Convert a Replica Set to a Sharded Cluster

Sharded clusters partition data across multiple servers based on a shard key. A sharded cluster scales better than a replica set for deployments with large data sets and high throughput operations.

This tutorial converts a single three-member replica set to a sharded cluster with two shards. Each shard in the new cluster is an independent three-member replica set.

You can convert to a sharded cluster in the UI for deployments hosted in MongoDB Atlas.

About This Task

- Some steps in this tutorial cause downtime for your deployment. Individual steps note when downtime will occur.
- This tutorial is for deployments that have authentication enabled.
- In this tutorial, you specify server settings with configuration files. Configuration files contain settings that are equivalent to the `mongod` and `mongos` command-line options.
- The sharded cluster you deploy with this tutorial contains ten servers:
  - One server for the `mongos`.
  - Three servers each for the two shards in the cluster (six servers in total).
  - Three servers for the config server replica set.

Server Architecture

This tutorial uses the following servers:

Hostname	Port	Description
`mongodb0.example.net`	`27017`	Member of the initial data-bearing shard, `rs0`.
`mongodb1.example.net`	`27017`	Member of the initial data-bearing shard, `rs0`.
`mongodb2.example.net`	`27017`	Member of the initial data-bearing shard, `rs0`.
`mongodb3.example.net`	`27018`	Member of the second data-bearing shard, `rs1`.
`mongodb4.example.net`	`27018`	Member of the second data-bearing shard, `rs1`.
`mongodb5.example.net`	`27018`	Member of the second data-bearing shard, `rs1`.
`mongodb6.example.net`	`27017`	The `mongos`, used to connect to the sharded cluster.
`mongodb7.example.net`	`27019`	Member of the config server replica set.

Hostname	Port	Description
`mongodb8.example.net`	`27019`	Member of the config server replica set.
`mongodb9.example.net`	`27019`	Member of the config server replica set.

The hostnames used in this tutorial are examples. Replace the hostnames used in the example commands with the hostnames used in your deployments.

IMPORTANT

To avoid configuration updates due to IP address changes, use DNS hostnames instead of IP addresses. It is particularly important to use a DNS hostname instead of an IP address when configuring replica set members or sharded cluster members.

Use hostnames instead of IP addresses to configure clusters across a split network horizon. Starting in MongoDB 5.0, nodes that are only configured with an IP address will fail startup validation and will not start.

Before You Begin

- To complete this tutorial, you must have a replica set that uses either keyfile or x.509 certificate authentication. To deploy a secure replica set that uses one of these authentication methods, see either:
  - Deploy Replica Set With Keyfile Authentication
  - Use x.509 Certificate for Membership Authentication
- This tutorial uses the default data directories `/data/db` and `/data/configdb`. To use different paths, set the `storage.dbPath` setting in your configuration file.

Steps

To convert a replica set to a sharded cluster:

1. Deploy the config server replica set.
2. Deploy a `mongos`.
3. Create an administrative user for the sharded cluster.
4. Migrate the `admin` database from the initial replica set to the sharded cluster.
5. Restart the members of the initial replica set as shard servers.
6. Add the initial replica set as a shard.

7. Update your application connection string.
8. Create a second shard and add it to the cluster.
9. Shard a collection.

1. Deploy the Config Server Replica Set

Deploy a three-member replica set for the config servers. In this example, the config servers use the following hosts:

- `mongodb7.example.net`
- `mongodb8.example.net`
- `mongodb9.example.net`

1 **Deploy the config server replica set members.**

Select the tab for your authentication mechanism:

Keyfile Authentication | x.509 Authentication

Start a `mongod` instance on each of these hosts:

- `mongodb7.example.net`
- `mongodb8.example.net`
- `mongodb9.example.net`

Specify these options in the configuration file for each `mongod` instance:

Option	Value
`security.keyFile`	The path to the key file used for your initial replica set.
`replication.replSetName`	`configReplSet`
`sharding.clusterRole`	`configsvr`
`net.bindIp`	`localhost`, followed by any other hostnames on which the `mongod` should listen for client connections.

```
security:
keyFile: <PATH_TO_KEYFILE>
replication:
replSetName: configReplSet
sharding:
clusterRole: configsvr
net:
```

```
bindIp: localhost,<hostname(s)>
```

Include additional options as appropriate for your deployment.

Deploy the `mongod` with your specified configuration:

```
mongod --config
<PATH_TO_CONFIG_FILE>
```

The config servers use the default data directory `/data/configdb` and the default port `27019`.

2 **Connect to one of the config servers.**

Keyfile Authentication x.509 Authentication

Use `mongosh` to connect to one of the config servers. For example:

```
mongosh
"mongodb://mongodb7.example.net:27019"
```

3 **Initiate the config server replica set.**

To initiate the replica set, run `rs.initiate()`:

```
rs.initiate( {
_id: "configReplSet",
configsvr: true,
members: [
{ _id: 0, host: "mongodb7.example.net:27019" },
{ _id: 1, host: "mongodb8.example.net:27019" },
{ _id: 2, host: "mongodb9.example.net:27019" }
]
} )
```

The preceding command uses the localhost exception to perform administrative actions without authentication.

IMPORTANT

Run `rs.initiate()` on *just one and only one* `mongod` instance for the replica set.

2. Deploy the `mongos`

The `mongos` provides the interface between the client applications and the sharded cluster.

1. **Create a configuration file for the mongos.**

   Keyfile Authentication x.509 Authentication

   Specify these options in your `mongos` configuration file:

Option	Value
`sharding.configDB`	`configReplSet`, followed by a slash `/` and at least one of the config server hostnames and ports.
`security.keyFile`	The path to the key file used for your initial replica set.
`net.bindIp`	`localhost`, followed by any other hostnames on which the `mongos` should listen for client connections.

   ```
   sharding:
   configDB:
   configReplSet/mongodb7.example.net:27019,mongodb8.example.net:27019,m
   security:
   keyFile: <PATH_TO_KEYFILE>
   net:
   bindIp: localhost,<hostname(s)>
   ```

   Include additional options as appropriate for your deployment.

2. **Deploy the mongos.**

   Deploy the `mongos` with your specified configuration:

   ```
   mongos --config
   <PATH_TO_CONFIG_FILE>
   ```

3. Create an Administrative User for the Sharded Cluster

After you deploy the `mongos`, use the localhost exception to create the first user on the cluster.

1. **Connect to the `mongos`.**

   Use `mongosh` to connect to the `mongos`.

Keyfile Authentication x.509 Authentication

```
mongosh
"mongodb://mongodb6.example.net:27017"
```

2 **Create an administrative user.**

TIP

Specify a different username for your sharded cluster administrative user than the usernames in your initial replica set.

Using different names avoids conflicts when restoring the users from the initial replica set to the sharded cluster.

The following `db.createUser()` method creates a user named `admin01` with the `clusterManager` and `restore` roles:

```
use admin
db.createUser(
{
user: "admin01",
pwd: passwordPrompt(),
roles: [
{ role: "clusterManager", db: "admin" },
{ role: "restore", db: "admin" }
]
}
)
```

After you run the command, the database prompts you to enter a password for the `admin01` user.

4. Migrate the `admin` Database from the Initial Replica Set

The `admin` database contains user and system information. To migrate the `admin` database from your initial replica set to the sharded cluster, perform the following steps.

Keyfile Authentication x.509 Authentication

1 **Create a backup of the replica set's admin database.**

Run the following `mongodump` command **from the command line**. Use the authentication credentials for your initial replica set:

```
mongodump --
uri="mongodb://mongodb0.example.net:27017,mongodb1.example.net:27017,r
replicaSet=rs0" \
--db=admin --username=<USERNAME> --out=<PATH_TO_DUMP_DIRECTORY>
```

After you enter the `mongodump` command, the server prompts you to enter your database user's password.

2. **Restore the `admin` database to the sharded cluster.**

   Run the following `mongorestore` command **from the command line**:

```
mongorestore --uri="mongodb://mongodb6.example.net:27017" --
nsInclude="admin.*" \
--username=<USERNAME> <PATH_TO_DUMP_DIRECTORY>
```

   After you enter the `mongorestore` command, the server prompts you to enter your database user's password.

5. Restart the Initial Replica Set as a Shard

In this example, your initial replica set is a three-member replica set. This step updates the initial replica set so that it can be added as a shard to your sharded cluster.

The replica set runs on these hosts:

- `mongodb0.example.net:27017`
- `mongodb1.example.net:27017`
- `mongodb2.example.net:27017`

For sharded clusters, you must set the role for each `mongod` instance in the shard to `shardsvr`. To specify the server role, set the `sharding.clusterRole` setting in the `mongod` configuration file.

> **NOTE**
>
> The default port for `mongod` instances with the `shardsvr` role is `27018`. To use a different port, specify the `net.port` setting.

1. **Connect to a member of your initial replica set.**

   Use `mongosh` to connect to one of the members of your **initial replica set**.

Keyfile Authentication x.509 Authentication

```
mongosh
"mongodb://<username>@mongodb0.example.net:27017"
```

2 **Determine the replica set's primary and secondaries.**

Run `rs.status()` to determine the primary and secondaries:

```
rs.status()
```

In the command output, the `replSetGetStatus.members[n].stateStr` field indicates which member is the primary and which members are secondaries.

3 **Restart the secondaries with the `--shardsvr` option.**

WARNING

This step requires some downtime for applications connected to the replica set secondaries.

After you restart a secondary, any applications that are connected to that secondary return a `CannotVerifyAndSignLogicalTime` error until you perform the steps in 6. Add the Initial Replica Set as a Shard.

You can also restart your application to stop it from receiving `CannotVerifyAndSignLogicalTime` errors.

Keyfile Authentication x.509 Authentication

1

Connect to a secondary.

Use `mongosh` to connect to one of the secondaries.

```
mongosh "mongodb://<username>@<host>:
<port>"
```

2

Shut down the secondary.

Run the following commands:

```
use admin
db.shutdownServer()
```

3

Edit the secondary's configuration file.

In the secondary's configuration file, set `sharding.clusterRole` to `shardsvr`:

```
security:
keyFile: <PATH_TO_KEYFILE>
replication:
replSetName: rs0
sharding:
clusterRole: shardsvr
net:
port: 27017
bindIp: localhost,<hostname(s)>
```

Include additional options as appropriate for your deployment.

4

Restart the secondary as a shard server.

Run the following command on the host containing the secondary:

```
mongod --config
<PATH_TO_CONFIG_FILE>
```

5

Repeat the shut down and restart steps for the other secondary.

4 **Restart the primary with the `--shardsvr` option.**

WARNING

This step requires some downtime for applications connected to the primary of the replica set.

After you restart the primary, any applications that are connected to the primary return a `CannotVerifyAndSignLogicalTime` error until you perform the steps in 6. Add the Initial Replica Set as a Shard.

You can also restart your application to stop it from receiving `CannotVerifyAndSignLogicalTime` errors.

Keyfile Authentication x.509 Authentication

1

Connect to the primary.

Use `mongosh` to connect to the primary:

```
mongosh "mongodb://<username>@<host>:
<port>"
```

2

Step down the primary.

Run the following command:

```
rs.stepDown()
```

3

Verify that the step down is complete.

Run `rs.status()` to confirm that the member you are connected to has stepped down and is now a secondary:

```
rs.status()
```

4

Shut down the former primary.

Run the following commands:

```
use admin
db.shutdownServer()
```

Wait for the shutdown to complete.

5

Edit the primary's configuration file.

In the primary's configuration file, set `sharding.clusterRole` to `shardsvr`:

```
security:
keyFile: <PATH_TO_KEYFILE>
replication:
replSetName: rs0
sharding:
clusterRole: shardsvr
net:
port: 27017
bindIp: localhost,<hostname(s)>
```

Include additional options as appropriate for your deployment.

6

Restart the primary as a shard server.

Run the following command on the host containing the primary:

```
mongod --config
<PATH_TO_CONFIG_FILE>
```

6. Add the Initial Replica Set as a Shard

After you convert the initial replica set (`rs0`) to a shard, add it to the sharded cluster.

1. **Connect to the `mongos` as your cluster's administrative user.**

   The `mongos` instance is running on host `mongodb6.example.net`.

   Keyfile Authentication x.509 Authentication

   To connect `mongosh` to the `mongos`, run the following command:

   ```
   mongosh
   "mongodb://admin01@mongodb6.example.net:27017"
   ```

   This command authenticates you as the `admin01` user you created on the sharded cluster. After you enter the command, enter your user's password.

2. **Add the shard.**

   To add a shard to the cluster, run the `sh.addShard()` method:

   ```
   sh.addShard(
   "rs0/mongodb0.example.net:27017,mongodb1.example.net:27017,mongodb2.ex
   ```

WARNING

Once the new shard is active, `mongosh` and other clients must **always** connect to the `mongos` instance. Do not connect directly to the `mongod` instances. If your clients connect to shards directly, you may create data or metadata inconsistencies.

7. Update Your Application Connection String

After you add the first shard to your cluster, update the connection string used by your applications to the connection string for your sharded cluster. Then, restart your applications.

8. Deploy a Second Replica Set

Deploy a new replica set called `rs1`. The members of replica set `rs1` are on the following hosts:

- `mongodb3.example.net`
- `mongodb4.example.net`

- `mongodb5.example.net`

1. **Start each member of the replica set.**

Keyfile Authentication x.509 Authentication

For each `mongod` instance in the replica set, create a configuration file with these options:

Option	Value
`security.keyFile`	The path to the key file used for your initial replica set.
`replication.replSetName`	`rs1`
`sharding.clusterRole`	`shardsvr`
`net.bindIp`	`localhost`, followed by any other hostnames on which the `mongod` should listen for client connections.

```
security:
keyFile: <PATH_TO_KEYFILE>
replication:
replSetName: rs1
sharding:
clusterRole: shardsvr
net:
bindIp: localhost,<hostname(s)>
```

Include additional options as appropriate for your deployment.

Deploy the `mongod` with your specified configuration:

```
mongod --config
<PATH_TO_CONFIG_FILE>
```

NOTE

When you specify the `--shardsvr` option for a `mongod` instance, the instance runs on port `27018` by default.

2. **Repeat the previous step for the other two members of the `rs1` replica set.**

3 Connect to a replica set member.

Use `mongosh` to connect to one of the replica set members. For example:

```
mongosh
"mongodb://mongodb3.example.net:27018"
```

4 Initiate the replica set.

In `mongosh`, run the `rs.initiate()` method to initiate a replica set that contains the current member:

```
rs.initiate( {
_id : "rs1",
members: [
{ _id: 0, host: "mongodb3.example.net:27018" },
{ _id: 1, host: "mongodb4.example.net:27018" },
{ _id: 2, host: "mongodb5.example.net:27018" }
]
} )
```

The preceding command requires the localhost exception to perform administrative actions without authentication.

IMPORTANT

Run `rs.initiate()` on *just one and only one* `mongod` instance for the replica set.

5 Add an administrative user for the replica set.

After you deploy the replica set, use the localhost exception to create the replica set's first user.

1 Determine the replica set primary.

To determine the primary, run `rs.status()`:

```
rs.status()
```

In the command output, the `replSetGetStatus.members[n].stateStr` field indicates which member is the primary.

2

Connect to the replica set primary.

Connect to the replica set primary with `mongosh`. For example, if the primary is `mongodb4.example.net`, run this command:

```
mongosh
"mongodb://mongodb4.example.net:27018"
```

3

Create an administrative user.

Run the following `db.createUser()` method to create a user named `rs1Admin` with the `userAdmin` role:

```
use admin
db.createUser(
{
user: "rs1Admin",
pwd: passwordPrompt(),
roles: [
{ role: "userAdmin", db: "admin" }
]
}
)
```

After you run the command, the database prompts you to enter a password for the `rs1Admin` user.

9. Add the Second Replica Set to the Cluster as a Shard

Add the new replica set, `rs1`, to the sharded cluster.

1 **Connect `mongosh` to the `mongos`.**

Run the following command **from the command line** to connect to the `mongos` instance running on host `mongodb6.example.net`:

Keyfile Authentication x.509 Authentication

```
mongosh
"mongodb://admin01@mongodb6.example.net:27017/admin"
```

This command authenticates you as the `admin01` user you created on the sharded cluster. After you enter the command, enter your user's password.

2. **Add the second shard.**

   After you connect to the `mongos`, add the replica set `rs1` as a shard to the cluster with the `sh.addShard()` method:

```
sh.addShard(
"rs1/mongodb3.example.net:27018,mongodb4.example.net:27018,mongodb5.ex
```

10. Shard a Collection

The final step of the procedure is to shard a collection in the sharded cluster.

1. **Determine the shard key.**

   Determine the shard key for the collection. The shard key indicates how MongoDB distributes the documents between shards. Good shard keys:

   - Have values that are evenly distributed among all documents.
   - Group documents that are often accessed at the same time into contiguous chunks.
   - Allow for effective distribution of activity among shards.

   For more information, see Choose a Shard Key.

   This procedure uses the `number` field as the shard key for the `test_collection` collection.

2. **Create an index on the shard key.**

   Before you shard a non-empty collection, create an index on the shard key:

```
use test
db.test_collection.createIndex( { "number" : 1 } )
```

3. **Shard the collection.**

   In the `test` database, shard the `test_collection`. Specify `number` as the shard key.

```
sh.shardCollection( "test.test_collection", { "number" :
1 } )
```

The next time that the balancer runs, it redistributes chunks of documents between shards. As clients insert additional documents into this collection, the `mongos` routes the documents to the

appropriate shard.

TIP

Schedule the balancing window

When the balancer redistributes chunks, it may negatively impact your application's performance. To minimize performance impact, you can specify when the balancer runs so it does not run during peak hours. To learn more, see Schedule the Balancing Window.