Migrate the Controller Database to Amazon Aurora

On this page:



You have the option to migrate an existing 4.4.3 or latest on-premises Controller database to Aurora DB. The Controller might already reside in AWS, or in your data center.

Although the Controller already contains a MySQL database, you are recommended to migrate the MySQL database to Aurora because it offers replication, high availability, and elasticity. The Amazon Relational Database Service (RDS) tool handles provisioning, patching, backup, recovery, failure detection, and repair of the database. Also, Aurora DB offers encryption at rest, with encryption of all automated backups, snapshots, and replicas in the same cluster.

If your Controller is not already in AWS, then follow Migrate the Controller to migrate it. Once this is complete, you should have a Controller running on one or two EC2 instances in AWS, depending on whether or not your existing Controller deployment is high availability (HA), with the MySQL database hosted on those instances.

Commands to start and stop the database do not work with Aurora DB.

Migrate MySQL to an Aurora Database for 4.4.3 or Latest

You can create and configure Amazon Aurora to serve as the Controller database for 4.4.3 or the latest version. This process, which uses mysqldump to migrate the database, is required for Controllers running MySQL version 5.7. See Back Up the Controller with mysqldump for more information.

Since Controller upgrades to 4.4.3 from 4.3.x or earlier would use MySQL 5.5, it is important that you know what your Controller MySQL version is. Please refer to Bundled MySQL Database Version to learn how to check your MySQL version and upgrade it if necessary.

Note that running a Controller on AWS requires that some of the `cluster` parameter group and `db` parameter group settings be adjusted. See Deploy the Controller on AWS for more information.

Migrating MySQL to an Aurora Database involves the following steps:

> If you attempt to upgrade or move a Controller migrated without its liquibase-stored procedures, the upgrade will fail. You must recreate these stored procedures manually in AWS.

Step 1: Provision an Empty Aurora Database

You first need to start up a new instance of Aurora, using the desired instance type and other custom settings as explained in Deploy the Controller on AWS. Ensure that the database instance is created using port 3388.

Step 2: Use mysqldump to Export from MySQL

> Before using `mysqldump`, first ensure that the Controller app server is stopped. If you attempt to run `mysqldump` while the app server is running, it will severely degrade the performance and stability of the Controller.

To use `mysqldump`, run the `mysqldump` executable, passing the root username, password, and output file:

1. Run the following command to navigate to the executable directory:

```
cd <controller_home>/db/bin
```

2. Use the following command to export the database from MySQL:

```
./mysqldump -u root --databases controller mds_account mds_alerting mds_configuration
mds_dashboard mds_entitygraph mds_entitysearch mds_federation mds_infra_core mds_infra_core
mds_infra_server mds_license mds_metadata mds_metering mds_metric_metadata mds_rbac
mds_topology --single-transaction --compress --order-by-primary -p"<password>" > backup.sql
```

3. In order to import the resulting file into Aurora, you need to replace the following line:

```
/*!50013 DEFINER=`controller`@`localhost` SQL SECURITY DEFINER */
```

With:

```
/*!50013 DEFINER=`controller`@`%` SQL SECURITY DEFINER */
```

Step 3: Use mysqldump to export stored procedures from the AppDynamics database

1. Run the following command to export the stored procedures from the AppDynamics database.

```
./mysqldump --user=root -p --protocol=TCP --host=127.0.0.1 --port=<controller MySQL port> --
no-create-db --skip-add-drop-table --no-create-info --skip-disable-keys mysql proc --result-
file=/staging/path/for/mysql.proc.sql
```

This command, through the --result-file option, dumps the stored procedures to `/staging/path/for/mysql.proc.sql`.

2. Drop all non-mysql and non-sys stored procedures.

Step 4: Use mysql to Import to Aurora

1. Run the following command to navigate to the executable directory:

```
cd <controller_home>/db/bin
```

2. Connect to the new Aurora instance:

```
./mysql -u root -p"<password>" -h <hostname>.<aws-region>.rds.amazonaws.com -P 3388 --
protocol=TCP
```

3. Then create the Controller user, and grant it permissions:

```
CREATE USER 'controller'@'%' IDENTIFIED BY 'controller';
GRANT USAGE ON *.* TO 'controller'@'%';
GRANT ALL PRIVILEGES ON `controller`.* TO 'controller'@'%';
GRANT ALL PRIVILEGES ON `mds_account`.* TO 'controller'@'%';
GRANT ALL PRIVILEGES ON `mds_configuration`.* TO 'controller'@'%';
GRANT ALL PRIVILEGES ON `mds_license`.* TO 'controller'@'%';
GRANT ALL PRIVILEGES ON `mds_metadata`.* TO 'controller'@'%';
GRANT ALL PRIVILEGES ON `mds_metering`.* TO 'controller'@'%';
GRANT ALL PRIVILEGES ON `mds_rbac`.* TO 'controller'@'%';
GRANT ALL PRIVILEGES ON `mds_topology`.* TO 'controller'@'%';
GRANT ALL PRIVILEGES ON `mds_metric_metadata`.* TO 'controller'@'%';
GRANT ALL PRIVILEGES ON `mds_entitygraph`.* TO 'controller'@'%';
GRANT ALL PRIVILEGES ON `mds_infra_core`.* TO 'controller'@'%';
GRANT ALL PRIVILEGES ON `mds_infra_server`.* TO 'controller'@'%';
GRANT ALL PRIVILEGES ON `mds_alerting`.* TO 'controller'@'%';
GRANT ALL PRIVILEGES ON `mds_entitysearch`.* TO 'controller'@'%';
GRANT ALL PRIVILEGES ON `mds_dashboard`.* TO 'controller'@'%';
GRANT ALL PRIVILEGES ON `mds_federation`.* TO 'controller'@'%';
GRANT ALL PRIVILEGES ON `mds_auth`.* TO 'controller'@'%';
FLUSH PRIVILEGES;
```

> **Note**
>
> The Aurora database is protected by security groups to prevent access from unauthorized sources.

4. Import the database backup:

```
./mysql -u controller --port=3388 --host=<hostname>.<aws-region>.rds.amazonaws.com -p"
controller" --protocol=TCP < backup.sql
```

5. Import the stored procedures:

```
./mysql -u controller --port=3388 --host=<hostname>.<aws-region>.rds.amazonaws.com -p"controller" --
protocol=TCP < /staging/path/for/mysql.proc.sql
```

Step 5: Configure the Controller to Use the Aurora Database

1. Change the configuration in the Controller to use the Aurora DB:
    a. In the file `<controller_home>/appserver/mq/lib/props/broker/default.properties`, set the property `imq.persist.jdbc.mysql.property.url` to your Aurora database:

    ```
    imq.persist.jdbc.mysql.property.url=jdbc\:mysql\://<aurora-db>.<aws-region>.rds.
    amazonaws.com\:3388/controller
    ```

    b. In the file `<controller_home>/appserver/glassfish/domains/domain1/config/domain.xml`, set the property `serverName` to the value of your Aurora database:

    ```
    <property name="serverName" value="<aurora-db>.<aws-region>.rds.amazonaws.com"><
    /property>
    <property name="portNumber" value="3388"></property>
    ```

    Add the following line to the same file, right before the `javaagent` option:

    ```
    <jvm-options>-Dappdynamics.controller.use.global.datadir.query.for.disk.space.
    check=false</jvm-options>
    ```

    c. In the file `<controller_home>/bin/controller_maintenance.xml`, set the property `db-host` to the value of your Aurora database:

    ```
    <property name="db-host" value="<aurora-db>.<aws-region>.rds.amazonaws.com"/>
    <property name="db-port" value="3388"/>
    ```

    d. In the file `<controller_home>/bin/setup.xml`, set the property `db-host` to the value of your Aurora database:

```
<property name="db-host"value="<aurora-db>.<aws-region>.rds.amazonaws.com"/>
<property name="db-port" value="3388"/>
```

2. Remove the following cache folders from `<controller_home>/appserver/glassfish/domains/domain1` as follows:
    a. `rm -rf osgi-cache/`
    b. `rm -rf generated/`
3. With the Controller service installed, start the Controller with root:

```
service controller start
```

4. Verify that the Controller is running successfully. The local MySQL database should be shut down, and you should see the migrated data in Aurora, which can be verified via the Controller UI.

Change the Aurora Database Password

By default, the password for the "controller" user of the Aurora database used in your AppDynamics deployment is `controller`. You can change this password as follows.

To change the Aurora database password:

You can change the Aurora database password by entering the following on the command line of the EC running the Controller, substituting `new password` with a password of your choosing.

```
bin/platform-admin.sh submit-job --service controller --job update_passwords --args
controllerDBPassword={newPassword} controllerDBHost={auroraHost}
```