><code>DBClusterArn</code></td>
</tr>
<tr>
<td><code>describe-db-parameter-groups</code></td>
<td><code>DBParameterGroupArn</code></td>
</tr>
<tr>
<td><code>describe-db-cluster-parameter-groups</code></td>
<td><code>DBClusterParameterGroupArn</code></td>
</tr>
<tr>
<td><code>describe-db-security-groups</code></td>
<td><code>DBSecurityGroupArn</code></td>
</tr>
<tr>
<td><code>describe-db-snapshots</code></td>
<td><code>DBSnapshotArn</code></td>
</tr>
<tr>
<td><code>describe-db-cluster-snapshots</code></td>
<td><code>DBClusterSnapshotArn</code></td>
</tr>
<tr>
<td><code>describe-db-subnet-groups</code></td>
<td><code>DBSubnetGroupArn</code></td>
</tr>
</tbody>
</table>

#### 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:

```bash
aws docdb describe-db-clusters
  --db-cluster-identifier sample-cluster
  --query 'DBClusters[*].DBClusterArn'
```

For Windows:

```bash
aws docdb describe-db-clusters
  --db-cluster-identifier sample-cluster
  --query 'DBClusters[*].DBClusterArn'
```

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

```json
[
  "arn:aws:rds:us-east-1:123456789012:cluster:sample-cluster"
]
```
Example - Finding ARNs for multiple parameter groups

For Linux, macOS, or Unix:

```bash
aws docdb describe-db-cluster-parameter-groups
   --query 'DBClusterParameterGroups[*].DBClusterParameterGroupArn'
```

For Windows:

```bash
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: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. 285).

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: _ . : / = + @ (Java regex: "^[[:\p{L}[:\p{Z}\p{N}_.:/=+\-]]*$")
- 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

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 \\n  --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**

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 \\
```

For Windows:

```
aws docdb list-tags-for-resource ^ \\
```

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

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:

```bash
aws docdb remove-tags-from-resource \
  --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster \
  --tag-keys key1 key3
```

For Windows:

```bash
aws docdb remove-tags-from-resource ^
  --resource-name arn:aws:rds:us-east-1:1234567890:cluster:sample-cluster \
  --tag-keys key1 key3
```

The removed-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. Such patching occurs infrequently (typically once every few months) and seldom requires more than a fraction of your maintenance window. 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. 247).

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. 293)
- Applying Amazon DocumentDB Updates (p. 294)
- User-Initiated Updates (p. 296)
- Managing Your Amazon DocumentDB Maintenance Windows (p. 297)

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:
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).

```json
{
    "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

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:

```bash
aws docdb apply-pending-maintenance-action 
  --apply-action system-update 
  --opt-in-type immediate
```

For Windows:

```bash
aws docdb apply-pending-maintenance-action 
  --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:

```bash
aws docdb describe-pending-maintenance-actions 
```

For Windows:

```bash
aws docdb describe-pending-maintenance-actions 
```

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

```json
{
  "PendingMaintenanceActions": [
    {
      "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:

```bash
aws docdb describe-pending-maintenance-actions \
  --filters Name=db-cluster-id,Values=sample-cluster1,sample-cluster2
```

For Windows:

```bash
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. 203)
- Modifying an Amazon DocumentDB Instance (p. 229)

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:

```bash
aws docdb describe-db-instances
   --query 'DBInstances[*].
      [DBClusterIdentifier,DBInstanceIdentifier,PendingModifiedValues]'
```

For Windows:

```bash
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.

```json
[
  [
    "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.

<table>
<thead>
<tr>
<th>Region</th>
<th>UTC Time Block</th>
</tr>
</thead>
<tbody>
<tr>
<td>US East (Ohio)</td>
<td>03:00-11:00</td>
</tr>
<tr>
<td>US East (N. Virginia)</td>
<td>03:00-11:00</td>
</tr>
</tbody>
</table>
### 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.

---

### 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. 203).
- For an instance, see Modifying an Amazon DocumentDB Instance (p. 229).
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. 234)
- Deleting an Amazon DocumentDB Cluster (p. 212)

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. 353).
- **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, and CloudWatch API.

**Topics**

- Monitoring an Amazon DocumentDB Cluster’s Status (p. 303)
- Monitoring an Amazon DocumentDB Instance’s Status (p. 305)
- Monitoring Amazon DocumentDB with CloudWatch (p. 308)
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. 303)
- Monitoring a Cluster's Status Using the AWS Management Console (p. 304)
- Monitoring a Cluster's Status Using the AWS CLI (p. 304)

**Cluster Status Values**

The following table lists the valid values for a cluster's status.

<table>
<thead>
<tr>
<th>Cluster Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>available</td>
<td>The cluster is healthy and available.</td>
</tr>
<tr>
<td>backing-up</td>
<td>The cluster is currently being backed up.</td>
</tr>
<tr>
<td>creating</td>
<td>The cluster is being created. It is inaccessible while it is being created.</td>
</tr>
<tr>
<td>deleting</td>
<td>The cluster is being deleted. It is inaccessible while it is being deleted.</td>
</tr>
<tr>
<td>failing-over</td>
<td>A failover from the primary instance to an Amazon DocumentDB replica is being performed.</td>
</tr>
<tr>
<td>inaccessible-encryption-credentials</td>
<td>The AWS KMS key used to encrypt or decrypt the cluster can't be accessed.</td>
</tr>
<tr>
<td>maintenance</td>
<td>A maintenance update is being applied to the cluster. This status is used for cluster-level maintenance that Amazon DocumentDB schedules well in advance.</td>
</tr>
<tr>
<td>migrating</td>
<td>A cluster snapshot is being restored to a cluster.</td>
</tr>
</tbody>
</table>
Monitoring a Cluster's Status Using the AWS Management Console

When using the AWS CLI to determine the status of a cluster, use the following procedure.

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.

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. 294).

### Topics

- [Instance Status Values](p. 305)
- [Monitoring an Instance's Status Using the AWS Management Console](p. 307)
- [Monitoring an Instance's Status Using the AWS CLI](p. 307)

## 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.

<table>
<thead>
<tr>
<th>Instance status</th>
<th>Billed</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>available</td>
<td>Billed</td>
<td>The instance is healthy and available.</td>
</tr>
<tr>
<td>backing-up</td>
<td>Billed</td>
<td>The instance is currently being backed up.</td>
</tr>
<tr>
<td>configuring-log-exports</td>
<td>Billed</td>
<td>Publishing log files to Amazon CloudWatch Logs is being enabled or disabled for this instance.</td>
</tr>
<tr>
<td>creating</td>
<td>Not billed</td>
<td>The instance is being created. The instance is not accessible while it is being created.</td>
</tr>
<tr>
<td>deleting</td>
<td>Not billed</td>
<td>The instance is being deleted.</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Instance status</th>
<th>Billed</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>failed</td>
<td>Not billed</td>
<td>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.</td>
</tr>
<tr>
<td>inaccessible-encryption-credentials</td>
<td>Not billed</td>
<td>The AWS KMS key that is used to encrypt or decrypt the instance could not be accessed.</td>
</tr>
<tr>
<td>incompatible-network</td>
<td>Not billed</td>
<td>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.</td>
</tr>
<tr>
<td>maintenance</td>
<td>Billed</td>
<td>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.</td>
</tr>
<tr>
<td>modifying</td>
<td>Billed</td>
<td>The instance is being modified because of a request to modify the instance.</td>
</tr>
<tr>
<td>rebooting</td>
<td>Billed</td>
<td>The instance is being rebooted because of a request or an Amazon DocumentDB process that requires the rebooting of the instance.</td>
</tr>
<tr>
<td>renaming</td>
<td>Billed</td>
<td>The instance is being renamed because of a request to rename it.</td>
</tr>
<tr>
<td>resetting-master-credentials</td>
<td>Billed</td>
<td>The master credentials for the instance are being reset because of a request to reset them.</td>
</tr>
<tr>
<td>restore-error</td>
<td>Billed</td>
<td>The instance encountered an error attempting to restore to a point-in-time or from a snapshot.</td>
</tr>
<tr>
<td>starting</td>
<td>Billed for storage</td>
<td>The instance is starting.</td>
</tr>
<tr>
<td>stopped</td>
<td>Billed for storage</td>
<td>The instance is stopped.</td>
</tr>
<tr>
<td>stopping</td>
<td>Billed for storage</td>
<td>The instance is being stopped.</td>
</tr>
<tr>
<td>storage-full</td>
<td>Billed</td>
<td>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.</td>
</tr>
</tbody>
</table>
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.

2. In the navigation pane, choose Instances.
3. In the Instance 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 following.

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"
  ]
]
```
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

- Viewing CloudWatch Data (p. 308)
- Amazon DocumentDB Metrics (p. 312)
- Amazon DocumentDB Dimensions (p. 315)
- Monitoring Database Connections (p. 315)

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.

CloudWatch Instance Metrics That Appear in the Amazon DocumentDB Console

To monitor the health and performance of your Amazon DocumentDB instance, you can view the following instance metrics in the Amazon DocumentDB console.

- ChangeStreamLogSize
- CPUUtilization
- DatabaseConnections
- FreeLocalStorage
- FreeableMemory
- WriteIOPS
- ReadIOPS
- WriteLatency
- ReadLatency
- WriteThroughput
- ReadThroughput
- NetworkThroughput
- NetworkTransmitThroughput
- NetworkReceiveThroughput
- EngineUptime
• SwapUsage
• DiskQueueDepth
• BufferCacheHitRatio
• BackupRetentionPeriodStorageUsed
• SnapshotStorageUsed
• TotalBackupStorageBilled

Viewing CloudWatch Data (Amazon DocumentDB Console)

To view CloudWatch metrics using the Amazon DocumentDB console

2. In the navigation pane, choose Instances. Then, from the list of instances, choose the name of the instance that you want metrics for.

   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.

To view graphical representations of the instances’ metrics, scroll down the page until you get to the CloudWatch section. 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 2 of the 17 CloudWatch metrics in the Amazon DocumentDB console.

![CloudWatch Metrics](image)

Viewing CloudWatch Data (AWS CLI)

You can view CloudWatch data for Amazon DocumentDB using 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
- Kilobytes
- Megabits
- Gigabits
- Terabits
- Percent
- Count
- Bytes/Second
- Kilobytes/Second
- Megabytes/Second
- Gigabytes/Second
- Terabytes/Second
- Bits/Second
- Kilobits/Second
- Megabits/Second
- Gigabits/Second
- Terabits/Second
- Count/Second
- None

**Example**

The following example finds the maximum CPUUtilization for a 2-hour period taking a sample every 60 seconds.

For Linux, macOS, or Unix:

```bash
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:

```bash
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.

```json
{
  "Label": "CPUUtilization",
  "Datapoints": [
    {
      "Unit": "Percent",
      "Maximum": 4.49152542374361,
      "Timestamp": "2019-02-11T05:51:00Z"
    },
    {
      "Unit": "Percent",
      "Maximum": 4.25000000000485,
      "Timestamp": "2019-02-11T06:44:00Z"
    },
    ********** some output omitted for brevity **********
    {
      "Unit": "Percent",
      "Maximum": 4.33333333331878,
```

311
Amazon DocumentDB Metrics

The following Amazon DocumentDB metrics are available through Amazon CloudWatch. You can use them to monitor various aspects of your Amazon DocumentDB deployment, such as replication lag, disk usage, CPU usage, and more. Amazon DocumentDB sends metrics to CloudWatch only when they have a non-zero value.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BackupRetentionPeriodStorageUsed</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>BufferCacheHitRatio</strong></td>
<td>The percentage of requests that are served by the buffer cache.</td>
</tr>
<tr>
<td><strong>ChangeStreamLogSize</strong></td>
<td>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 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. 340).</td>
</tr>
<tr>
<td><strong>CPUUtilization</strong></td>
<td>The percentage of CPU used by an instance.</td>
</tr>
<tr>
<td><strong>DatabaseConnections</strong></td>
<td>The number of connections to an instance.</td>
</tr>
<tr>
<td><strong>DBInstanceReplicaLag</strong></td>
<td>The amount of lag, in milliseconds, when replicating updates from the primary instance to a replica instance.</td>
</tr>
<tr>
<td><strong>DBClusterReplicaLagMaximum</strong></td>
<td>The maximum amount of lag, in milliseconds, between the primary instance and each Amazon DocumentDB instance in the cluster.</td>
</tr>
<tr>
<td><strong>DBClusterReplicaLagMinimum</strong></td>
<td>The minimum amount of lag, in milliseconds, between the primary instance and each replica instance in the cluster.</td>
</tr>
<tr>
<td><strong>DiskQueueDepth</strong></td>
<td>The number of outstanding read/write requests waiting to access the disk.</td>
</tr>
<tr>
<td><strong>EngineUptime</strong></td>
<td>The amount of time, in seconds, that the instance has been running.</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FreeableMemory</td>
<td>The amount of available random access memory, in bytes.</td>
</tr>
<tr>
<td>FreeLocalStorage</td>
<td>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.</td>
</tr>
<tr>
<td>NetworkReceiveThroughput</td>
<td>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.</td>
</tr>
<tr>
<td>NetworkThroughput</td>
<td>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.</td>
</tr>
<tr>
<td>NetworkTransmitThroughput</td>
<td>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.</td>
</tr>
<tr>
<td>ReadIOPS</td>
<td>The average number of disk read I/O operations per second. Amazon DocumentDB reports read and write IOPS separately, and on one-minute intervals.</td>
</tr>
<tr>
<td>ReadLatency</td>
<td>The average amount of time taken per disk I/O operation.</td>
</tr>
<tr>
<td>ReadThroughput</td>
<td>The average number of bytes read from disk per second.</td>
</tr>
<tr>
<td>SnapshotStorageUsed</td>
<td>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.</td>
</tr>
<tr>
<td>SwapUsage</td>
<td>The amount of swap space used on the instance.</td>
</tr>
<tr>
<td>TotalBackupStorageBilled</td>
<td>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.</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>VolumeBytesUsed</td>
<td>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.</td>
</tr>
<tr>
<td>VolumeReadIOPs</td>
<td>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. For example, if the billed read operations returns 13,686, then the billed read operations per second is 45 (13,686 / 300 = 45.62). 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.</td>
</tr>
<tr>
<td>VolumeWriteIOPs</td>
<td>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. For example, if the billed write operations returns 13,686, then the billed write operations per second is 45 (13,686 / 300 = 45.62).</td>
</tr>
<tr>
<td>WriteIOPS</td>
<td>The average number of disk write I/O operations per second. Read and write IOPS are reported separately, on 1-minute intervals.</td>
</tr>
<tr>
<td>WriteLatency</td>
<td>The average amount of time, in milliseconds, taken per disk I/O operation.</td>
</tr>
</tbody>
</table>
| WriteThroughput    | The average number of bytes written to disk per second.  


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.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBClusterIdentifier</td>
<td>Filters the data that you request for a specific Amazon DocumentDB cluster.</td>
</tr>
<tr>
<td>DBClusterIdentifier, Role</td>
<td>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.</td>
</tr>
<tr>
<td>DBInstanceIdentifier</td>
<td>Filters the data that you request for a specific database instance.</td>
</tr>
</tbody>
</table>

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. 321)](#).

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. 317)**
Supported Operations

Amazon DocumentDB profiler supports the following operations:

- aggregate
- count
- delete
- distinct
- find (OP_QUERY and command)
- findAndModify
- insert
- update

Limitations

The profiler does not currently support the logging of find operators that return a cursor.

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. 261)
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. 262).
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. 261) and Creating an Amazon DocumentDB Cluster (p. 189).
2. Using an available custom cluster parameter group, modify the following parameters. For more information, see Modifying Amazon DocumentDB Cluster Parameter Groups (p. 262).
• 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 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 associate it with 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. Run the following command and review 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. 261).

   ```
   aws docdb describe-db-cluster-parameter-groups --query 'DBClusterParameterGroups[*].
   [DBClusterParameterGroupName,DBParameterGroupFamily]
   
   In the following output (in JSON format), only sample-parameter-group meets both criteria.
   
   [ 
     [ 
       "custom-pg-profiler",
       "docdb3.6"
     ],
     [ 
       "default.docdb3.6",
       "docdb3.6"
     ]
   ]
   
   2. 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.
Disabling the Profiler

You can disable the `profiler` parameter using either the AWS Management Console or AWS CLI.

**Using the AWS Management Console**

The following procedure uses the AWS Management Console to disable Amazon DocumentDB profiler.

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. On the **Actions** menu, choose **Modify**.
4. Locate the `profiler` parameter. Choose the button to the left of its name, and then choose **Edit**.
5. In the **Modify profiler** dialog box, choose **disabled** in the list.

**Disabling the Amazon DocumentDB Profiler**

To disable the profiler, you disable both the `profiler` parameter and the export of `profiler` logs to CloudWatch Logs.

Topics

- Disabling the Profiler (p. 319)
- Disabling Profiler Logs Export (p. 320)
6. Choose **Modify cluster parameter**.

### Using the AWS CLI

To disable profiler on a cluster using the AWS CLI, modify the cluster as shown here.

```bash
aws docdb modify-db-cluster-parameter-group \
   --db-cluster-parameter-group-name custom-pg-profiler \
   --parameters ParameterName=profiler,Value=disabled,ApplyType=immediate
```

### Disabling Profiler Logs Export

You can disable exporting profiler logs to CloudWatch Logs by using either the AWS Management Console or AWS CLI.

#### Using the AWS Management Console

The following procedure uses the AWS Management Console to disable Amazon DocumentDB exporting logs to CloudWatch.

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 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:

```bash
aws docdb modify-db-cluster \
   --db-cluster-identifier sample-cluster \
   --cloudwatch-logs-export-configuration '{"DisableLogTypes":["profiler"]}'
```

For Windows:

```bash
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 (JSON format).

```json
{
```
Accessing Your Amazon DocumentDB Profiler Logs

Follow these steps to access your profile logs on Amazon 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 profile logs for your cluster, in the list, choose /aws/docdb/yourClusterName/profile.

The profile logs for each of your instances are available under each of the respective instance names.

Common Queries

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.

Topics
- Get the 10 Slowest Operations on a Specified Collection (p. 322)
Get All the Update Operations on a Collection That Took More Than 60 ms (p. 322)
Get the 10 Slowest Operations in the Last Month (p. 322)
Get All the Queries with a COLLSCAN Plan Summary (p. 322)

Get the 10 Slowest Operations on a Specified Collection

```bash
filter ns="test.foo" | sort millis desc | limit 10
```

Get All the Update Operations on a Collection That Took More Than 60 ms

```bash
filter millis > 60 and op = "update"
```

Get the 10 Slowest Operations in the Last Month

```bash
sort millis desc | limit 10
```

Get All the Queries with a COLLSCAN Plan Summary

```bash
filter planSummary="COLLSCAN"
```
Developing with Amazon DocumentDB

These sections cover development using Amazon DocumentDB (with MongoDB compatibility).

Topics
- Connecting Programmatically to Amazon DocumentDB (p. 323)
- Using Change Streams with Amazon DocumentDB (p. 340)
- Connecting to Amazon DocumentDB as a Replica Set (p. 347)
- Connecting to an Amazon DocumentDB Cluster from Outside an Amazon VPC (p. 350)
- Connecting to an Amazon DocumentDB Cluster Using AWS Cloud9 (p. 351)

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. 104).

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. 350).

Important
The certificate authority (CA) certificate for Amazon DocumentDB 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 and certificates, see Updating Your Amazon DocumentDB TLS Certificates (p. 139).

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. 323)
- Connecting with TLS Enabled (p. 325)
- Connecting with TLS Disabled (p. 334)

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**

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.

```bash
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. 262).

**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.

```bash
aws docdb describe-db-cluster-parameters \
  --db-cluster-parameter-group-name sample-parameter-group
```

324
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. 325)
- Connecting with TLS Disabled (p. 334)

## 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.

**Python**

The following code demonstrates how to connect to Amazon DocumentDB using Python when TLS is enabled.

```python
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/?ssl=true&ssl_ca_certs=rds-combined-ca-bundle.pem&replicaSet=rs0&readPreference=secondaryPreferred')

##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.

```javascript
var MongoClient = require('mongodb').MongoClient,
    f = require('util').format,
    fs = require('fs');

//Specify the Amazon DocumentDB cert
var ca = [fs.readFileSync("rds-combined-ca-bundle.pem")];

//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?ssl=true&replicaSet=rs0&readPreference=secondaryPreferred',
    { sslValidate: true,
      sslCA: ca,
      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()
          });
      });
    });
```
Amazon DocumentDB Developer Guide
Connecting with TLS Enabled

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';
$SSL_DIR = "/home/ubuntu";
$SSL_FILE = "rds-combined-ca-bundle.pem";
//Specify the Amazon DocumentDB cert
$ctx = stream_context_create(array(
"ssl" => array(
"cafile" => $SSL_DIR . "/" . $SSL_FILE,
))
);
//Create a MongoDB client and open connection to Amazon DocumentDB
$client = new MongoDB\Client("mongodb://<sample-user>:<password>@samplecluster.node.us-east-1.docdb.amazonaws.com:27017", array("ssl" => true),
array("context" => $ctx));
//Specify the database and collection to be used
$col = $client->sample-database->sample-collection;
//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"

327


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?ssl=true&replicaSet=rs0&readPreference=%s"
)

func main() {
    connectionURI := fmt.Sprintf(connectionStringTemplate, username, password, clusterEndpoint, readPreference)

tlsConfig, err := getCustomTLSConfig(caFilePath)
if err != nil {
    log.Fatal("Failed getting TLS configuration: %v", err)
}

client, err := mongo.NewClient(options.Client().ApplyURI(connectionURI).SetTLSConfig(tlsConfig))
if err != nil {
    log.Fatal("Failed to create client: %v", err)
}

cancel := context.WithTimeout(context.Background(), connectTimeout*time.Second)
def er cancel()

err = client.Connect(ctx)
if err != nil {
    log.Fatal("Failed to connect to cluster: %v", err)
}

// Force a connection to verify our connection string
err = client.Ping(ctx, nil)
if err != nil {
    log.Fatal("Failed to ping cluster: %v", err)
}

fmt.Println("Connected to DocumentDB!")

collection := client.Database("sample-database").Collection("sample-collection")

cancel = context.WithTimeout(context.Background(), queryTimeout*time.Second)
def er cancel()

res, err := collection.InsertOne(ctx, bson.M{"name": "pi", "value": 3.14159})
if err != nil {
    log.Fatal("Failed to insert document: %v", err)
}

id := res.InsertedID
log.Printf("Inserted document ID: %s", id)
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:


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.
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>


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 ${storepassword} | grep Alias | cut -d " " -f3- | while read alias
  expiry="keytool -list -v -keystore "$truststore" -storepass ${storepassword} -alias "$alias" | grep Valid | perl -ne 'if(/until: (.*?)\n/) { print "$1\n"; }'"
  echo " Certificate $alias expires in '$expiry'"
done
```

330
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.

```java
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?
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>", truststorePassword = "<truststorePassword>";
        System.setProperty("javax.net.ssl.trustStore", truststore);
        System.setProperty("javax.net.ssl.trustStorePassword", truststorePassword);

        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();
        }
    }
}
```
The following code demonstrates how to connect to Amazon DocumentDB using C# / .NET when TLS is enabled.

```csharp
using System;
using System.Text;
using System.Linq;
using System.Collections.Generic;
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}/sample-database?ssl=true&replicaSet=rs0&readpreference={3}";
            string username = "<sample-user>";
            string password = "<password>";
            string readPreference = "secondaryPreferred";
            string clusterEndpoint="sample-cluster.node.us-east-1.docdb.amazonaws.com:27017";
            string connectionString = String.Format(template, username, password, clusterEndpoint, readPreference);

            string pathToCAFile = "<path_to_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(pathToCAFile);
            try
            {
                localTrustStore.Open(OpenFlags.ReadWrite);
                localTrustStore.AddRange(certificateCollection);
            }
            catch (Exception ex)
            {
                Console.WriteLine("Root certificate import failed: " + ex.Message);
                throw;
            }
            finally
            {
                localTrustStore.Close();
            }

            var settings = MongoClientSettings.FromUrl(new MongoUrl(connectionString));
            var client = new MongoClient(settings);

            var database = client.GetDatabase("sample-database");
            var collection = database.GetCollection<BsonDocument>("sample-collection");
        }
    }
}
```
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.

```
mongo --ssl --host sample-cluster.node.us-east-1.docdb.amazonaws.com:27017 --sslCAFile 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.

```
library(mongolite)

#Include the mongolite library.
library(mongolite)

mongourl <- paste("mongodb://<sample-user>:<password>@sample-cluster.node.us-
est-1.docdb.amazonaws.com:27017/test2?ssl=true&","readPreference=secondaryPreferred&replicaSet=rs0", sep="")

#Create a MongoDB client, open a connection to Amazon DocumentDB as a replica set and specify the read preference as secondary preferred
client <- mongo(url = mongourl, options = ssl_options(weak_cert_validation = F, ca = <path to '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 = {
  :ssl: true,
  :ssl_ca_file: 'rds-combined-ca-bundle.pem',
  :ssl_cert_reqs: :require_client
}
client = Mongo::Client.new(client_host, client_options)
```

```ruby
var docToInsert = new BsonDocument { { "pi", 3.14159 } };
collection.InsertOne(docToInsert);
```
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.

```python
begin
    ## 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/?replicaSet=rs0&readPreference=secondaryPreferred')

    ## 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)
end
```
Amazon DocumentDB Developer Guide
Connecting with TLS Disabled
##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.useast-1.docdb.amazonaws.com:27017/sample-database?
replicaSet=rs0&readPreference=secondaryPreferred',
{
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>@samplecluster.node.us-east-1.docdb.amazonaws.com:27017");
//Specify the database and collection to be used
$col = $client->sample-database->sample-collection;
//Insert a single document
$result = $col->insertOne( [ 'hello' => 'Amazon DocumentDB'] );

335


The following code demonstrates how to connect to Amazon DocumentDB using Go when TLS is disabled.

```go
package main

import {
    "context"
    "fmt"
    "log"
    "time"

    "go.mongodb.org/mongo-driver/bson"
    "go.mongodb.org/mongo-driver/mongo"
    "go.mongodb.org/mongo-driver/mongo/options"
}

const {
    // Timeout operations after N seconds
    connectTimeout = 5
    queryTimeout    = 30
    username        = "<sample-user>"
    password        = "<password>"
    clusterEndpoint = "sample-cluster.node.us-east-1.docdb.amazonaws.com:27017"

    // Which instances to read from
    readPreference           = "secondaryPreferred"
    connectionStringTemplate = "mongodb://%s:%s@%s/sample-database?replicaSet=rs0&readpreference=%s"
}

func main() {
    connectionURI := fmt.Sprintf(connectionStringTemplate, username, password, clusterEndpoint, readPreference)

    client, err := mongo.NewClient(options.Client().ApplyURI(connectionURI))
    if err != nil {
        log.Fatal("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.Fatal("Failed to connect to cluster: %v", err)
    }

    // Force a connection to verify our connection string
    err = client.Ping(ctx, nil)
    if err != nil {
        log.Fatal("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)
def cancel()

res, err := collection.InsertOne(ctx, bson.M("name": "pi", "value": 3.14159))
if err != nil {
    log.Fatal("Failed to insert document: %v", err)
}

id := res.InsertedID
log.Printf("Inserted document ID: %s", id)

collection := client.Database("sample-database").Collection("sample-collection")

ctx, cancel = context.WithTimeout(context.Background(), queryTimeout*time.Second)
def cancel()

cur, err := collection.Find(ctx, bson.D{})
if err != nil {
    log.Fatal("Failed to run find query: %v", err)
}
def 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)
    }
}

}
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.Net.Security;
using MongoDB.Driver;
using MongoDB.Bson;

namespace CSharpSample
{
    class Program
    {
        static void Main(string[] args)
        {
            string template = "mongodb://{0}:{1}@{2}/sample-database?
&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("sample-database");
var collection = database.GetCollection<BsonDocument>("sample-collection");
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>',
ssl: true,
ssl_verify: true,
ssl_ca_cert: '<path to rds-combined-ca-bundle.pem>
}

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.stringify(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
- Cross-Region replication
- Full-text search with Amazon Elasticsearch Service (Amazon ES)
- 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.

Topics
- Supported Operations (p. 341)
- Billing (p. 341)
- Limitations (p. 341)
- Enabling Change Streams (p. 341)
- Example: Using Change Streams with Python (p. 343)
- Full Document Lookup (p. 343)
- Resuming a Change Stream (p. 344)
- Modifying the Change Stream Log Retention Duration (p. 345)
Supported Operations

Amazon DocumentDB supports the following operations for change streams:

- All change events supported in the MongoDB 3.6 `db.collection.watch()` API.
- Full document lookup for updates.
- Aggregation stages: `$match`, `$project`, `$redact`, and `$addFields` ($replaceRoot is not currently supported).
- Resuming a change stream.

Billing

The Amazon DocumentDB change streams feature is disabled by default and does not incur any additional charges until the feature is enabled and used. You can use the `modifyChangeStreams` API operation to enable this feature for collection in the cluster. Using change streams in a cluster incurs additional IOPS and storage costs. For more information, 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. 343) section).
- Events written to a change stream for a collection are available for up to 24 hours (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.

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: ""}]}]});
Example: Using Change Streams with Python

The following is an example of using an Amazon DocumentDB change stream with Python.

```python
from pymongo import MongoClient, ReadPreference

client = MongoClient("your-cluster-endpoint")
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
result = coll.insert_one({'x': 1})

# Read the next change event from the stream (if any)
stream.try_next()

# Output:
{
  '_id': {'_data': '015daf94f600000002010000002000009025'},
  '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
stream.try_next()

# Generate a new change event by updating a document
result = coll.update_one({'x': 1}, {'$set': {'x': 2}})

# Read the next change event from the stream (if any)
stream.try_next()

# Output:
{
  '_id': {'_data': '015daf99d4000000010100000010000009025'},
  'clusterTime': Timestamp(1571789268, 1),
  'documentKey': {'_id': ObjectId('5daf9502ea258751778163d7')},
  'fullDocument': {'_id': ObjectId('5daf9502ea258751778163d7'), 'x': 1},
  'ns': {'coll': 'foo', 'db': 'bar'},
  'operationType': 'update',
  'updateDescription': {'removedFields': [], 'updatedFields': {'x': 2}}
}
```

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.
#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.

```python
#Generate a new change event by updating a document
result = coll.update_one({'x': 3}, {'$set': {'x': 4}})

event = stream.try_next()

#Python provides a nice shortcut for getting a stream's resume token
stream.resume_token

#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 = coll.watch(full_document='updateLookup', resume_after=token)

#Our first change event is the update with the specified _id
stream.try_next()

#Output:
{'_id': {'_data': '015daf9d190000000101000000100009025'},
 'clusterTime': Timestamp(1571790105, 1),
 'documentKey': {'_id': ObjectID('5daf94f6ea258751778163d6')},
 'fullDocument': {'_id': ObjectID('5daf94f6ea258751778163d6'), 'x': 5},
 'ns': {'coll': 'foo', 'db': 'bar'},
 'operationType': 'update',
 'updateDescription': {'removedFields': [], 'updatedFields': {'x': 5}}}

#Followed by the insert
stream.try_next()

#Output:
{'_id': {'_data': '015daf9d820000000101000000100009025'},
 'clusterTime': Timestamp(1571790216, 1),
 'documentKey': {'_id': ObjectID('5daf9b8ea258751778163d9')},
 'fullDocument': {'_id': ObjectID('5daf9b8ea258751778163d9'), 'y': 5}}
```
Modifying the Change Stream Log Retention Duration

You can modify the change stream log retention duration to be between 1 hour and 24 hours using the AWS Management Console or the AWS CLI.

Using the AWS Management Console

To modify the change stream log retention duration,
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. 259).
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 24 hours.
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. 259).
- `--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:

```bash
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:

```bash
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).

```json
{
    "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:

```bash
aws docdb describe-db-cluster-parameters \
    --db-cluster-parameter-group-name sample-parameter-group
```

For Windows:

```bash
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).

```json
{
    "Parameters": [
        {
            "ParameterName": "audit_logs",
            "ParameterValue": "disabled",
            "Description": "Enables auditing on cluster.",
            "Source": "system",
            "ApplyType": "dynamic",
            "DataType": "string",
            "AllowedValues": "enabled,disabled",
            "IsModifiable": true,
```
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.

For more information about Amazon DocumentDB endpoints, see Amazon DocumentDB Endpoints (p. 7).
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.

```python
## 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 topography 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. 11).

```python
## 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. 308).

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. 49)
• Amazon DocumentDB Quotas and Limits (p. 353)

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. 357).

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.

```python
## 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

```python
secondaryPreferred = pymongo.MongoClient('mongodb://<username>:<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.

```python
# 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.

```bash
> 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.cluster-cu52jq5kfdgg.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.
Connecting to an Amazon DocumentDB Cluster Using AWS Cloud9

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.

For development, test, and management, you can use AWS Cloud9 to connect to and access your Amazon DocumentDB cluster from your web browser. AWS Cloud9 is a cloud-based integrated development environment (IDE) that lets you write, run, and debug your code using just a browser. It provides a web-based terminal access (Amazon Linux or Ubuntu) so that you can install mongo shell or any MongoDB SDK and connect to Amazon DocumentDB. For more information, see What Is AWS Cloud9? in the AWS Cloud9 User Guide.

Follow these steps to connect to Amazon DocumentDB from AWS Cloud9.

To connect to Amazon DocumentDB from AWS Cloud9

1. Create an Amazon DocumentDB cluster.

   If you already have an Amazon DocumentDB cluster to use, go to the next step. To create a cluster, see Step 1: Create an Amazon DocumentDB Cluster (p. 34). For your cluster, note the AWS Region and the Amazon VPC that the cluster is deployed in. You will want to deploy your AWS Cloud9 environment in the same Region and VPC.
2. **Create an AWS Cloud9 environment.**

   The most common configuration for setting up your AWS Cloud9 environment is to create a development environment hosted by AWS Cloud9 (to provide more options). To do that, follow the detailed steps in *Creating an Amazon EC2 Environment* in the *AWS Cloud9 User Guide*.

3. **Access and use your Amazon DocumentDB cluster using the mongo shell.**

   After your AWS Cloud9 environment is set up, from the terminal, install the mongo shell and query your cluster using the instructions in *Step 3: Access and Use Your Amazon DocumentDB Cluster Using the mongo Shell* (p. 42).

   Every AWS Cloud9 development environment that is associated with an Amazon VPC must meet the specific requirements described in *VPC Settings for AWS Cloud9 Development Environments*. We recommend that you deploy the AWS Cloud9 environment in the same VPC as your Amazon DocumentDB cluster.

   Additionally, configure the security groups for your Amazon DocumentDB cluster to accept inbound connections from your AWS Cloud9 environment. If your security groups are not set up correctly, you will experience connection timeouts and won't be able to connect. For more information, see *Troubleshooting Amazon DocumentDB* (p. 360).
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. 353)
- Supported Regions (p. 354)
- Regional Quotas (p. 354)
- Aggregation Limits (p. 356)
- Cluster Limits (p. 356)
- Instance Limits (p. 357)
- Naming Constraints (p. 358)
- TTL Constraints (p. 359)

Supported Instance Types

Amazon DocumentDB supports on-demand instances and the following instance types:

- Memory Optimized:
  - Current Generation:
    - R5 instance types: `db.r5.large`, `db.r5.2xlarge`, `db.r5.4xlarge`, `db.r5.12xlarge`, `db.r5.24xlarge`.
    - R4 instance types: `db.r4.large`, `db.r4.2xlarge`, `db.r4.4xlarge`, `db.r4.8xlarge`, `db.r4.16xlarge`.

For more information on the supported instance types and their specifications, see Instance Class Specifications (p. 220).
Supported Regions

Amazon DocumentDB is available in the following AWS regions:

<table>
<thead>
<tr>
<th>Region Name</th>
<th>Region</th>
<th>Availability Zones (compute)</th>
</tr>
</thead>
<tbody>
<tr>
<td>US East (Ohio)</td>
<td>us-east-2</td>
<td>3</td>
</tr>
<tr>
<td>US East (N. Virginia)</td>
<td>us-east-1</td>
<td>6</td>
</tr>
<tr>
<td>US West (Oregon)</td>
<td>us-west-2</td>
<td>4</td>
</tr>
<tr>
<td>Asia Pacific (Mumbai)</td>
<td>ap-south-1</td>
<td>3</td>
</tr>
<tr>
<td>Asia Pacific (Seoul)</td>
<td>ap-northeast-2</td>
<td>3</td>
</tr>
<tr>
<td>Asia Pacific (Singapore)</td>
<td>ap-southeast-1</td>
<td>3</td>
</tr>
<tr>
<td>Asia Pacific (Sydney)</td>
<td>ap-southeast-2</td>
<td>3</td>
</tr>
<tr>
<td>Asia Pacific (Tokyo)</td>
<td>ap-northeast-1</td>
<td>3</td>
</tr>
<tr>
<td>Canada (Central)</td>
<td>ca-central-1</td>
<td>3</td>
</tr>
<tr>
<td>Europe (Frankfurt)</td>
<td>eu-central-1</td>
<td>3</td>
</tr>
<tr>
<td>Europe (Ireland)</td>
<td>eu-west-1</td>
<td>3</td>
</tr>
<tr>
<td>Europe (London)</td>
<td>eu-west-2</td>
<td>3</td>
</tr>
<tr>
<td>Europe (Paris)</td>
<td>eu-west-3</td>
<td>3</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Resource</th>
<th>AWS Default Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clusters</td>
<td>40</td>
</tr>
<tr>
<td>Cluster parameter groups</td>
<td>50</td>
</tr>
<tr>
<td>Event subscriptions</td>
<td>20</td>
</tr>
<tr>
<td>Instances</td>
<td>40</td>
</tr>
<tr>
<td>Manual cluster snapshots</td>
<td>100</td>
</tr>
<tr>
<td>Read replicas per cluster</td>
<td>15</td>
</tr>
<tr>
<td>Subnet groups</td>
<td>50</td>
</tr>
<tr>
<td>Subnets per subnet group</td>
<td>20</td>
</tr>
</tbody>
</table>
Regional Quotas

<table>
<thead>
<tr>
<th>Resource</th>
<th>AWS Default Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tags per resource</td>
<td>50</td>
</tr>
<tr>
<td>VPC security groups per instance</td>
<td>5</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of supported stages</td>
<td>500</td>
</tr>
</tbody>
</table>

Cluster Limits

The following table describes Amazon DocumentDB cluster limits.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster size (sum of all collections and indexes)</td>
<td>64 TB</td>
</tr>
<tr>
<td>Collection size (sum of all collections can't exceed cluster limit) – does not include the index size</td>
<td>32 TB</td>
</tr>
<tr>
<td>Collections per cluster</td>
<td>100,000</td>
</tr>
<tr>
<td>Databases per cluster</td>
<td>100,000</td>
</tr>
<tr>
<td>Database size (sum of all databases can't exceed cluster limit)</td>
<td>64 TB</td>
</tr>
<tr>
<td>Document nesting depth</td>
<td>100 levels</td>
</tr>
<tr>
<td>Document size</td>
<td>16 MB</td>
</tr>
<tr>
<td>Index key size (for arrays, the entire array must be less than 2,048 bytes)</td>
<td>2,048 bytes</td>
</tr>
<tr>
<td>Indexes per collection</td>
<td>64</td>
</tr>
<tr>
<td>Keys in a compound index</td>
<td>32</td>
</tr>
<tr>
<td>Maximum number of writes in a single batch command</td>
<td>100,000</td>
</tr>
<tr>
<td>Number of users per cluster</td>
<td>1000</td>
</tr>
</tbody>
</table>
## Instance Limits

The following table describes Amazon DocumentDB limits per instance.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of concurrent connections by instance type</td>
<td></td>
</tr>
<tr>
<td>db.r5.large</td>
<td>1,100</td>
</tr>
<tr>
<td>db.r4.large</td>
<td>1,100</td>
</tr>
<tr>
<td>db.r5.xlarge</td>
<td>2,700</td>
</tr>
<tr>
<td>db.r4.xlarge</td>
<td>2,700</td>
</tr>
<tr>
<td>db.r5.2xlarge</td>
<td>4,500</td>
</tr>
<tr>
<td>db.r4.2xlarge</td>
<td>4,500</td>
</tr>
<tr>
<td>db.r5.4xlarge</td>
<td>4,500</td>
</tr>
<tr>
<td>db.r4.4xlarge</td>
<td>4,500</td>
</tr>
<tr>
<td>db.r4.8xlarge</td>
<td>4,500</td>
</tr>
<tr>
<td>db.r5.12xlarge</td>
<td>4,500</td>
</tr>
<tr>
<td>db.r4.16xlarge</td>
<td>4,500</td>
</tr>
<tr>
<td>db.r5.24xlarge</td>
<td>4,500</td>
</tr>
<tr>
<td>Maximum number of open cursors per instance</td>
<td>450</td>
</tr>
</tbody>
</table>
# Naming Constraints

The following table describes naming constraints in Amazon DocumentDB.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Default Limit</th>
</tr>
</thead>
</table>
| Cluster identifier              | • 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 index name: `<db>..<col>..<index>` | Length is [6–127] characters.                                                                                                                   |
| Index name: `<col>..<index>`    | Length is [3–63] characters.                                                                                                                   |
| Instance identifier             | • 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                 | • Length is [8–100] printable ASCII characters.  
• Can use any printable ASCII characters except for the following:  
  • `/` (forward slash)  
  • `"` (double quotation mark)  
  • `@` (at symbol)                                                                 |
| Master user name                | • 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.                                                                                               |
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.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Default Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>• First character must be a letter.</td>
<td></td>
</tr>
<tr>
<td>• Cannot end with a hyphen or contain two consecutive hyphens.</td>
<td></td>
</tr>
</tbody>
</table>
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

- Cannot Connect to an Amazon DocumentDB Endpoint (p. 360)
- Testing a Connection to an Amazon DocumentDB Instance (p. 363)
- Connecting to an Invalid Endpoint (p. 363)
- Identifying Billable Amazon DocumentDB Resources (p. 364)
- Index Creation (p. 366)
- How Can I See the executionStats for a Query Plan? (p. 367)
- Performance and Resource Utilization (p. 368)

Important

The certificate authority (CA) certificate for Amazon DocumentDB 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 and certificates, see Updating Your Amazon DocumentDB TLS Certificates (p. 139).

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/
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

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. 350). 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 (JSON format).

```
[  "sample-cluster.cluster-corcjozrlsfc.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'
```
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.

```bash
aws docdb describe-db-instances
   --db-instance-identifier sample-cluster-instance
   --query 'DBInstances[*].DBSubnetGroup.VpcId'
```

To verify the Amazon VPC of your Amazon EC2 instance, run the following command.

```bash
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.

```bash
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.

```bash
aws docdb describe-db-clusters
   --db-cluster-identifier sample-cluster
```
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 `port` with the port of your instance):

```
nc -zv cluster-endpoint port
```

For example, the following shows 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.cluster-ccuszbx3pn5e.us-east-1.docdb.amazonaws.com:27017 \
   --sslCAFile rds-combined-ca-bundle.pem \
   --username <user-name> \n   --password <password>
```

The output looks like this:

```
MongoDB shell version v3.6
connecting to: mongodb://sample-cluster.cluster-ccuszbx3pn5e.us-east-1.docdb.amazonaws.com:27017/
2018-11-14T17:21:18.516-0800 I NETWORK [thread1] getaddrinfo("sample-cluster.cluster-ccuszbx3pn5e.us-east-1.docdb.amazonaws.com") failed:
node not known 2018-11-14T17:21:18.537-0800 E QUERY [thread1]
Error: couldn't initialize connection to host sample-cluster.cluster-ccuszbx3pn5e.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:

```bash
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:

```bash
aws docdb describe-db-instances \
  --db-instance-identifier db-cluster-instance \
  --query 'DBInstances[*].[Endpoint.Address,Endpoint.Port]'
```

For more information, see Understanding Amazon DocumentDB Endpoints (p. 279).

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.

Topics
- Using the AWS Management Console (p. 364)
- Using the AWS CLI (p. 365)
- Deleting Unwanted Billable Resources (p. 366)

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

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:

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

For Windows:

```bash
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 (JSON format).

```
"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:

```bash
aws docdb describe-db-cluster-snapshots \
```

For Windows:

```bash
aws docdb describe-db-cluster-snapshots ^
```

Output from this operation looks something like the following (JSON format).
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. 234).

  **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. 212).
- To delete manual snapshots, see Deleting a Cluster Snapshot (p. 178).

Index Creation

The following topics address what to do if your index or background index build fails.

**Topics**

- Index Build Fails (p. 366)
- Background Index Build Fails (p. 367)

**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. 86) 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. 217). With per-second instance pricing, you only pay for the resource you use up to the second.

**Background Index Build Fails**

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.

**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 `<code>$hint</code>` operator to enforce selection of a preferred index.

Run the query that you want to improve under the `explain()` command as follows.

```javascript
db.runCommand({explain: {query document}}).
explain("executionStats").executionStats;
```

The following is an example operation.

```javascript
db.fish.find({}).limit(2).explain("executionStats");
```

Output from this operation looks something like the following.

```json
{
   "queryPlanner" : {
      "plannerVersion" : 1,
      "namespace" : "test.fish",
      "memDB" : "init_memdb",
      "stage" : "executionStats",
      "planType" : "indexScan",
      "index" : "_id"
   }
}
```
"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;

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. 279).

Topics
- How Do I Find and Terminate Long Running or Blocked Queries? (p. 369)
- How Can I See a Query Plan and Optimize a Query? (p. 370)
- How Do I List All Running Operations on an Instance? (p. 371)
- How Do I Know When a Query Is Making Progress? (p. 373)
- How Do I Determine Why a System Suddenly Runs Slowly? (p. 375)
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.

```javascript
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.

```javascript
db.adminCommand({
  aggregate: 1,
  pipeline: [
    {$currentOp: {}},
    {$match: {$or: [{secs_running: {$gt: 10}}, {WaitState: {$exists: true}}]}},
    {$project: {_id:0, opid: 1, secs_running: 1}}],
  cursor: {};
});
```

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

```json
{
```
2. Terminate the query using the `killOp` operation.

```javascript
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.

```javascript
db.runCommand({explain: {<query document>}})
```

The following is an example operation.

```javascript
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).

```json
{
    "queryPlanner": {
        "plannerVersion": 1,
        "namespace": "db.test",
        "winningPlan": {
            "stage": "COLLSCAN"
        }
    },
    "serverInfo": {
        "host": "....",
        "port": ....,
        "version": "...."
}
```
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. 123).)

With the mongo shell, you can use the following query to list all the running operations on an Amazon DocumentDB instance.

```javascript
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).

```json
{
}
```

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.

```json
{
    "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. 123).)

With the mongo shell, you can use the following query to list all the running operations on an Amazon DocumentDB instance.

```javascript
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).

```json
{
}
```
"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,
      "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.

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}}],
```
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.

```javascript
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).

```json
{
    "waitedMS": NumberLong(0),
    "cursor": {
        "firstBatch": [
            {"opid": 201,
             "command": {
                 "aggregate": ...
             },
            "secs_running": 208,
            "WaitState": "IO" },
        ],
    "id": NumberLong(0),
    "ns": "admin.$cmd",
    "ok": 1
}
```
I/O 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.

```json
{
    "inprog" : [
        {
            "client" : "...",
            "desc" : "Conn",
            "active" : true,
            "killPending" : false,
            "opid" : 75,
            "ns" : "...",
            "command" : {
                "find" : "...",
                "filter" : {
                    ...
                },
            },
            "op" : "query",
            "$db" : "test",
            "secs_running" : 9,
            "microsecs_running" : NumberLong(9449440),
            "threadId" : 24773,
            "clientMetaData" : {
                "application" : {
                    "name" : "MongoDB Shell"
                },
                "driver" : {
                    ...
                },
                "os" : {
                    ...
                }
            },
            "WaitState" : "CollectionLock",
            "blockedOn" : "INTERNAL"
        },
        {
            "desc" : "INTERNAL"
        },
        {
            "client" : "...",
            ...
        }
    }
}
```
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.

```javascript
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).

```json
{
    "waitedMS": NumberLong(0),
    "cursor": {
        "firstBatch": [
            {
                "_id": {
                    "desc": "Conn",
                    "ns": "db.test",
                    "WaitState": "CollectionLock"
                },
                "count": 2
            },
            {
                "_id": {
                    "desc": "Conn",
                    "ns": "admin.$cmd"
                },
                "count": 1
            },
            {
                "_id": {
                    "desc": "TTLMonitor"
                },
                "count": 1
            }
        ]
    }
}
```
In the preceding output, two user queries in namespace "db.test" are blocked on collection lock: one query in the namespace "admin.$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. 369)

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. 375)
- To identify and terminate long running queries on a particular instance, see How Do I Find and Terminate Long Running or Blocked Queries? (p. 369)
- To understand whether a query is progressing, see How Do I Know When a Query Is Making Progress? (p. 373)
- To determine why a query takes a long time to run, see How Can I See a Query Plan and Optimize a Query? (p. 370)
- To track long-running queries over time, see Profiling Amazon DocumentDB Operations (p. 316).

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. 347).

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. 220).

- 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. 221).

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. Note, there is a limit
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" : "1.0.202313", "ok" : 1 }
```

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.

```javascript
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:

```javascript
{
  ...
  "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.

```javascript
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.

```javascript
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.

```javascript
db.adminCommand({currentOp: 1, $all: 1});
```

• The following code lists all long running or blocked queries.

```javascript
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: {}});
```

• The following code terminates a query.

```javascript
db.adminCommand({killOp: 1, op: <opid of running or blocked query>});
```

• Use the following code to get an aggregated view of the system state.

```javascript
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 Getting Started (p. 32).

Topics
• Actions (p. 380)
• Data Types (p. 487)
• Common Errors (p. 533)
• Common Parameters (p. 535)

Actions

The following actions are supported:

• AddTagsToResource (p. 382)
• ApplyPendingMaintenanceAction (p. 384)
• CopyDBClusterParameterGroup (p. 386)
• CopyDBClusterSnapshot (p. 388)
• CreateDBCluster (p. 392)
• CreateDBClusterParameterGroup (p. 398)
• CreateDBClusterSnapshot (p. 400)
• CreateDBInstance (p. 402)
• CreateDBSubnetGroup (p. 406)
• DeleteDBCluster (p. 408)
• DeleteDBClusterParameterGroup (p. 410)
• DeleteDBClusterSnapshot (p. 412)
• DeleteDBInstance (p. 414)
• DeleteDBSubnetGroup (p. 416)
• DescribeCertificates (p. 418)
• DescribeDBClusterParameterGroups (p. 420)
• DescribeDBClusterParameters (p. 422)
• DescribeDBClusters (p. 424)
• DescribeDBClusterSnapshotAttributes (p. 426)
- DescribeDBClusterSnapshots (p. 428)
- DescribeDBEngineVersions (p. 431)
- DescribeDBInstances (p. 434)
- DescribeDBSubnetGroups (p. 436)
- DescribeEngineDefaultClusterParameters (p. 438)
- DescribeEventCategories (p. 440)
- DescribeEvents (p. 442)
- DescribeOrderableDBInstanceOptions (p. 445)
- DescribePendingMaintenanceActions (p. 447)
- FailoverDBCluster (p. 449)
- ListTagsForResource (p. 451)
- ModifyDBCluster (p. 453)
- ModifyDBClusterParameterGroup (p. 458)
- ModifyDBClusterSnapshotAttribute (p. 460)
- ModifyDBInstance (p. 462)
- ModifyDBSubnetGroup (p. 466)
- RebootDBInstance (p. 468)
- RemoveTagsFromResource (p. 470)
- ResetDBClusterParameterGroup (p. 472)
- RestoreDBClusterFromSnapshot (p. 474)
- RestoreDBClusterToPointInTime (p. 479)
- StartDBCluster (p. 484)
- StopDBCluster (p. 486)
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. 535).

**ResourceName**

The Amazon DocumentDB resource that the tags are added to. This value is an Amazon Resource Name (ARN).

Type: String

Required: Yes

**Tags.Tag.N**

The tags to be assigned to the Amazon DocumentDB resource.

Type: Array of Tag (p. 531) objects

Required: Yes

Errors

For information about the errors that are common to all actions, see Common Errors (p. 533).

**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
• 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 a DB instance).

**Request Parameters**

For information about the parameters that are common to all actions, see Common Parameters (p. 535).

**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. 384).

Type: ResourcePendingMaintenanceActions (p. 529) object

**Errors**

For information about the errors that are common to all actions, see Common Errors (p. 533).

**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
- 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. 535).

**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:cluster-pg:custom-cluster-group1`.

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. 531) objects

Required: No

**Response Elements**

The following element is returned by the service.
DBClusterParameterGroup

Detailed information about a cluster parameter group.

Type: DBClusterParameterGroup (p. 497) object

Errors

For information about the errors that are common to all actions, see Common Errors (p. 533).

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
- 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.

To cancel the copy operation after it is in progress, delete the target cluster snapshot identified by TargetDBClusterSnapshotIdentifier while that DB cluster snapshot is in the copying status.

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 535).

SourceDBClusterSnapshotIdentifier

The identifier of the cluster snapshot to copy. This parameter is not case sensitive.

You can't copy an encrypted, shared cluster snapshot from one AWS Region to another.

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 Region in another 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 an encrypted cluster snapshot from another 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 encrypted DB cluster snapshot to be copied. The presigned URL request must contain the following parameter values:

- **KmsKeyId** - The AWS KMS key identifier for the key to use to encrypt the copy of the cluster snapshot in the destination AWS Region. This is the same identifier for both the CopyDBClusterSnapshot action that is called in the destination AWS Region, and the action contained in the presigned URL.
- **DestinationRegion** - The name of the AWS Region that the DB cluster snapshot will be created in.
- **SourceDBClusterSnapshotIdentifier** - The cluster snapshot identifier for 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-west-2 AWS Region, then your SourceDBClusterSnapshotIdentifier looks like the following example: arn:aws:rds:us-west-2:123456789012:cluster-snapshot:my-cluster-snapshot-20161115.

Type: String

Required: No

**Tags.Tag.N**

The tags to be assigned to the cluster snapshot.

Type: Array of Tag (p. 531) objects

Required: No
Response Elements

The following element is returned by the service.

**DBClusterSnapshot**

Detailed information about a cluster snapshot.

Type: DBClusterSnapshot (p. 499) object

Errors

For information about the errors that are common to all actions, see Common Errors (p. 533).

**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
- 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. 535).

**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

**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: Yes

**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: Yes

**AvailabilityZones.AvailabilityZone.N**

A list of Amazon EC2 Availability Zones that instances in the cluster can be created in.
CreateDBCluster

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.
Type: Array of strings
Required: No

EngineVersion
The version number of the database engine to use.
Type: String
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 `ReplicationSourceIdentifier` identifies an encrypted source, then Amazon DocumentDB uses the encryption key that is used to encrypt the source. Otherwise, Amazon DocumentDB uses your default encryption key.
- If the `StorageEncrypted` parameter is `true` and `ReplicationSourceIdentifier` is not specified, 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 Region.

If you create a replica of an encrypted cluster in another AWS Region, you must set `KmsKeyId` to a KMS key ID that is valid in the destination AWS Region. This key is used to encrypt the replica in that AWS Region.

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).


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

**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. 531) 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. 492) object

### Errors
For information about the errors that are common to all actions, see Common Errors (p. 533).

**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

**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

**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
- 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 DB cluster.

A cluster parameter group is initially created with the default parameters for the database engine used by instances in the cluster. To provide custom values for any of the parameters, you must modify the group after you create it. After you create a DB cluster parameter group, you must associate it with your cluster. For the new DB cluster parameter group and associated settings to take effect, you must then reboot the instances in the cluster without failover.

**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 cluster 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. 535).

**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. 531) objects

Required: No
Response Elements

The following element is returned by the service.

**DBClusterParameterGroup**

- Detailed information about a cluster parameter group.

  Type: [DBClusterParameterGroup](#) (p. 497) object

Errors

For information about the errors that are common to all actions, see [Common Errors](#) (p. 533).

**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
- 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. 535).

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. 531) objects

Required: No

Response Elements

The following element is returned by the service.

DBClusterSnapshot

Detailed information about a cluster snapshot.

Type: DBClusterSnapshot (p. 499) object

Errors

For information about the errors that are common to all actions, see Common Errors (p. 533).
**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
- 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. 535).

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

Indicates that minor engine upgrades are applied automatically to the instance during the maintenance window.

Default: true

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

Constraint: The AvailabilityZone parameter can't be specified if the MultiAZ parameter is set to true. The specified Availability Zone must be in the same AWS Region as the current endpoint.

Type: String

Required: No

PreferredMaintenanceWindow

The time range each week during which system maintenance can occur, in Universal Coordinated Time (UTC).


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. 531) objects

Required: No

Response Elements

The following element is returned by the service.

DBInstance

Detailed information about an instance.

Type: DBInstance (p. 506) object
Errors

For information about the errors that are common to all actions, see Common Errors (p. 533).

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
- 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. 535).

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. 531) objects
Required: No

Response Elements

The following element is returned by the service.

DBSubnetGroup

Detailed information about a subnet group.

Type: DBSubnetGroup (p. 511) object

Errors

For information about the errors that are common to all actions, see Common Errors (p. 533).
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
- 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. 535).

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.
DeleteDBCluster

DBCluster

Detailed information about a cluster.

Type: DBCluster (p. 492) object

Errors

For information about the errors that are common to all actions, see Common Errors (p. 533).

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
- 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. 535).

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. 533).

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
- 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. 535).

**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. 499) object

**Errors**

For information about the errors that are common to all actions, see Common Errors (p. 533).

**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
- 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. 535).

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. 506) object

Errors
For information about the errors that are common to all actions, see Common Errors (p. 533).

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
- 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. 535).

**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. 533).

**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
• 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. 535).

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. 518) 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. 489) 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. 533).

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
- 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. 535)`.

**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. 518)` 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. 497)` 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. 533).

DBParameterGroupNameNotFound

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
- 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. 535).

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. 518) 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. 521) objects

Errors

For information about the errors that are common to all actions, see Common Errors (p. 533).

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
- 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. 535).

**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. 518) 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. 492) 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. 533).

**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
- 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. 535).

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. 503) object

Errors

For information about the errors that are common to all actions, see Common Errors (p. 533).

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
- AWS SDK for JavaScript
- AWS SDK for PHP V3

426
- 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. 535).

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. 518) 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. 499) 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. 533).

**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
- 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. 535).

### 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:** `5.1.49`

  - **Type:** String
  - **Required:** No

### Filters.Filter.N

This parameter is not currently supported.

- **Type:** Array of Filter (p. 518) 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. 504) 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. 533).

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
DescribeDBEngineVersions

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- 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. 535).

**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. 518) 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. 506) 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. 533).

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
- 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. 535).

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. 518) 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. 511) 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. 533).

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
- 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. 535).

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. 518) 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. 514) object
Errors

For information about the errors that are common to all actions, see Common Errors (p. 533).

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
- 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. 535).

Filters.Filter.N

This parameter is not currently supported.

Type: Array of Filter (p. 518) objects

Required: No

SourceType

The type of source that is generating the events.

Valid values: db-instance, db-parameter-group, db-security-group, db-snapshot

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. 517) objects

Errors

For information about the errors that are common to all actions, see Common Errors (p. 533).

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
- 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. 535).

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. 518) 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
DescribeEvents

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. 515) 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. 533).

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
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DescribeOrderableDBInstanceOptions

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. 535).

**Engine**

The name of the engine to retrieve instance options for.

Type: String

Required: Yes

**DBInstanceClass**

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. 518) 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

OrderableDBInstanceOptions.OrderableDBInstanceOption.N
The options that are available for a particular orderable instance.
Type: Array of OrderableDBInstanceOption (p. 519) objects

Errors
For information about the errors that are common to all actions, see Common Errors (p. 533).

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
- 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. 535).

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. 518) 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. 529) objects

Errors

For information about the errors that are common to all actions, see Common Errors (p. 533).

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
- 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. 535).

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. 492) object

Errors

For information about the errors that are common to all actions, see Common Errors (p. 533).

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
- 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. 535).

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. 518) 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. 531) objects

Errors

For information about the errors that are common to all actions, see Common Errors (p. 533).

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
- 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. 535).

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. 491) 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. Changing this parameter results in an outage. The change is applied during the next maintenance window unless the ApplyImmediately parameter is set to true.

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. 492) object

### Errors

For information about the errors that are common to all actions, see Common Errors (p. 533).
**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

**DBClusterParameterGroupNameNotFound**

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
- 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. 535).

**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. 521) 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. 533).

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
- 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 DB 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 DB 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. 535).

**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
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. 503) object

Errors

For information about the errors that are common to all actions, see [Common Errors](p. 533).

**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
- 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. 535).

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

Indicates that minor version upgrades are applied automatically to the instance during the maintenance window. Changing this parameter doesn't result in an outage except in the following case, and the change is asynchronously applied as soon as possible. An outage results if this parameter is set to true during the maintenance window, and a newer minor version is available, and Amazon DocumentDB has enabled automatic patching for that engine version.

Type: Boolean
Required: No

CACertificateIdentifier

Indicates the certificate that needs to be associated with the instance.

Type: String
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

**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

**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.


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. 506) object

**Errors**

For information about the errors that are common to all actions, see Common Errors (p. 533).

**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 the depends on can't be modified.

HTTP Status Code: 400

**InsufficientDBInstanceCapacity**

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
- 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. 535).

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. 511) object

Errors

For information about the errors that are common to all actions, see Common Errors (p. 533).

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
- 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](#).

**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* object

Errors

For information about the errors that are common to all actions, see [Common Errors](#).

**DBInstanceNotFound**

- *DBInstanceIdentifier* doesn't refer to an existing instance.
- HTTP Status Code: 404

**InvalidDBObjectInstanceState**

- 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
- 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. 535).

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. 533).

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
- 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. 535).

**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. 521) 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. 533).

**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
- 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. 535).

**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. 531) 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. 492) object

**Errors**
For information about the errors that are common to all actions, see Common Errors (p. 533).

**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
- 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. 535).

**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

**Tags.Tag.N**

The tags to be assigned to the restored cluster.
Type: Array of Tag (p. 531) 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. 492) object

### Errors

For information about the errors that are common to all actions, see Common Errors (p. 533).

**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
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
StartDBCluster

Starts 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. 535).

DBClusterIdentifier


Type: String
Required: Yes

Response Elements

The following element is returned by the service.

DBCluster

Detailed information about a cluster.

Type: DBCluster (p. 492) object

Errors

For information about the errors that are common to all actions, see Common Errors (p. 533).

DBClusterNotFoundFault

DBClusterIdentifier doesn't refer to an existing cluster.

HTTP Status Code: 404

InvalidDBClusterStateException

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
• 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. 535).

DBClusterIdentifier


Type: String

Required: Yes

Response Elements

The following element is returned by the service.

DBCluster

Detailed information about a cluster.

Type: DBCluster (p. 492) object

Errors

For information about the errors that are common to all actions, see Common Errors (p. 533).

DBClusterNotFoundFault

DBClusterIdentifier doesn't refer to an existing cluster.

HTTP Status Code: 404

InvalidDBClusterStateException

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
Data Types

The following data types are supported:

- AvailabilityZone (p. 488)
- Certificate (p. 489)
- CloudwatchLogsExportConfiguration (p. 491)
- DBCluster (p. 492)
- DBClusterMember (p. 496)
- DBClusterParameterGroup (p. 497)
- DBClusterRole (p. 498)
- DBClusterSnapshot (p. 499)
- DBClusterSnapshotAttribute (p. 502)
- DBClusterSnapshotAttributesResult (p. 503)
- DBEngineVersion (p. 504)
- DBInstance (p. 506)
- DBInstanceStatusInfo (p. 510)
- DBSubnetGroup (p. 511)
- Endpoint (p. 513)
- EngineDefaults (p. 514)
- Event (p. 515)
- EventCategoriesMap (p. 517)
- Filter (p. 518)
- OrderableDBInstanceOption (p. 519)
- Parameter (p. 521)
- PendingCloudwatchLogsExports (p. 523)
- PendingMaintenanceAction (p. 524)
- PendingModifiedValues (p. 526)
- ResourcePendingMaintenanceActions (p. 529)
- Subnet (p. 530)
- Tag (p. 531)
- UpgradeTarget (p. 532)
- VpcSecurityGroupMembership (p. 533)
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
- 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
- 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
- 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. 498) 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

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. 496) 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`.

494
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

**Status**

Specifies the current state of this cluster.

Type: String
Required: No

**StorageEncrypted**

Specifies whether the cluster is encrypted.

Type: Boolean
Required: No


Provides a list of virtual private cloud (VPC) security groups that the cluster belongs to.

Type: Array of VpcSecurityGroupMembership (p. 533) 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
- 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
- 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
- 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 IAM role that is associated with the DB cluster.

Type: String
Required: No

Status

Describes the state of association between the IAM role and the cluster. The Status property returns one of the following values:

- ACTIVE - The IAM role ARN is associated with the cluster and can be used to access other AWS services on your behalf.
- PENDING - The IAM role ARN is being associated with the DB cluster.
- INVALID - The IAM role ARN is associated with the cluster, but the cluster cannot assume the IAM role 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
- 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
- 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

AttributeValue.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
- 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**

The list of attributes and values for the cluster snapshot.

Type: Array of **DBClusterSnapshotAttribute** (p. 502) 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
- 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.N

A list of engine versions that this database engine version can be upgraded to.

Type: Array of UpgradeTarget (p. 532) 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
- AWS SDK for Ruby V3
DBInstance

Detailed information about an instance.

Contents

Note
In the following list, the required parameters are described first.

AutoMinorVersionUpgrade
Indicates that minor version patches are applied automatically.
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

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
**DBInstanceId**

Contains a user-provided database identifier. This identifier is the unique key that identifies an instance.

- Type: String
- Required: No

**DBInstanceStatus**

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. 511) object
- Required: No

**EnabledCloudwatchLogsExports.member.N**

A list of log types that this instance is configured to export to Amazon CloudWatch Logs.

- Type: Array of strings
- Required: No

**Endpoint**

Specifies the connection endpoint.

- Type: Endpoint (p. 513) 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. 526) 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. 510) objects

Required: No

**StorageEncrypted**

Specifies whether or not the instance is encrypted.

Type: Boolean

Required: No


Provides a list of VPC security group elements that the instance belongs to.

Type: Array of `VpcSecurityGroupMembership` (p. 533) 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
- AWS SDK for Ruby V3
DBInstanceStatusInfo

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
- 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. 530) 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
- 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
- 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. 521) 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
- 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
• 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
- 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
- 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. 488) 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
• 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
- 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
- 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
• 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. 523)` 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
• AWS SDK for Ruby V3
ResourcePendingMaintenanceActions

Represents the output of `ApplyPendingMaintenanceAction` (p. 384).

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. 524) 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
- 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. 488) 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
- 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: "^([\p{L}\p{Z}\p{N}_.:/=+\-]*)$").

Type: String
Required: Yes

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: "^([\p{L}\p{Z}\p{N}_.:/=+\-]*)$").

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
- 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
- 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
- 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

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

ValidationError
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
The following table describes the documentation for this release of the *Amazon DocumentDB Developer Guide*.

<table>
<thead>
<tr>
<th>update-history-change</th>
<th>update-history-description</th>
<th>update-history-date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Added support for role-based access control.</td>
<td>Amazon DocumentDB added support for role-based access control using built-in roles.</td>
<td>March 26, 2020</td>
</tr>
<tr>
<td>Added support for an additional Availability Zone in Canada (Central) (ca-central-1).</td>
<td>Amazon DocumentDB is now available in the Canada (Central) Region (ca-central-1) with R5 class instances and 3 Availability Zones.</td>
<td>March 26, 2020</td>
</tr>
<tr>
<td>Added support for two additional MongoDB APIs.</td>
<td>Amazon DocumentDB added support for <code>$dateFromString</code> and <code>executionStats</code> MongoDB APIs.</td>
<td>March 23, 2020</td>
</tr>
<tr>
<td>Added support for five additional MongoDB APIs.</td>
<td>Amazon DocumentDB added support for <code>$objectToArray</code>, <code>$arrayToObject</code>, <code>$slice</code>, <code>$mod</code>, and <code>$range</code> MongoDB APIs.</td>
<td>February 6, 2020</td>
</tr>
<tr>
<td>Added support for Canada (Central).</td>
<td>Amazon DocumentDB is now available in the Canada (Central) Region (ca-central-1) with R5 class instances.</td>
<td>December 11, 2019</td>
</tr>
<tr>
<td>Added support for ChangeStreamLogSize.</td>
<td>Amazon DocumentDB added support for <code>ChangeStreamLogSize</code> for Cloudwatch metrics.</td>
<td>November 22, 2019</td>
</tr>
<tr>
<td>Added support for Europe (Paris) region</td>
<td>Amazon DocumentDB is now available in the Europe (Paris) region (eu-west-3) with R5 class instances.</td>
<td>October 30, 2019</td>
</tr>
<tr>
<td>Added support for Asia Pacific (Mumbai) region</td>
<td>Amazon DocumentDB is now available in the Asia Pacific (Mumbai) region (ap-south-1) with R5 class instances.</td>
<td>October 17, 2019</td>
</tr>
<tr>
<td>Added support for three additional MongoDB APIs</td>
<td>Amazon DocumentDB added support for the $addFields, $concatArrays, and $lookup MongoDB APIs.</td>
<td>October 16, 2019</td>
</tr>
<tr>
<td>Added support for Asia Pacific (Singapore) region</td>
<td>Amazon DocumentDB is now available in the Asia Pacific (Singapore) region (ap-southeast-1) with R5 class instances.</td>
<td>October 14, 2019</td>
</tr>
<tr>
<td>Added new document for updating TLS certificates</td>
<td>Added instructions for updating CA certificates to use the new CA certificate to create TLS connections.</td>
<td>October 2, 2019</td>
</tr>
<tr>
<td>Added API support for certificates</td>
<td>Amazon DocumentDB a new Certificate data type for instances. For more information, see <code>DBInstance</code>.</td>
<td>October 1, 2019</td>
</tr>
<tr>
<td>Support for query profiling</td>
<td>Amazon DocumentDB added the ability to profile supported operations on your cluster's instances and databases.</td>
<td>August 19, 2019</td>
</tr>
<tr>
<td>Added third AZ in Asia Pacific (Tokyo)</td>
<td>Amazon DocumentDB added a third Availability Zone (AZ) for your compute instances in Asia Pacific (Tokyo).</td>
<td>August 9, 2019</td>
</tr>
<tr>
<td>Support for additional MongoDB APIs</td>
<td>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 <code>top()</code> command for collection level diagnostics and the ability to modify the <code>expireAfterSeconds</code> parameter for TTL indexes using the <code>collMod()</code> command.</td>
<td>July 31, 2019</td>
</tr>
<tr>
<td>Added support for Europe (London)</td>
<td>Amazon DocumentDB is now available in Europe (London) (eu-west-2) with R5 class instances.</td>
<td>July 18, 2019</td>
</tr>
<tr>
<td>Added code samples</td>
<td>Added code examples in R and Ruby for programmatically connecting to Amazon DocumentDB.</td>
<td>July 17, 2019</td>
</tr>
<tr>
<td>Added best practice</td>
<td>Added a Best Practice to help you manage your Amazon DocumentDB costs.</td>
<td>July 17, 2019</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Date</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>Support for stopping and starting a cluster</td>
<td>Amazon DocumentDB added support for stopping and starting clusters to manage costs for development and test environments.</td>
<td>July 1, 2019</td>
</tr>
<tr>
<td>Support for cluster deletion protection</td>
<td>To protect your clusters from accidental deletion, Amazon DocumentDB added deletion protection. For more information, see the following topics: <a href="https://docs.aws.amazon.com/documentdb/latest/developerguide/documentdb-createcluster.html">Creating an Amazon DocumentDB Cluster</a>, <a href="https://docs.aws.amazon.com/documentdb/latest/developerguide/documentdb-modifycluster.html">Modifying an Amazon DocumentDB Cluster</a>, <a href="https://docs.aws.amazon.com/documentdb/latest/developerguide/documentdb-deletecluster.html">Deleting an Amazon DocumentDB Cluster</a>, and <a href="https://docs.aws.amazon.com/documentdb/latest/developerguide/documentdb-api-dbcluster.html">DeletionProtection in the API topic DBCluster</a>.</td>
<td>July 1, 2019</td>
</tr>
<tr>
<td>Functional differences update</td>
<td>Added Implicit Transactions to Functional Differences.</td>
<td>June 26, 2019</td>
</tr>
<tr>
<td>Functional differences addition</td>
<td>Added note regarding storage and index compression in Amazon DocumentDB.</td>
<td>June 13, 2019</td>
</tr>
<tr>
<td>Additional region supported</td>
<td>Amazon DocumentDB is now available in Asia Pacific (Sydney) (ap-southeast-2) with R5 class instances.</td>
<td>June 5, 2019</td>
</tr>
<tr>
<td>R5 instance class supported in additional regions</td>
<td>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.</td>
<td>May 17, 2019</td>
</tr>
<tr>
<td>Additional regions supported</td>
<td>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 <a href="https://docs.aws.amazon.com/documentdb/latest/developerguide/supported-instance-classes.html">Supported Instance Classes by Region</a> and <a href="https://docs.aws.amazon.com/documentdb/latest/developerguide/instance-class-specifications.html">Instance Class Specifications</a>.</td>
<td>May 8, 2019</td>
</tr>
<tr>
<td>Added more connection code examples</td>
<td>Added code examples in Java and C# for connecting to Amazon DocumentDB.</td>
<td>April 24, 2019</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Date</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>Additional Mongo API support</td>
<td>Added support for seven aggregation string operators ($indexOfBytes, $indexOfCP, $strLenBytes, $strLenCP, $toLower, $toUpper, and $split), nine date-time operators ($dayOfYear, $dayOfMonth, $dayOfWeek, $year, $month, $hour, $minute, $second, and $millisecond), and the $sample aggregation pipeline stage.</td>
<td>April 4, 2019</td>
</tr>
<tr>
<td>Added connection code examples</td>
<td>Added code examples in Python, Node.js, PHP, and Go for connecting to Amazon DocumentDB.</td>
<td>March 21, 2019</td>
</tr>
<tr>
<td>Support for Frankfurt Region and R5 instances</td>
<td>Added support for Europe (Frankfurt) Region (eu-central-1) with R5 instance classes. For more information, see <a href="https://docs.aws.amazon.com/documentdb/latest/developerguide/region.html">Supported Instance Classes by Region</a> and <a href="https://docs.aws.amazon.com/documentdb/latest/developerguide/capacity.html">Instance Class Specifications</a>.</td>
<td>March 13, 2019</td>
</tr>
<tr>
<td>Aggregation pipeline operators support</td>
<td>Added support for new aggregation string operators ($concat, $substr, $substrBytes, $substrCP, $strcasecmp), an array aggregation operator ($size), an aggregation group accumulator operator ($push), and aggregation stages ($redact and $indexStats). We also added support for positional array operators ($[] and $[[&lt;identifier&gt;]]) and hint().</td>
<td>February 28, 2019</td>
</tr>
<tr>
<td>Engine upgrades</td>
<td>Added documentation for determining pending cluster modifications and upgrading your cluster's engine version.</td>
<td>February 15, 2019</td>
</tr>
<tr>
<td>Auditing events</td>
<td>Added support for auditing database events with Amazon CloudWatch Logs.</td>
<td>February 12, 2019</td>
</tr>
<tr>
<td>Quick Start</td>
<td>Added a Quick Start topic to help you easily start with Amazon DocumentDB using AWS CloudFormation.</td>
<td>January 11, 2019</td>
</tr>
<tr>
<td><strong>Public Release (p. 537)</strong></td>
<td>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.</td>
<td>January 9, 2019</td>
</tr>
</tbody>
</table>