

CRQ000001930210: AWS Aurora RDS PostgreSQL upgrade from 15.x to 16.x

⚠ This document is specific to Aurora RDS PostgreSQL upgrade from 15.x to 16.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

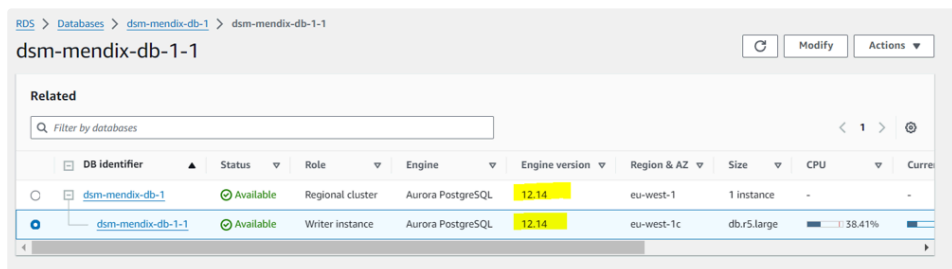
https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/USER_UpgradeDBInstance.PostgreSQL.html#USER_UpgradeDBInstance.PostgreSQL.UpgradeVersion

Overview of upgrading PostgreSQL

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](#),

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

Prerequisites

1. Mendix Compatibility

Mendix version [8.9.10](#) supports PostgreSQL 12, 13, 14, 15, 16

2. Validate Supported DB engines for DB instance classes

Refer [doc](#)

For Aurora Postgresql 16.6, instance class has to be at least db.r5.large

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

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.

```
1 SELECT count(*) FROM pg_catalog.pg_class c, pg_catalog.pg_namespace n,  
   pg_catalog.pg_attribute a  
2   WHERE c.oid = a.attrelid  
3     AND NOT a.attisdropped  
4     AND a.atttypid IN ('pg_catalog.regproc'::pg_catalog.regtype,  
5                        'pg_catalog.regprocedure'::pg_catalog.regtype,  
6                        'pg_catalog.regoper'::pg_catalog.regtype,  
7                        'pg_catalog.regoperator'::pg_catalog.regtype,  
8                        'pg_catalog.regconfig'::pg_catalog.regtype,  
9                        'pg_catalog.regdictionary'::pg_catalog.regtype)  
10    AND c.relnamespace = n.oid  
11    AND n.nspname NOT IN ('pg_catalog', 'information_schema');
```

4. To find the `unknown` data type in your database:

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

5. To list your currently installed extension

```
1 SELECT * FROM pg_extension;
```

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

```
1 SELECT idx.indrelid::regclass AS table_name,  
2        idx.indexrelid::regclass AS index_name  
3 FROM pg_catalog.pg_index idx  
4     JOIN pg_catalog.pg_class cls ON cls.oid = idx.indexrelid  
5     JOIN pg_catalog.pg_am am ON am.oid = cls.relam
```

```
6 WHERE am.amname = 'hash'
7 AND NOT idx.indisvalid;
```

Update the below DB cluster parameters in blue Database

1. **rds.logically_replicate_unlogged_tables** should be on (set value as 1).

psql15-tls12

Details

Parameter Group Type: Custom | Resource Type: DB cluster | Parameter group family: aurora-postgresql15

Parameters (410) [Info](#)

Q rds.logically_replicate_unlogged_tables X 1 match

Name	Value	Apply type	Data type
rds.logically_replicate_unlogged_tables	1	Static	Boolean

2. **rds.logical_replication** should be on (set value as 1)

Details

Parameter Group Type: Custom | Resource Type: DB cluster | Parameter group family: aurora-postgresql15

Parameters (410) [Info](#)

Q rds.logi X 4 matches

Name	Value	Apply type	Data type
rds.logical_decoding_work_disk	-	Dynamic	Integer
rds.logical_replication	1	Static	Boolean

3. **max_replication_slots** – This must be set to at least the number of subscriptions expected to connect, plus some reserve for table synchronization. There will be one subscription per database, so make sure to set a number greater than the number of databases.

Details

Parameter Group Type: Custom | Resource Type: DB cluster | Parameter group family: aurora-postgresql15 | Description: psql15-tls12

Parameters (410) [Info](#)

Q max_replic X 1 match

Name	Value	Apply type	Data type	Value type	Source
max_replication_slots	145	Static	Integer	Modifiable	Modified

4. **max_wal_senders** – This is the maximum number of background processes that the system can support. It's recommended to set this number slightly higher than max_replication_slots.

Details

Parameter Group Type

Custom

Resource Type

DB cluster

Parameter group family

aurora-postgresql15

Parameters (410)

Info

Q

max_wal_

X

1 match

Name	Value	Apply type	Data type
max_wal_senders	147	Static	Integer

5. **max_logical_replication_worker** – You should set this to a number of databases, plus some reserve for the table synchronization workers and parallel apply workers.

Details

Parameter Group Type

Custom

Resource Type

DB cluster

Parameter group family

aurora-postgresql15

Parameters (410)

Info

Q max_logical

X

1 match

Name

▲

Value

▼

Apply type

▼

Data type

max_logical_replication_workers

146

Static

Integer

6. **max_worker_processes** – This is the maximum number of background processes that the system can support. It should be set to at minimum `max_logical_replication_worker + 1` or higher.

Details

Parameter Group Type

Custom

Resource Type

DB cluster

Parameter group family

aurora-postgresql15

Parameters (410)

Info

Q

max_work

X

2 matches

Name	Value	Apply type
autovacuum_max_workers	GREATEST((DBInstanceClassMemory/64371566592),3)	Static
max_worker_processes	147	Static

7. In addition, make sure that the '**synchronous_commit**' parameter is set to on.

Details

Parameter Group Type

Custom

Resource Type

DB cluster

Parameter group family

aurora-postgresql15

Parameters (410)

Info

Q

synchronous_commit

X

2 matches

Name	Value	Apply type	Data type
pglogical.synchronous_commit	-	Static	Boolean
synchronous_commit	on	Dynamic	String

- After you configure the required parameters, reboot the DB instance so that your changes take effect.

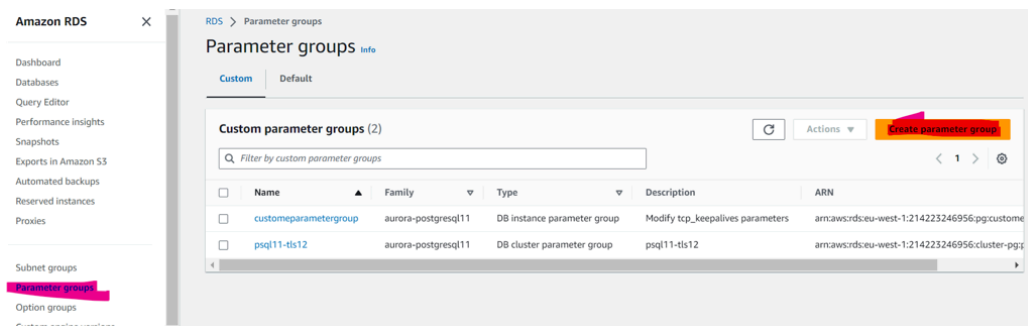
Note : Make sure that all tables in the DB cluster have a primary key. PostgreSQL logical replication doesn't allow UPDATE or DELETE operations on tables that don't have a primary key.

- **Create cluster parameters for higher (selected) version of AWS RDS (16.6 V).**

As the Mendix application will only support the TLS 1.2 version, we need to create a custom parameter group for the database (v16.6) 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.

[RDS](#) > [Parameter groups](#) > Create parameter group

Create parameter group

Parameter group details

Parameter group family

DB family that this DB parameter group will apply to

aurora-postgresql12

Group Name

Identifier for the DB parameter group

DB Cluster Parameter Group

Group Name

Identifier for the DB parameter group

pgsql12-tls12

Description

Description for the DB parameter group

pgsql12-tls12

[Cancel](#) [Create](#)

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.

RD5
Parameter groups
Modify parameter group: pgsql12-tls1.2

Modifiable parameters (5)

X

	Name	Value	Apply type	Data type	Source
<input type="checkbox"/>	rd5.force_ssl	<input type="text" value="0"/> X	Dynamic	Boolean	System default
<input type="checkbox"/>	ssl	<input type="text" value="1"/> X	Dynamic	Boolean	System default
<input type="checkbox"/>	ssl_ciphers	<div> Allowed values DHE-RSA-AES128-SHA, DHE-RSA-AES128-SHA256, DHE-RSA-AES128-GCM-SHA256, DHE-RSA-AES256-SHA, DHE-RSA-AES256-SHA256, DHE-RSA-AES256-GCM-SHA384, ECDHE-RSA-AES128-SHA, ECDHE-RSA-AES128-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-RSA-AES256-SHA, ECDHE-RSA-AES256-SHA384, ECDHE-RSA-AES256-GCM-SHA384, TLS_RSA_WITH_AES_256_GCM_SHA384, TLS_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_128_GCM_SHA256, TLS_RSA_WITH_AES_128_CBC_SHA256, TLS_RSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256, TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384, TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA, TLS_AES_128_GCM_SHA256, TLS_AES_256_GCM_SHA384 </div> <input type="text" value="DHE-RSA-AES128-SHA, DHE-RSA-AES128-SHA256, DHE-RSA-AES128-SHA"/>	Dynamic	list	Engine default
<input type="checkbox"/>	ssl_max_protocol_version	<div>Allowed values TLSv1,TLSv1.1,TLSv1.2,TLSv1.3</div> <input type="text" value="TLSv1.3"/>	Dynamic	String	Engine default
<input checked="" type="checkbox"/>	ssl_min_protocol_version	<div>Allowed values TLSv1,TLSv1.1,TLSv1.2,TLSv1.3</div> <input type="text" value="TLSv1.2"/>	Dynamic	String	Engine default

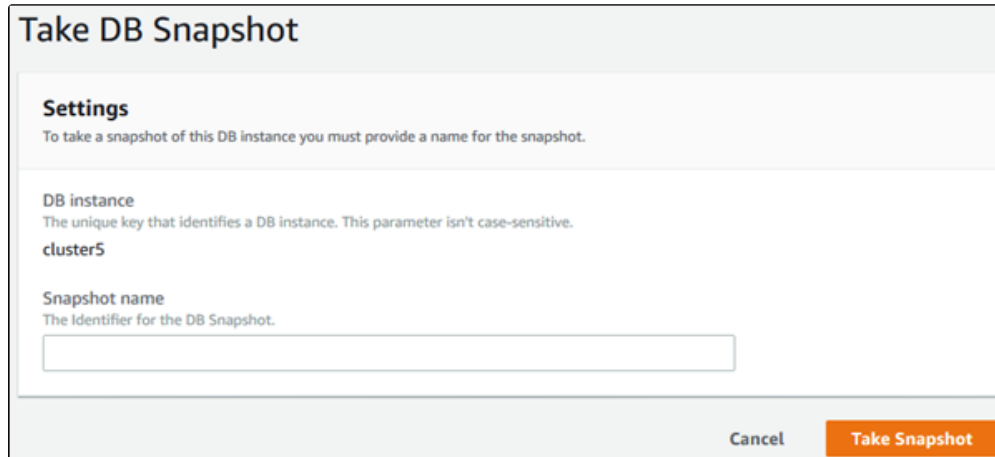
- **Create a Manual Snapshot of AWS RDS**

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.

The image shows a 'Take DB Snapshot' dialog box. It has a title bar 'Take DB Snapshot'. Below the title bar is a 'Settings' section with a note: 'To take a snapshot of this DB instance you must provide a name for the snapshot.' There are two input fields: 'DB instance' with the value 'cluster5' and 'Snapshot name' which is empty. At the bottom right are 'Cancel' and 'Take Snapshot' buttons.

Take DB Snapshot

Settings
To take a snapshot of this DB instance you must provide a name for the snapshot.

DB instance
The unique key that identifies a DB instance. This parameter isn't case-sensitive.
cluster5

Snapshot name
The Identifier for the DB Snapshot.

Cancel Take Snapshot

6. Choose **Take Snapshot**.

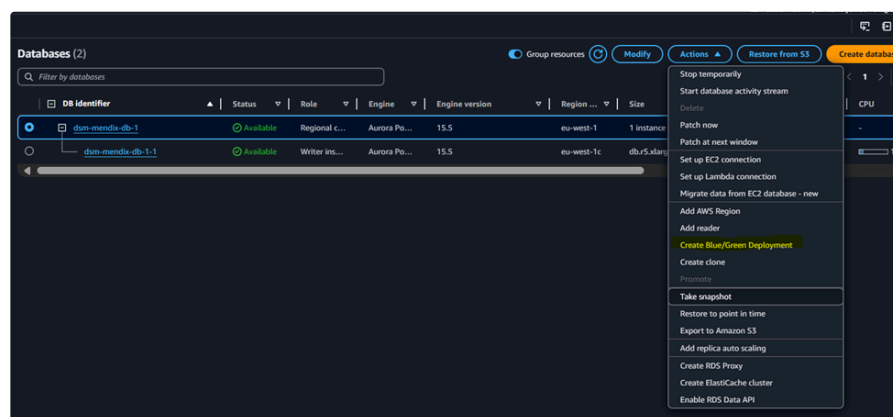
- **Check for unsupported usage:**

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.

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

Steps to create blue/green Deployment

1. Navigate to AWS PostgreSQL RDS
2. Select DB cluster and click on Action then choose Blue/Green Deployment



3. Review the blue database identifiers. Make sure that they match the DB instances that you expect in the blue environment. If they don't, choose cancel.
4. For Blue/Green Deployment identifier, enter a name for your blue/green deployment.

Create Blue/Green Deployment

Create a Blue/Green Deployment that clones the resources of your current production environment (blue) to a staging environment (green). You can modify the green environment without affecting the blue environment.

Identifiers

Blue database identifier: `dam-memdb-1`

Blue/Green Deployment identifier: `bg-deployment-1`

Next

5. Select higher engine version of Green deployment
6. Select the parameter group.

Configure Blue/Green Deployment: bg-deployment-1

Engine configurations

Blue instance configuration: Blue engine version 15.5

Blue DB cluster parameter group: `pg15-1612`

Blue DB parameter group: `default-aurora-postgresql15`

Green instance configuration: Green engine version 16.4

Choose the engine version for green databases: `Aurora PostgreSQL (Compatible with PostgreSQL 16.4) - recommended`

Choose the DB cluster parameter group for green databases: `pg16-1612`

Choose the DB parameter group for green databases: `default-aurora-postgresql16`

Next

7. [In the remaining sections, specify the settings for the green environment. For information about each setting, see \[Settings for creating blue/green deployments\]\(#\).](#)
8. Choose Create staging environment.

Review and confirm

Step 1: Create Blue/Green Deployment

Blue/Green Deployment identifier: `bg-deployment-1`

Step 2: Configure Blue/Green Deployment

Configuration details

	Blue database	Green database
DB cluster ID	<code>dam-memdb-1</code>	<code>dam-memdb-1-green</code>
Engine	Aurora PostgreSQL	Aurora PostgreSQL
Engine version	15.5	16.4 *
DB Parameter group	<code>default-aurora-postgresql15</code>	<code>default-aurora-postgresql16</code> *
DB cluster Parameter group	<code>pg15-1612</code>	<code>pg16-1612</code> *

* Indicates change from your blue database.

Estimated monthly costs for green database 447.30 USD

A Blue/Green Deployment creates new databases in the green environment. The costs for the databases on the green environment are similar to the costs for the databases in the blue environment. These costs include the current standard pricing for the DB instances, storage, I/Os, in addition to enabled features such as a Multi-AZ deployment, backups, and Amazon RDS Performance Insights.

Create

Cluster Creating Status

Databases (4)

Filter by databases

DB Identifier	Status	Role	Engine	Engine version
dsd-mendix-db-1 Blue	Available	Regional cluster	Aurora PostgreSQL	15.5
dsd-mendix-db-1-1 Blue	Available	Writer instance	Aurora PostgreSQL	15.5
bg-deployment-1	Provisioning	Blue/Green Deployment	-	-
dsd-mendix-db-1-green-kcp54h Green	Creating	Regional cluster	Aurora PostgreSQL	15.5

DB instance creating status

Databases (5)

Filter by databases

DB Identifier	Status	Role	Engine	Engine version	Region
dsd-mendix-db-1 Blue	Available	Regional cluster	Aurora PostgreSQL	15.5	eu-west-1
dsd-mendix-db-1-1 Blue	Available	Writer instance	Aurora PostgreSQL	15.5	eu-west-1
bg-deployment-1	Provisioning	Blue/Green Deployment	-	-	-
dsd-mendix-db-1-green-kcp54h Green	Available	Regional cluster	Aurora PostgreSQL	15.5	eu-west-1
dsd-mendix-db-1-1-green-lbs8lj Green	Creating	Reader instance	Aurora PostgreSQL	15.5	eu-west-1

DB upgrade status

Amazon RDS

dsd-mendix-db-1-1

Related

DB Identifier	Status	Role	Engine	Engine version	Region & AZ	Size	CPU	Current activity	Maintenance	VPC	Multi-AZ
dsd-mendix-db-1 Blue	Available	Regional cluster	Aurora PostgreSQL	15.5	eu-west-1	1 instance	-	-	available	-	-
dsd-mendix-db-1-1 Blue	Available	Writer instance	Aurora PostgreSQL	15.5	eu-west-1	db.r5.large	17.54%	111.50 seconds	none	vpc-0158178113b...	No
bg-deployment-1	Provisioning	Blue/Green Deployment	-	-	-	-	-	-	-	-	-
dsd-mendix-db-1-green-kcp54h	Upgrading	Regional cluster	Aurora PostgreSQL	15.5	eu-west-1	1 instance	-	-	-	-	-
dsd-mendix-db-1-1-green-lbs8lj	Upgrading	Writer instance	Aurora PostgreSQL	15.5	eu-west-1	db.r5.large	25.79%	0.26 seconds	none	vpc-0158178113b...	No

Connectivity & security | Monitoring | Logs & events | Configuration | Maintenance & backups | Tags | Recommendations

CloudWatch (4)

Available status of blue/green deployment

Amazon RDS

bg-deployment-1

Related

DB Identifier	Status	Role	Engine	Engine version	Region & AZ	Size	CPU	Current activity	Maintenance	VPC	Multi-AZ
dsd-mendix-db-1 Blue	Available	Regional cluster	Aurora PostgreSQL	15.5	eu-west-1	1 instance	-	-	available	-	-
dsd-mendix-db-1-1 Blue	Available	Writer instance	Aurora PostgreSQL	15.5	eu-west-1	db.r5.large	28.02%	2.48 seconds	none	vpc-0158178113b...	No
bg-deployment-1	Available	Blue/Green Deployment	-	-	-	-	-	-	-	-	-
dsd-mendix-db-1-green-kcp54h	Available	Regional cluster	Aurora PostgreSQL	16.6	eu-west-1	1 instance	-	-	-	-	-
dsd-mendix-db-1-1-green-lbs8lj	Available	Writer instance	Aurora PostgreSQL	16.6	eu-west-1	db.r5.large	22.29%	0.26 seconds	none	vpc-0158178113b...	No

Some green environment settings are different from blue environment settings:
The blue instance engine version is 15.5 and the green instance engine version is 16.6.

Connectivity & security | Monitoring | Logs & events | Configuration | Status | Tags | Recommendations

Blue connectivity and security Blue

Endpoint and port

Endpoint: dsd-mendix-db-1-1.cxas22wcfm.eu-west-1.rds.amazonaws.com

Port: 5432

Networking

Availability Zone: eu-west-1

VPC: dsd-vpc-0158178113b...

Green connectivity and security Green

Endpoint and port

Endpoint: dsd-mendix-db-1-1-green-lbs8lj.cxas22wcfm.eu-west-1.rds.amazonaws.com

Port: 5432

Networking

Availability Zone: eu-west-1

VPC: dsd-vpc-0158178113b...

9. [For information about modifying a DB cluster, see Modifying an Amazon Aurora DB cluster.](#)

Test your staging environment.

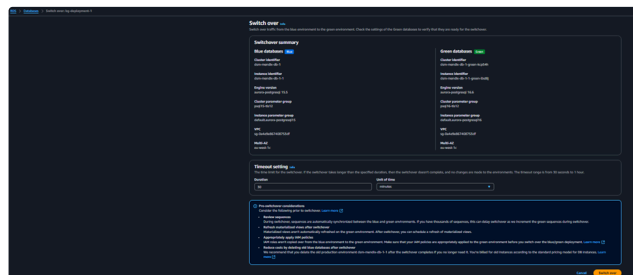
During testing, we recommend that you keep your databases in the green environment read only. Enable write operations on the green environment with caution because they can result in replication conflicts. They can also result in unintended data in the production databases after switchover. For Aurora PostgreSQL, set the `default_transaction_read_only` parameter to off at the session level

Validate db connections

- Connect to pgadmin and connect to the few application database and check
- Validate application logs
- Validate from the RDS monitoring graphs that sessions are coming in to the database

Switching over a blue/green deployment

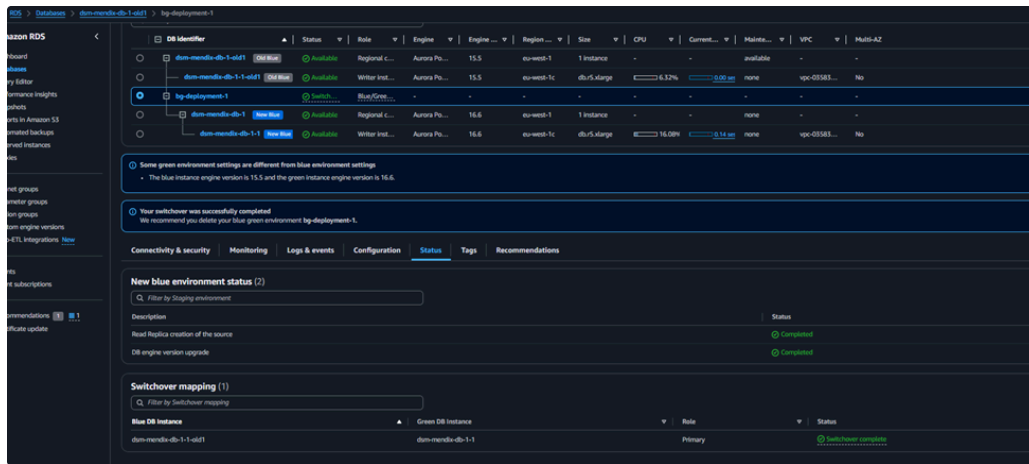
1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. In the navigation pane, choose Databases, and then choose the blue/green deployment that you want to switch over.
3. For Actions, choose Switch over.
4. On the Switch over page, review the switchover summary. Make sure the resources in both environments match what you expect. If they don't, choose Cancel.



5. For Timeout settings, enter the time limit for switchover.

Deleting a blue/green deployment in Amazon Aurora

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>
2. Before delete validate the status of switchover mapping as it should be switchover complete as shown in below snap.



3. In the navigation pane, choose Databases, and then choose the blue/green deployment that you want to delete.
4. For Actions, choose Delete. - The Delete Blue/Green Deployment? window appears.
5. To delete the green databases, select Delete the green databases in this Blue/Green Deployment.
6. Enter delete me in the box.
7. Choose Delete.

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

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

Drop unknown data types, depending on your target version.

PostgreSQL version 10 stopped supporting the unknown data type. If a above 10

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

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

To find the unknown data type in your database so you can remove the offending column or change it to a


version database uses

the `unknown` data type, an upgrade to a version 10 shows an error message such as the following.

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

• 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.

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

To list your currently installed extensions, use the PostgreSQL [pg_extension](#) catalog in the following command.

```
1 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.

```
1 SELECT * FROM pg_available_extension_versions;
```

• 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:

```
1 ANALYZE VERBOSE;
```

Run `REINDEX` on any hash indexes

supported data type, use the following SQL code.

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

If you upgraded to PostgreSQL version 16, run **REINDEX** on any hash indexes you have. Hash indexes were changed in version 16 and must be rebuilt. To locate invalid hash indexes, run the following SQL for each database that contains hash indexes.

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