Migrating from MongoDB to MongoDB Atlas using Atlas Live Migration

This tutorial describes and implements a migration from a self-managed MongoDB replica set (https://docs.mongodb.com/manual/replication/) that contains databases to a fully managed cluster in MongoDB Atlas (https://www.mongodb.com/cloud/atlas) by using MongoDB's Atlas Live Migration Service (Atlas Live Migration) (https://docs.atlas.mongodb.com/import/live-import/).

This tutorial is intended for database architects, database administrators, and database engineers who are interested in a fully hosted MongoDB service, or are responsible for migrating MongoDB databases in a MongoDB replica set to a MongoDB Atlas cluster.

Objectives

- Set up your self-managed source by creating and loading documents into a sample MongoDB replica set.
- Set up a migration target cluster in MongoDB Atlas.
- Use Atlas Live Migration to migrate a database from your self-managed MongoDB replica set to a fully managed MongoDB Atlas cluster.
- Understand and select testing, cutover, and fallback strategies.

This tutorial uses a MongoDB replica set as the source. Migrating a sharded MongoDB cluster (https://docs.atlas.mongodb.com/import/live-import-sharded/) to MongoDB Atlas is not covered in this tutorial. The architectural difference between a MongoDB replica set and a sharded MongoDB cluster is explained in this Stack Exchange: Database Administrators (https://dba.stackexchange.com/questions/52632/difference-between-sharding-and-replication-on-mongodb) post.

To complete the migration for this tutorial, you use Atlas Live Migration (https://docs.atlas.mongodb.com/import/live-import/), not the Atlas `mongomirror` (https://docs.atlas.mongodb.com/reference/mongomirror/) utility. The `mongomirror` utility requires the installation of an agent on the source MongoDB environment and operates at a lower level of abstraction.

The migration setup in this tutorial is a homogeneous migration (/solutions/database-migration-concepts-principles-part-1#heterogeneous_migration_versus_homogeneous_migration) with a copy semantics. Data is not transformed during the migration, and no database consolidation or data resharding occurs. You can implement functionality beyond a copy semantics with integration technology such as Striim (https://www.striim.com/).

The migration in this tutorial is a one-way migration from a source MongoDB replica set to a target MongoDB Atlas cluster. After the cutover from the source MongoDB replica set to the target cluster is complete, the source databases are not kept up to date with changes to the target cluster. Thus, if you implement this solution in a production environment, you cannot switch your applications to up-to-date source databases in a fallback. For more information on fallback processes, see Database migration: Concepts and principles (Part 2) (/solutions/database-migration-concepts-principles-part-2).

Costs

This tutorial uses the following billable components of Google Cloud:

- Compute Engine (/compute/all-pricing)

To complete this tutorial, you cannot use the MongoDB Atlas free tier. The available machine types in the free tier do not support Atlas Live Migration. The minimum required machine type (M10 at the time of this writing) has an hourly service cost in MongoDB Atlas. To generate a price estimate, go to MongoDB Atlas Pricing (https://www.mongodb.com/cloud/atlas/pricing), click **Google Cloud Platform**, and then follow the instructions. If you implement this migration in production, we recommend that you use the regular hosted version of MongoDB Atlas (https://console.cloud.google.com/marketplace/details/mongodb/atlas-pro).

When you finish this tutorial, you can avoid continued billing by deleting the resources you created. For more information, see Cleaning up (#clean-up).

Before you begin

1. In the Cloud Console, on the project selector page, select or create a Cloud project.

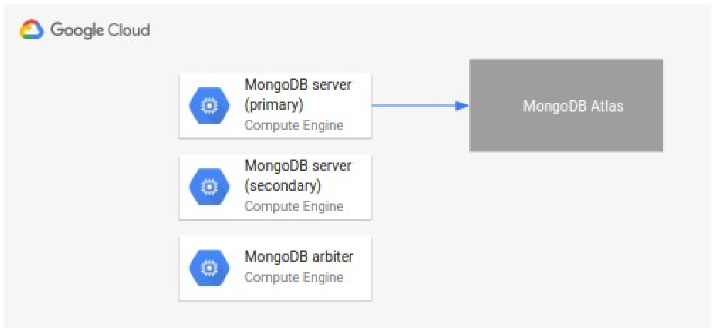
★ **Note**: If you don't plan to keep the resources that you create in this procedure, create a project instead of selecting an existing project. After you finish these steps, you can delete the project, removing all resources associated with the project.

Go to the project selector page (https://console.cloud.google.com/projectselector2/home/dashboard)

2. Make sure that billing is enabled for your Google Cloud project. Learn how to confirm billing is enabled for your project (/billing/docs/how-to/modify-project).

Migration architecture

The following diagram shows the deployment architecture that you create in this tutorial.


Google Cloud
MongoDB server (primary) Compute Engine
MongoDB server (secondary) Compute Engine
MongoDB arbiter Compute Engine
MongoDB Atlas

The arrow represents the data migration path from the source MongoDB replica set running on Compute Engine to the target cluster running in MongoDB Atlas on Google Cloud.

The deployment architecture contains the following components:

- **Source database:** a self-managed MongoDB replica set running on three Compute Engine instances
- **Target database:** a fully managed MongoDB Atlas cluster
- **Migration service:** an Atlas Live Migration configuration to migrate data from the source to the target

Although this tutorial uses a self-managed MongoDB replica set on Compute Engine instances, you can also deploy a source MongoDB replica set in an on-premises data center or another cloud environment.

Atlas Live Migration supports a zero-downtime database migration approach (/solutions/database-migration-concepts-principles-part-1#migration_downtime_zero_versus_minimal_versus_significant). During the migration from the source MongoDB replica set, your applications can still access the source databases without impact. After the initial load (/database-migration-concepts-principles-part-2#phase_1_initial_load), Atlas Live Migration migrates changes as they occur after the migration starts.

To perform the cutover from the source database to the target cluster after the initial dataset is migrated, you take the following steps:

1. Suspend write access to the source database.

2. Wait for Atlas Live Migration to capture the remaining changes and apply them to the target database.
3. Execute the cutover in Atlas Live Migration.
4. Stop the source database.

After all data is migrated, Atlas Live Migration notifies you on the progress status line in the user interface. At this point, the data migration is complete and the application systems can begin accessing the target cluster as the new system of record.

Creating a self-managed MongoDB replica set

First, you install the MongoDB replica set on Google Cloud. This database serves as your source database. Next, you check whether your source database meets all required preconditions.

Note: The reason that this tutorial includes steps for checking for preconditions is to prepare you for this step in a production environment. Even if a MongoDB replica already exists in your production environment, you still need to check for preconditions.

After you complete the preconditions check, you must enable authentication and restart the source MongoDB instance. Finally, to test the migration, you add a sample data set to the source MongoDB instance that is migrated to the target database.

Install the MongoDB replica set

1. In the Google Cloud Marketplace, go to the MongoDB replica set installation on Compute Engine. At the time of this writing, the current version is MongoDB 4.0.

   Go to MongoDB in Cloud Marketplace (https://console.cloud.google.com/marketplace/details/click-to-deploy-images/mongodb)

2. Click **Launch**. Because several Google Cloud APIs are enabled, the launch process can take a while.

   If you have permissions for several projects, a list of projects is displayed. Select the project for your MongoDB installation.

   A MongoDB replica set is deployed on a set of Compute Engine instances according to a Deployment Manager template.

3. Accept all the default configuration settings.
4. Click **Deploy**.
5. In the Cloud Console, activate Cloud Shell.

   Activate Cloud Shell (https://console.cloud.google.com/?cloudshell=true)

   At the bottom of the Cloud Console, a Cloud Shell (/shell/docs/features) session starts and displays a command-line prompt. Cloud Shell is a shell environment with the Cloud SDK already installed, including the `gcloud` (/sdk/gcloud) command-line tool, and with values already set for your current project. It can take a few seconds for the session to initialize.

6. Use `ssh` to log in to the Compute Engine instance that runs the MongoDB primary:

   ```
   gcloud compute ssh MONGODB_VM_NAME --project PROJECT_ID --zone ZONE_OF_VM
   ```

   Replace the following:

   - ***MONGODB_VM_NAME***: the name of the primary replica of the MongoDB replica set
   - ***PROJECT_ID***: the name of your Cloud project ID
   - ***ZONE_OF_VM***: the zone in which your virtual machine (VM) instance resides

For more information, see Geography and regions (/docs/geography-and-regions).

If an SSH key is generated, the system asks for a passphrase. If you don't want to provide a passphrase, press `Enter`. If you do provide one, note it for future reference.

If you are unable to connect by using Cloud Shell, click **SSH to a servers tier VM** in the Deployment Manager.

7. Launch the `mongo` shell (https://docs.mongodb.com/manual/mongo/):

```
mongo
```

8. List the existing databases:

```
show dbs
```

The output is similar to the following:

```
admin   0.000GB
config  0.000GB
local   0.000GB
```

Keep the `mongo` shell open for upcoming commands.

You have created and accessed your MongoDB replica set and confirmed that it is operational.

Check preconditions for source database

Atlas Live Migration requires that the source MongoDB replica set meets specific configuration criteria, or preconditions. The checks are outlined in the Atlas Migration Overview (https://www.mongodb.com/cloud/atlas/migrate/migrate-from-aws) and in the Atlas Documentation (https://docs.atlas.mongodb.com/import/live-import/#prerequisites). The following commands are derived from these resources. After you verify these preconditions and make any necessary changes, your source database requires further configurations and a restart (#enable-auth).

Note: Although the source MongoDB replica set that you installed as part of this tutorial complies with the required version, you still need to check for preconditions in a production environment. For this reason, we have included a procedure here to demonstrate that check.

To check whether all preconditions are met, do the following:

1. In the `mongo` shell, check that the version of MongoDB is 2.6 or later. (In a production database instance, you open the `mongo` shell, connect to the MongoDB server by using an SSH connection, and then run this command to determine the version.)

```
db.version()
```

The output displays the version. If your version is earlier than 2.6, you need to follow the upgrade instructions (https://docs.mongodb.com/v2.6/release-notes/2.6-upgrade/).

2. Check that your current deployment is a MongoDB replica set:

```
rs.status()
```

The output is a status for the MongoDB replica set. The following output shows that a MongoDB instance started *without* the MongoDB replica set being enabled.

```
{
    "ok" : 0,
    "errmsg" : "not running with --replSet",
    "code" : 76,
    "codeName" : "NoReplicationEnabled"
}
```

In this case, stop and restart the MongoDB instance with the MongoDB replica set enabled. If you have a standalone instance of MongoDB, upgrade the MongoDB (https://docs.mongodb.com/manual/tutorial/convert-standalone-to-replica-set/) instance to a MongoDB replica set.

3. Check that authentication is enabled on your source cluster by logging in:

```
mongo -u YOUR_ADMIN_USERNAME -p --authenticationDatabase admin
```

Replace the following:

- ***YOUR_ADMIN_USERNAME***: the administrator username of your deployment

The MongoDB replica set created earlier does not have authentication enabled.

If authentication is not enabled, you need to follow instructions to enable authentication (https://docs.mongodb.com/v4.0/tutorial/enable-authentication/). The following is an example command to enable the administration, with an example username and password:

```
use admin
db.createUser(
  {
    user: "myUserAdmin",
    pwd: "myUserAdminPassword",
    roles: [ { role: "userAdminAnyDatabase", db: "admin" }, "readWriteAnyDatabase", "clusterMonitor" ]
  }
)
```

After authentication is enabled, the MongoDB role `clusterMonitor` is required in order to execute `rs.status()`. The preceding command specifies this role.

4. Check that the administrator user has the proper roles assigned for the version of the MongoDB replica set. For a list of roles that correspond to a particular version, see the discussion on source cluster security (https://docs.atlas.mongodb.com/import/live-import/#source-cluster-security) in the Atlas Live Migration documentation.

```
use admin
db.getUser("YOUR_ADMIN_USERNAME")
```

The username must be placed between quotation marks.

5. (Optional) If your MongoDB deployment is based on a version earlier than 4.2, it contains indexes with keys which exceed the 1024 byte index key limit (https://docs.mongodb.com/v4.0/reference/limits/#Index-Key-Limit). In this case, set the MongoDB server parameter `failIndexKeyTooLong` (https://docs.mongodb.com/v4.0/reference/parameters/#param.failIndexKeyTooLong) to false before starting the Atlas Live Migration procedure.

Enable authentication and restart the MongoDB replica set

To turn on authentication, key files are required in addition to creating an administrator. The following steps show how to create key files manually. In a production environment, you might consider using scripts to automate the process.

1. In Cloud Shell, create a key file:

   ```
   openssl rand -base64 756 > PATH_TO_KEY_FILE
   ```

   Replace the following:

   - ***PATH_TO_KEY_FILE***: the location where your SSH key is stored, for example, `/etc/mongo-key`

2. Enable authorization for each of the three VMs:

   a. Copy the key file to the VM:

      ```
      gcloud compute copy-files PATH_TO_KEY_FILE NAME_OF_THE_VM:PATH_TO_KEY_FILE --zone=ZONE_OF_VM
      ```

      Replace the following:

      - ***NAME_OF_THE_VM***: the name of one of the VMs running a replica of the replica set
      - ***ZONE_OF_VM***: the Google Cloud zone where the VM resides that is referred to in ***NAME_OF_THE_VM***

   b. Use `ssh` to log in to the VM and change the owner as well as the access permissions of the key file:

      ```
      sudo chown mongodb:mongodb PATH_TO_KEY_FILE

      sudo chmod 400 PATH_TO_KEY_FILE
      ```

   c. In your preferred text editor, open the `mongod.conf` file in edit mode. If you want to write back any changes, you might need to use the `sudo` command to start your text editor.

   d. Edit the `security` section as follows:

      ```
      security:
        authorization: enabled
        keyFile: PATH_TO_KEY_FILE
      ```

   e. Restart the replica:

      ```
      sudo service mongod restart
      ```

3. Verify that you can log in to the primary of the MongoDB replica set:

   ```
   mongo -u YOUR_ADMIN_USERNAME -p --authenticationDatabase admin
   ```

Insert sample data

In the following steps, you insert sample data into the source database and then verify that the documents are successfully inserted:

1. In Cloud Shell, use `ssh` to connect to the MongoDB primary Compute Engine instance:

   ```
   gcloud compute ssh MONGODB_VM_NAME --project PROJECT_ID --zone ZONE_OF_VM
   ```

   You might be required to provide the passphrase for the SSH key.

2. Start the `mongo` shell:

   ```
   mongo -u YOUR_ADMIN_USERNAME -p --authenticationDatabase admin
   ```

   Provide the password that you specified when you created the administrator username.

3. Create a database:

   ```
   use migration
   ```

4. Create a collection:

   ```
   db.createCollection("source")
   ```

5. Verify that the collection is empty:

   ```
   db.source.count()
   ```

6. Add the following five documents as the initial data set:

   ```
   db.source.insert({"document_number": 1})
   db.source.insert({"document_number": 2})
   db.source.insert({"document_number": 3})
   db.source.insert({"document_number": 4})
   db.source.insert({"document_number": 5})
   ```

   The output for each of these commands is similar to the following:

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

7. Verify that you added the five documents successfully into the collection migration. The result must be 5.

   ```
   db.source.count()
   ```

After the database migration is set up and started, these documents are migrated to the target cluster in MongoDB Atlas.

Creating a cluster in MongoDB Atlas

A MongoDB replica set is called a *cluster* in MongoDB Atlas. If you do not have a cluster set up as your target database, follow the steps in this section. These steps are based on the MongoDB documentation (https://docs.atlas.mongodb.com/tutorial/create-new-cluster/) . If you already have a cluster set up as your target database, you can skip this section.

Note: This tutorial uses a version of MongoDB Atlas with an hourly cost. If you implement this migration in production, we recommend that you use the regular hosted version of MongoDB Atlas (https://console.cloud.google.com/marketplace/details/mongodb/atlas-pro). For more information, see Costs (#costs).

1. In Cloud Marketplace, go to the **MongoDB Atlas - Free Tier Installation** page.

   Go to MongoDB Atlas on Marketplace (https://console.cloud.google.com/marketplace/details/gc-launcher-for-mongodb-atlas/mongodb-atlas)

2. Click **Visit MongoDB Site to Sign Up**.
3. Click **Launch your first cluster**.
4. Fill in the required information and click **Get started free**. Note the information that you provided.
5. Click **Advanced Configuration Options**.
6. For **Cloud Provider & Region**, select **Google Cloud Platform** and **Iowa (us-central1)**.
7. Click the **Cluster Tier** tab, and then select **M10**.
8. Click the **Additional Settings** tab, select **MongoDB 4.0 or MongoDB 4.2**, and then turn off backup.
9. Click **Create Cluster**.

   Wait until the creation of the cluster is completed. Note that the project name is `Project 0` (with a blank space) and the cluster name is `Cluster0` (without the blank space).

★ **Note:** Up-to-date screenshots of these processes are included in the MongoDB Atlas documentation (https://docs.atlas.mongodb.com/tutorial/create-new-cluster/).

The target cluster is set up and running in MongoDB Atlas.

Testing the failover of the MongoDB Atlas cluster

After the migration completes, the cluster in MongoDB Atlas executes a rolling restart. Each of the cluster members restarts in turn. In order to ensure that this process works, test the failover by following the steps in the MongoDB documentation (https://docs.atlas.mongodb.com/tutorial/test-failover/).

Starting the live migration

To migrate the data from the source to the target database, do the following:

1. Log into MongoDB Atlas (https://account.mongodb.com/account/login).
2. Go to the **Clusters** page, and then select the cluster that you want to migrate to.
3. In the target cluster (`Cluster 0`) pane, click the ellipsis button •••.
4. Select **Migrate Data to this Cluster**.

5. In the window that opens, review the information. When you're ready to migrate, click **I'm ready to migrate**.

   A window with data migration instructions is displayed. The IP addresses that are listed must be able to access the MongoDB replica set. If you have not created a firewall rule for those addresses, use Cloud Shell to add a firewall rule based on this example command:

   ```
   gcloud compute firewall-rules create "allow-mongodb-atlas" --allow=tcp:27027 --source-ranges="35.170.231.208
   ```

6. In the **Hostname:Port of the primary of your replica set** field, enter the IP address and port for the primary of the MongoDB replica set—for example, ***IP_ADDRESS*:*PORT_FOR_PRIMARY***.

   a. To determine the primary instance, run the following command in the `mongo` shell on either instance running in your Cloud project:

      ```
      rs.isMaster().primary
      ```

   b. To look up the corresponding external IP address, go to the Compute Engine **VM instances** page (https://console.cloud.google.com/compute/instances). The standard MongoDB port is `27017`.

7. Enter the administrator username and password of your MongoDB replica set.

   Leave all other settings with their default values.

8. Click **Validate**, and then do one of the following:

   - If the validation succeeds, click **Start Migration**.
   - If the validation does not succeed, troubleshoot by using the instructions that are provided. For example, if MongoDB Atlas cannot connect to the MongoDB replica set, it provides the IP addresses from which MongoDB Atlas is trying to connect. For these addresses, add a firewall rule that allows TCP traffic on port 27017 for the servers of the MongoDB replica set.

   The MongoDB Atlas screen shows the progress made. Wait for the message **Initial Sync Complete!** in the progress bar.

The initial load from the MongoDB replica set is now complete. The next step is to verify that the initial load is successful.

After the initial migration is done, MongoDB Atlas provides an estimate of the number of hours left until you must make the cutover to the target cluster. You might also receive an email from MongoDB that provides you with the number of hours left, the ability to extend that time, and a warning that if a final cutover is not made within the given time, *the migration will be canceled*.

Verifying the database migration

It is important to design and implement a database migration verification strategy to confirm that the database migration is successful. While the particular verification strategy depends on your specific use case, we recommend that you perform these checks:

- **Completeness check.** Verify that the initial document set successfully migrated from the source databases (initial load).
- **Dynamic check.** Verify that changes in the source databases are being transferred to the target databases (ongoing migration).

First, verify that the initial load is successful:

1. In MongoDB Atlas, click **Clusters**.
2. Click **Collections**.
3. Verify that a database named `migrations` exists and that the collection named `source` has five documents.

Next, verify that ongoing changes to the source databases are reflected in the target databases:

1. In Cloud Shell, use `ssh` to log in to the primary VM of the source MongoDB replica set.
2. Start the `mongo` shell. `mongo`
3. Insert another document:

   ```
   use migration
   db.source.insert({"document_number": 6})
   ```

4. In the **MongoDB Atlas Collections** page for the migration collection, click **Refresh** to observe that one document is added to the collection `source`.

You have verified that all original data from, and any ongoing changes to, the source are migrated to the target automatically by Atlas Live Migration.

Testing your Atlas target cluster

In a production environment, testing is paramount. It is necessary to test applications that access target databases to ensure that they function properly. This section discusses several testing strategies.

Test applications with a target database during a migration

As the preceding section demonstrates, you can perform application testing during an ongoing database migration. This approach might work if applications do not change the target in such a way that it conflicts with data being migrated from the source databases. Whether this approach is an option for you depends on your environment and dependencies. If the test application writes data into the target database, it might conflict with the ongoing migration.

Test applications with a temporary target database

If you cannot test applications during a production database migration, then you might migrate the data to temporary target databases that you use only for testing, and then delete the test targets after a test migration.

For this method, you stop the test migration at some point (as if the database migration is completed) and test applications against these test databases. After testing is complete, you delete the target databases and start the production database migration to migrate the data to permanent target databases. The benefit of this strategy is that the target databases can be read and written because they are only for testing.

Test applications with a target database after migration is complete

If neither of these first two strategies is viable, the remaining strategy is to test the application on the database after the migration is complete. After all data is in the target databases, test the applications before making them available to users. If testing includes writing data, then it is important that test data is written, not production data, in order to avoid production data inconsistencies. The test data must be removed once the tests are completed to avoid data inconsistencies or superfluous data in the target database.

We recommended that you back up the target databases before opening them to production access by application systems. This step helps ensure that there is a consistent starting point that you can recreate, if needed.

Cutting over from the source MongoDB replica set to the target cluster

After you complete any tests and verify that ongoing changes are reflected in the target database, you can plan the cutover.

First, you need to stop any changes to the source database so that Atlas Live Migration can drain the not-yet-migrated changes to the target. After all changes are captured in the target, you can initiate the Atlas Live Migration cutover process. After that process is complete, you can switch clients from the source to the target databases.

1. In MongoDB Atlas, click **Clusters**.
2. In the `Cluster0` pane, click **Prepare to Cutover**. A step-by-step explanation of the cutover process and a connection string to the target cluster are displayed.
3. Click **Cut over**.

   When the migration is complete, the message **Success! Your cluster migration is complete.** is displayed.

You successfully migrated your MongoDB replica set to a MongoDB Atlas cluster.

Preparing a fallback strategy

After a cutover is finished, the target cluster is the system of record; the source databases are out of date and eventually removed. However, you might want to fall back to the source databases in case of severe failures on the new target databases. For example, a failure can occur if business logic in an application is not executed during testing and then later fails to function properly. Another failure is when the performance or latency behavior does not match the source databases and causes errors.

To fall back from such failures, you might want to keep the original source databases up to date with target database changes. Atlas Live Migration does not provide a fallback mechanism. For more information on fallback strategies, see Database migration: Concepts and principles (Part 2) (/solutions/database-migration-concepts-principles-part-2).

Cleaning up

Delete the Cloud project

To avoid incurring charges to your Google Cloud account for the resources used in this tutorial, you can delete the Cloud project that you created for this tutorial.

Caution: Deleting a project has the following effects:

- **Everything in the project is deleted.** If you used an existing project for this tutorial, when you delete it, you also delete any other work you've done in the project.
- **Custom project IDs are lost.** When you created this project, you might have created a custom project ID that you want to use in the future. To preserve the URLs that use the project ID, such as an `appspot.com` URL, delete selected resources inside the project instead of deleting the whole project.

If you plan to explore multiple tutorials and quickstarts, reusing projects can help you avoid exceeding project quota limits.

1. In the Cloud Console, go to the **Manage resources** page.

   Go to the Manage resources page (https://console.cloud.google.com/iam-admin/projects)
2. In the project list, select the project that you want to delete and then click **Delete** .
3. In the dialog, type the project ID and then click **Shut down** to delete the project.

Pause or terminate the MongoDB Atlas cluster

To avoid further charges for the MongoDB Atlas cluster, you need to pause or terminate the cluster. For more information about billing implications, see Pause or terminate a cluster (https://docs.atlas.mongodb.com/pause-terminate-cluster/).

What's next

- Check out Google Cloud data migration content (/db-migration).
- Try out other Google Cloud features for yourself. Have a look at our tutorials (/docs/tutorials).

Last updated 2020-06-16.