CRQ000001898701: AWS Aurora RDS PostgreSQL upgrade from 12.x to 15.x

⚠ This document is specific to Aurora RDS PostgreSQL upgrade from 11.x to 12.x. For any other versions do refer to the official AWS documents for those versions and create a separate document for the implementation.

Announcement: Amazon Aurora PostgreSQL 12.14 end of support is March 31, 2024

https://docs.aws.amazon.com/AmazonRDS/latest/PostgreSQLReleaseNotes/postgresql-release-calendar.html#Release.Calendar

REFERENCE

https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/USER_UpgradeDBInstance.PostgreSQL.html#USER_UpgradeDBInstance.PostgreSQL.UpgradeVersion

Overview of upgrading PostgreSQL

Mendix Compatibility	Mendix version 8,9,10 supports PostgreSQL 12, 13, 14, 15, 16
Validate Supported DB engines for DB instance classes	Refer doc For Aurora Postgresql 15.5, instance class has to be at least db.r5.large, db.r5.2xlarge
Pre Steps	• Verify that there are no uses of unsupported *reg** data types, use the following query for each database.

```
SELECT count(*) FROM pg_catalog.pg_class c, pg_catalog.pg_namespace n, pg_catalog.pg_attribute a
  WHERE c.oid = a.attrelid
      AND NOT a.attisdropped
      AND a.atttypid IN ('pg_catalog.regproc'::pg_catalog.regtype,
'pg_catalog.regprocedure'::pg_catalog.regtype,
'pg_catalog.regoper'::pg_catalog.regtype,
```

```
'pg_catalog.regoperator'::pg_catalog.regtype,
'pg_catalog.regconfig'::pg_catalog.regtype,
'pg_catalog.regdictionary'::pg_catalog.regtype)
      AND c.relnamespace = n.oid
      AND n.nspname NOT IN ('pg_catalog', 'information_schema');
```

- To find the `unknown` data type in your database:

```
SELECT DISTINCT data_type FROM information_schema.columns WHERE data_type ILIKE 'unknown';
```

- To list your currently installed extensions:

```
SELECT * FROM pg_extension;
```

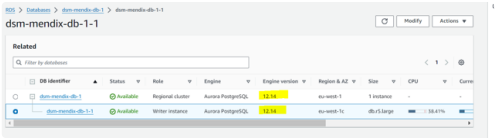
- To locate invalid hash indexes, run the following SQL for each database that contains hash indexes.

```
SELECT idx.indrelid::regclass AS table_name,
   idx.indexrelid::regclass AS index_name
FROM pg_catalog.pg_index idx
   JOIN pg_catalog.pg_class cls ON cls.oid = idx.indexrelid
   JOIN pg_catalog.pg_am am ON am.oid = cls.relam
WHERE am.amname = 'hash'
AND NOT idx.indisvalid;
```

Check current version

Sign in to the AWS Management Console, use the region selector in the navigation bar to choose the AWS Region for your deployment, and open the AWS RDS console at https://console.aws.amazon.com/rds.

In the navigation pane, click **Databases**. From there, select your database instance



Determining which engine version to upgrade to

As per [AWS doc](),

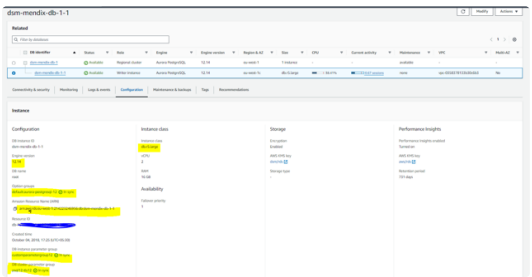
```
aws --region eu-west-1 rds describe-db-engine-versions  --engine aurora-postgresql  --engine-version 12.14 --query 'DBEngineVersions[].ValidUpgradeTarget[?IsMajorVersionUpgrade == `true`]'
```

How to perform a major version upgrade

Major version upgrades can contain database changes that are not backward-compatible with previous versions of the database. This functionality can cause your existing applications to stop working correctly. As a result, Amazon Aurora doesn't apply major version upgrades automatically. To perform a major version upgrade, you modify your DB cluster manually.

Before applying an upgrade to your production DB clusters, make sure that you thoroughly test any upgrade to verify that your applications work correctly.

Check if using custom parameter/options groups or the default for the version. Since we are using the default, specify the default DB instance, DB cluster parameter group, or both for the new DB engine version.



Disable automatic deployment trigger for all applications (Major / Minor Version)

Stop all applications using the database. We scale down all the deployments to 0 to stop all the pods using the database

```
kubectl scale deploy -n mendix --replicas=0 --all
```

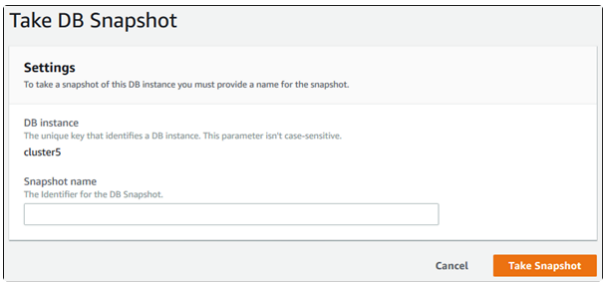
Validate from the RDS monitoring graphs that sessions are down to 0.

Perform a backup.

The upgrade process creates a DB cluster snapshot of your DB cluster during upgrading. If you also want to do a manual backup before the upgrade process, follow below steps:

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Databases**.
3. In the list of DB instances, choose a writer instance for the DB cluster.
4. Choose **Actions**, and then choose **Take snapshot**.

   The **Take DB Snapshot** window appears.
5. Enter the name of the DB cluster snapshot in the **Snapshot name** box.



6. Choose **Take Snapshot**.

Check for unsupported usage:

Commit or roll back all open prepared transactions before attempting an upgrade. You can use the following query to verify

that there are no open prepared transactions on your instance.

```
SELECT count(*) FROM pg_catalog.pg_prepared_xacts;
```

Remove all uses of the *reg** data types before attempting an upgrade.

Except for `regtype` and `regclass`, you can't upgrade the *reg** data types. The pg_upgrade utility can't persist this data type, which is used by Amazon Aurora to do the upgrade.

To verify that there are no uses of unsupported *reg** data types, use the following query for each database.

```

SELECT count(*) FROM pg_catalog.pg_class c, pg_catalog.pg_namespace n, pg_catalog.pg_attribute a
  WHERE c.oid = a.attrelid
      AND NOT a.attisdropped
      AND a.atttypid IN ('pg_catalog.regproc'::pg_catalog.regtype,
'pg_catalog.regprocedure'::pg_catalog.regtype,
'pg_catalog.regoper'::pg_catalog.regtype,
'pg_catalog.regoperator'::pg_catalog.regtype,
'pg_catalog.regconfig'::pg_catalog.regtype,
'pg_catalog.regdictionary'::pg_catalog.regtype)
      AND c.relnamespace = n.oid
      AND n.nspname NOT IN ('pg_catalog', 'information_schema');
```

Upgrade certain extensions to the latest available version before performing the major version upgrade. The extensions to update include the following:

Run the following command for each extension that you are using.

- `pgRouting`
- `postgis_raster`
- `postgis_tiger_geocoder`
- `postgis_topology`
- `address_standardizer`
- `address_standardizer_data_us`

```
ALTER EXTENSION PostgreSQL-extension
UPDATE TO 'new-version'
```

Drop `unknown` data types, depending on your target version.

PostgreSQL version 10 stopped supporting the `unknown` data type. If a version 9.6 database uses the `unknown` data type, an upgrade to a version 10 shows an error message such as the following.

```
Database instance is in a state that
cannot be upgraded: PreUpgrade checks
failed:
The instance could not be upgraded because
the 'unknown' data type is used in user
tables.
Please remove all usages of the 'unknown'
data type and try again."
```

To find the `unknown` data type in your database so you can remove the offending column or change it to a supported data type, use the following SQL code.

```
SELECT DISTINCT data_type FROM
information_schema.columns WHERE data_type
ILIKE 'unknown';
```

Perform a dry run upgrade.

Consider testing your application on the upgraded database with a similar workload to verify that everything works as expected. After the upgrade is verified, you can delete this test instance.

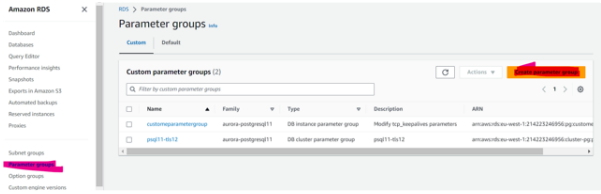
We highly recommend testing a major version upgrade on a duplicate of your production database before trying the upgrade on your production database. To create a duplicate test instance, you can

either restore your database from a recent snapshot or clone your database.

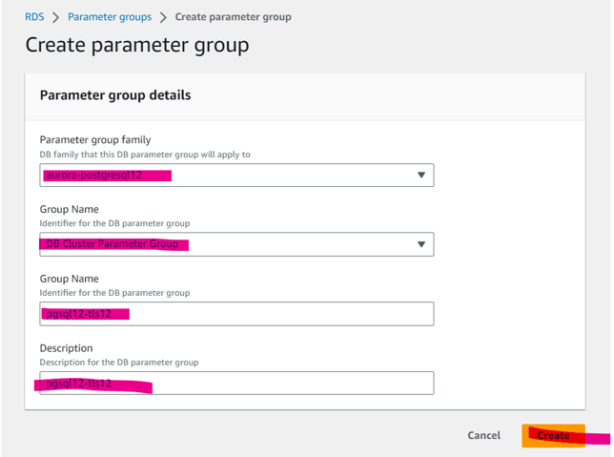
As the Mendix application will only support the TLS 1.2 version, we need to create a custom parameter group for the database (v12.14) upgrade for both the DB instance and DB cluster. In the parameter group, we should set the TLS v1.2 as the default value.

To create custom parameter group and set the TLS v1.2 as the default value for the both the DB instance and DB cluster by using the console

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Parameter groups**, and then click on the create parameter group.

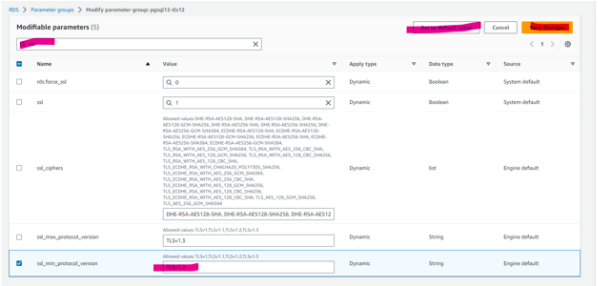


3. On the "**Create parameter group**" page, select the desired parameter group family, specify the Group Name (either DB parameter or DB cluster parameter), provide a Group Name and Description, and then click on the "Create" button.



4. To set the TLS 1.2 version as the default, open the custom created parameter

group, click on "Edit," search for "ssl_min_protocol_version," which contains the TLS 1.2 version, then click on "Set to default value," and finally, save the changes.



Upgrade your instance.

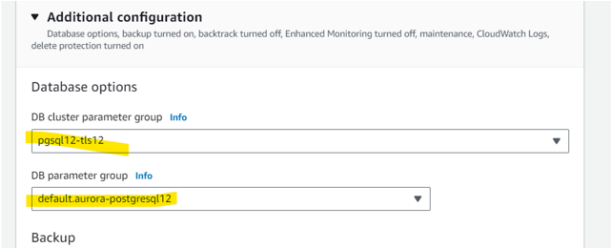
> During the upgrade process, you can't do a point-in-time restore of your cluster. Aurora PostgreSQL takes a DB cluster snapshot during the upgrade process if your backup retention period is greater than 0. You can perform a point-in-time restore to times before the upgrade began and after the automatic snapshot of your instance has completed.

For information about an upgrade in progress, you can use Amazon RDS to view two logs that the pg_upgrade utility produces. These are `pg_upgrade_internal.log` and `pg_upgrade_server.log`. Amazon Aurora appends a timestamp to the file name for these logs. You can view these logs as you can any other log.

Sample Logs

To upgrade the engine version of a DB cluster by using the console

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Databases**, and then choose the DB cluster that you want to upgrade.
3. Choose **Modify**. The **Modify DB cluster** page appears.
4. For **Engine version**, choose the new version.
5. select the custom parameter group for **DB cluster** and **DB parameter group** which we created earlier to set the TLS 1.2 as the default version



```
October 28, 2021, 4:29:25 AM UTC Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-dsm-mendix-db-1-9-6-19-to-10-18-2021-10-28-04-25].
October 28, 2021, 4:31:19 AM UTC Upgrade in progress: Cloning volume.
October 28, 2021, 4:33:36 AM UTC Upgrade in progress: Upgrading writer.
October 28, 2021, 5:04:41 AM UTC Database cluster major version has been upgraded
```

6. Choose **Continue** and check the summary of modifications.
7. To apply the changes immediately, choose **Apply immediately**. Choosing this option can cause an outage in some cases. For more information, see Modifying an Amazon Aurora DB cluster.
8. On the confirmation page, review your changes. If they are correct, choose **Modify Cluster** to save your changes.

   Or choose **Back** to edit your changes or **Cancel** to cancel your changes.

Upgrade PostgreSQL extensions (If applicable)

A PostgreSQL engine upgrade doesn't automatically upgrade any PostgreSQL extensions. To update an extension after an engine upgrade, use the `ALTER EXTENSION UPDATE` command.

> If you are running the `PostGIS` extension in your Amazon RDS PostgreSQL DB instance, make sure that you follow the PostGIS upgrade instructions in the PostGIS documentation before you upgrade the extension.

To upgrade an extension, use the following command.

```
ALTER EXTENSION extension_name UPDATE TO 'new_version';
```

To list your currently installed extensions, use the PostgreSQL pg_extension catalog in the following command.

```
SELECT * FROM pg_extension;
```

To view a list of the specific extension versions that are available for your installation, use the PostgreSQL pg_available_extension_versions view in the following command.

```
SELECT * FROM pg_available_extension_versions;
```

Validate db connection from pg admin

- Connect to pgadmin and connect to the few application database and check

After you complete a major version upgrade, following is recommended. At least for large databases execute ANALYZE.

- Run the `ANALYZE` operation to refresh the `pg_statistic` table.
- Connect to pgadmin and connect to the each database and execute:

```
ANALYZE VERBOSE;
```

Run `REINDEX` on any hash indexes

- If you upgraded to PostgreSQL version 15, run `REINDEX` on any hash indexes you have. Hash indexes were changed in version 15 and must be rebuilt. To locate invalid hash indexes, run the following SQL for each database that contains hash indexes.

```
SELECT idx.indrelid::regclass AS table_name,
   idx.indexrelid::regclass AS index_name
FROM pg_catalog.pg_index idx
   JOIN pg_catalog.pg_class cls ON cls.oid = idx.indexrelid
   JOIN pg_catalog.pg_am am ON am.oid = cls.relam
WHERE am.amname = 'hash'
AND NOT idx.indisvalid;
```

Start few sample application before starting all applications

```
kubectl scale deployment.apps/<DEPLOYMENT_NAME> -n mendix --replicas=1
```

Validate application logs

```
INFO - ConnectionBus: Database: PostgreSQL 15.5, name: 'jdbc:postgresql://dsm-mendix-db-1.cluster-cxasoi2wccfm.eu-west-1.rds.amazonaws.com:5432/dfs-materials?tcpKeepAlive=true'
Driver: PostgreSQL JDBC Driver 42.2.5
```

Validate from the RDS monitoring graphs that sessions are coming in to the database

Start all the applications by scaling up the deployments.

```
kubectl scale deploy -n mendix --replicas=1 --all
```

Enable automatic deployment trigger for all applications

Lesson learned from previous upgrade

Connection timed out error in dev db upgrade from 12.14 to 15.5