

Migrating from MySQL to Amazon RDS

A step-by-step guide for migrating a self-managed MySQL database instance to a fully managed database on Amazon Relational Database Service (Amazon RDS)

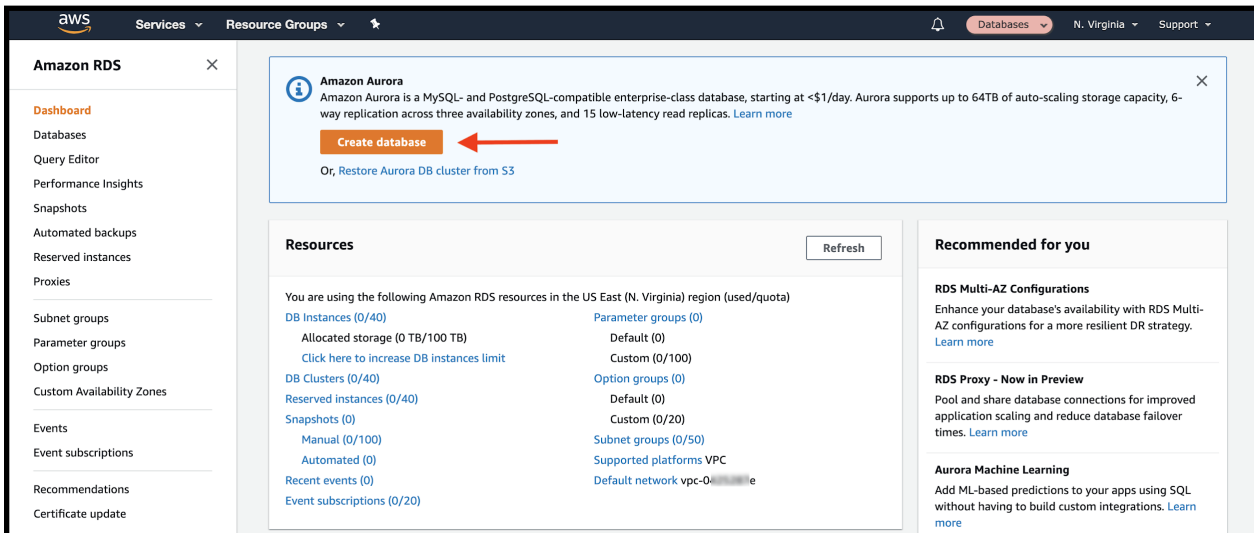
Steps:

1. Create a MySQL database instance in Amazon RDS [2](#)
2. Create a replication instance in AWS DMS [10](#)
3. Create endpoints in AWS DMS [16](#)
4. Create a replication task in AWS DMS [20](#)
5. Complete the migration and clean up resources [25](#)

1. Create a MySQL database instance in Amazon RDS

In this step, you will create a MySQL database instance in Amazon RDS. This instance will be used as your primary database after you copy your existing data into it by using AWS Database Migration Service (AWS DMS).


To get started, navigate to the [Amazon RDS console](#). On the main page, choose Create database.





This initiates the database creation wizard. In the **Engine options** section, choose **MySQL** as the **Engine type**. Then choose the version of MySQL you want to use. Note that AWS DMS supports only MySQL versions 5.5, 5.6, and 5.7 (as of August of 2020).


Engine options


Engine type [Info](#)


☐ Amazon Aurora


☒ MySQL


☐ MariaDB


☐ PostgreSQL


☐ Oracle



☐ Microsoft SQL Server


Edition

☒ MySQL Community

Version [Info](#)

MySQL 5.7.28

 **Known Issues/Limitations**
Review the [Known Issues/Limitations](#) to learn about potential compatibility issues with specific database versions.

The database creation wizard includes templates to make it easier to configure the settings for your Amazon RDS database. If you are creating this database to be used in production, you should choose the production template.

In the **Settings** section, give your database a name, and set the master username and password. Do not autogenerate your password for this lab, and make sure you write down your password. You need the password to connect to your database and create additional users.

Settings

DB instance identifier [Info](#)

Type a name for your DB instance. The name must be unique cross all DB instances owned by your AWS account in the current AWS Region.

my-database

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric characters or hyphens (1 to 15 for SQL Server). First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

▼ Credentials Settings

Master username [Info](#)

Type a login ID for the master user of your DB instance.

admin

1 to 16 alphanumeric characters. First character must be a letter

☐ Auto generate a password

Amazon RDS can generate a password for you, or you can specify your own password

Master password [Info](#)

.....

Constraints: At least 8 printable ASCII characters. Can't contain any of the following: / (slash), "(double quote) and @ (at sign).

Confirm password [Info](#)

.....

Next, choose the database instance size. You should choose the size based on your estimated capacity. If you are managing your own database on [Amazon Elastic Cloud Compute](#), or Amazon EC2, you can compare your current Amazon EC2 instance size to an Amazon RDS instance size.

If you want to increase or decrease your database instance size in the future, Amazon RDS enables you to do that easily. However, you may incur some downtime to change your database size.

aws training and
certification

Page 4

DB instance size

DB instance class [Info](#)

Choose a DB instance class that meets your processing power and memory requirements. The DB instance class options below are limited to those supported by the engine you selected above.

☒ Standard classes (includes m classes)

☐ Memory Optimized classes (includes r and x classes)

☐ Burstable classes (includes t classes)

db.m5.large

2 vCPUs 8 GiB RAM EBS: 3500 Mbps

▼

☐ Include previous generation classes

Next, configure the storage options for your Amazon RDS database. There are two storage options in Amazon RDS: general purpose and provisioned IOPS. With general-purpose storage, you receive 3 IOPS (I/O operations per second) per GiB of storage allocated (with a minimum of 100 IOPS). Thus, 100 GiB of storage would have 300 IOPS. Additionally, you receive burst capacity up to 3,000 IOPS.

With provisioned storage, you provision IOPS separately from your storage capacity. This enables you to fine-tune your storage and operations settings to fit your needs. Additionally, you can enable [storage autoscaling](#). With storage autoscaling, Amazon RDS automatically increases your storage capacity as your database is close to running out of free disk space.

Storage

Storage type [Info](#)

General Purpose (SSD) ▼

Allocated storage

100

GiB

(Minimum: 20 GiB, Maximum: 65536 GiB) Higher allocated storage [may improve](#) IOPS performance.

Storage autoscaling [Info](#)

Provides dynamic scaling support for your database's storage based on your application's needs.

☒ **Enable storage autoscaling**
Enabling this feature will allow the storage to increase once the specified threshold is exceeded.

Maximum storage threshold [Info](#)

Charges will apply when your database autoscales to the specified threshold

1000

GiB

Minimum: 101 GiB, Maximum: 65536 GiB

Next, decide whether to create a standby instance. A standby instance is a replica of your data that is available in the event of failure. It is located in a different Availability Zone in the same AWS Region as your primary Amazon RDS instance to limit the impact of infrastructure failures. If you are running a production database where uptime is essential, a standby instance is recommended.

Availability & durability

Multi-AZ deployment [Info](#)

☒ **Create a standby instance (recommended for production usage)**
Creates a standby in a different Availability Zone (AZ) to provide data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups.

☐ **Do not create a standby instance**

The next section in the Amazon RDS database creation wizard is about connectivity. You must specify the [Amazon Virtual Private Cloud, or Amazon VPC](#), in which your database will reside, as well as the network subnet and security groups for your database instance.

If you are migrating from a self-managed database instance on Amazon EC2, you can use the same Amazon VPC and security groups as your existing database.

If you are migrating from a database that is not hosted on Amazon Web Services (AWS) but your application is hosted on AWS, choose the same Amazon VPC that is used for your application. Then create a new security group for your database instance.

Connectivity

Virtual Private Cloud (VPC) [Info](#)
VPC that defines the virtual networking environment for this DB instance.

Default VPC (vpc-0...)

Only VPCs with a corresponding DB subnet group are listed.

After a database is created, you can't change the VPC selection.

Additional connectivity configuration

Subnet group [Info](#)
DB subnet group that defines which subnets and IP ranges the DB instance can use in the VPC you selected.

default

Publicly accessible [Info](#)

☐ Yes
Amazon EC2 instances and devices outside the VPC can connect to your database. Choose one or more VPC security groups that specify which EC2 instances and devices inside the VPC can connect to the database.

☒ No
RDS will not assign a public IP address to the database. Only Amazon EC2 instances and devices inside the VPC can connect to your database.

VPC security group
Choose one or more RDS security groups to allow access to your database. Ensure that the security group rules allow incoming traffic from EC2 instances and devices outside your VPC. (Security groups are required for publicly accessible databases.)

☐ Choose existing
Choose existing VPC security groups

☒ Create new
Create new VPC security group

New VPC security group name

mysql-database

Database port [Info](#)
TCP/IP port the database will use for application connections.

3306

training and certification

Page 7

Finally, choose the database authentication methods to allow in your database. MySQL databases traditionally allow for username and password authentication. With MySQL on Amazon RDS, you also can choose to allow for authentication by using [AWS Identity and Access Management](#), or IAM. This integrates easily with your application compute and removes the need for credential rotation. For certain MySQL versions, you also can [allow Kerberos authentication](#) to connect with [AWS Directory Service for Microsoft Active Directory](#).

It is recommended that you allow password and IAM database authentication. If you only want to start with password authentication, you can add IAM database authentication later, but it will result in some downtime for your Amazon RDS database instance.

Database authentication

Database authentication options [Info](#)

- ☐ Password authentication
Authenticates using database passwords.
- ☒ Password and IAM database authentication
Authenticates using the database password and user credentials through AWS IAM users and roles.
- ☐ Password and Kerberos authentication (not available for this version)
Choose a directory in which you want to allow authorized users to authenticate with this DB instance using Kerberos Authentication.

There are some additional configuration options you can configure, including settings for backups, monitoring, maintenance, and automated upgrades. The default settings work for most situations, but you should review them to ensure they work for your needs.

► **Additional configuration**

Database options, encryption enabled, backup enabled, backtrack disabled, Performance Insights enabled, Enhanced Monitoring enabled, maintenance, CloudWatch Logs, delete protection enabled


The end of the database creation wizard shows you the estimated monthly costs for your database instance. Choose **Create database** to create your database instance.

Estimated monthly costs

DB instance	124.83 USD
Storage	11.50 USD
Multi-AZ standby instance	124.83 USD
Total	261.16 USD

This billing estimate is based on on-demand usage as described in [Amazon RDS Pricing](#). Estimate does not include costs for backup storage, I/Os (if applicable), or data transfer.

Estimate your monthly costs for the DB Instance using the [AWS Simple Monthly Calculator](#).

 You are responsible for ensuring that you have all of the necessary rights for any third-party products or services that you use with AWS services.

Cancel

Create database

As Amazon RDS is provisioning your infrastructure and initializing your database, the **Status** of your database is *Creating*.

Creating database my-database.
Your database might take a few minutes to launch.

View credential details

RDS > Databases

Databases

Group resources

Modify

Actions

Restore from S3

Create database

Filter databases

DB identifier	Role	Engine	Region & AZ	Size	Status	CPU	Current acti
my-database	Instance	MySQL Community	us-east-1c	db.m5.large	Creating		0 Se

When your database is ready to use, its **Status** is *Available*.

RDS > Databases

Databases

Group resources

Modify

Actions

Restore from S3

Create database

Filter databases

DB identifier	Role	Engine	Region & AZ	Size	Status	CPU	Cu
my-database	Instance	MySQL Community	us-east-1c	db.m5.large	Available	1.00%	

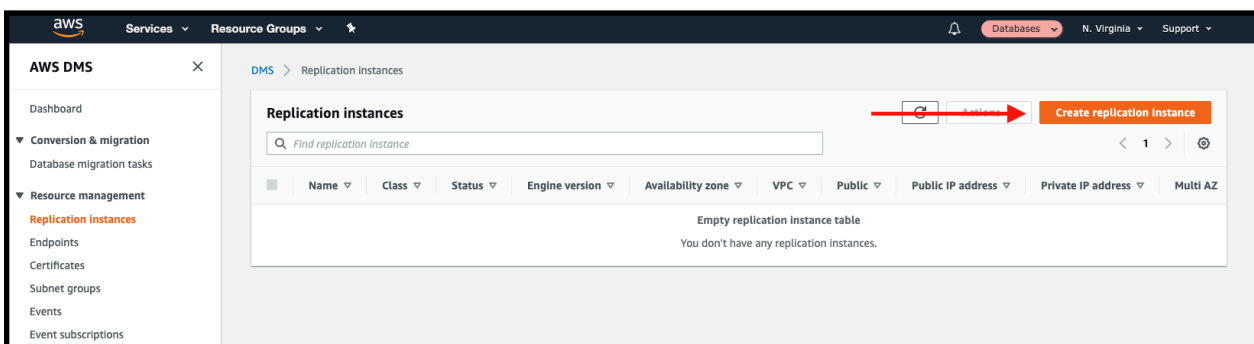
In this step, you created a fully managed, production-ready MySQL database instance in Amazon RDS. In the next step, you will create a replication instance in AWS DMS.

2. Create a replication instance in AWS DMS

In this step, you will create a replication instance in AWS DMS.

AWS DMS is a service that copies data and provides ongoing replication from an existing database to a fully managed AWS database. A replication instance is an Amazon EC2 instance that can host replication tasks in AWS DMS. In the next step, you will set up endpoints.

To create a replication instance, go to the [Replication instances section](#) of the AWS DMS console. Choose **Create replication instance** to begin the replication instance creation wizard.



In the **Replication instance configuration** section, give your replication instance a name and description. Then choose your instance class. The instance class you use depends on the size of your existing database and the amount of data flowing through it.

Then choose an engine version for AWS DMS. Finally, choose the amount of allocated storage for your replication instance.

Replication instance configuration

Name
The name must be unique among all of your replication instances in the current AWS region.

myReplicationInstance

Replication instance name must not start with a numeric value

Description

Replication instance for moving to fully-managed MySQL in RDS

The description must only have unicode letters, digits, whitespace, or one of these symbols: _:/=+-@. 1000 maximum character.

Instance class
Choose an appropriate instance class for your replication needs. Each instance class provides differing levels of compute, network and memory capacity.

dms.t2.medium ▼

Billing is based on [DMS pricing](#).

Engine version
Choose an AWS DMS version to run on your replication instance.

3.3.1 ▼

Allocated storage (GiB)
Choose the amount of storage space you want for your replication instance. AWS DMS uses this storage for log files and cached transactions while replication tasks are in progress.

50

As you continue in the **Replication instance configuration** section, choose a VPC for your replication instance. Choose the same VPC in which you provisioned your Amazon RDS database to ease network access for the replication instance.

You can choose to have a Multi-AZ setup for your replication instance for redundancy. If you are using AWS DMS to keep two databases in sync over a long period of time, you might want to use a Multi-AZ setup. If you are performing a one-time migration of your data from an existing database to a fully managed database in Amazon RDS, you likely don't need a Multi-AZ setup.

Finally, choose whether your replication instance should be publicly accessible. If your existing database is in the same VPC as your new database and your replication instance, you don't need your replication instance to be publicly accessible. If your existing database is not in the same VPC, you should use AWS Direct Connect or a VPN to [allow connection](#) from your VPC to your existing data center. You should avoid making your replication instance publicly accessible in most cases to avoid potential security issues.

VPC
Choose an Amazon Virtual Private Cloud (VPC) where your replication instance should run.

vpc-0-.....e ▼

☐ **Multi AZ**
If you choose this option, AWS DMS will perform a multi-AZ deployment, with a primary instance in one availability zone (AZ) and a standby instance in another AZ. This configuration provides a highly available, fault-tolerant replication environment.

Billing is based on [DMS pricing](#).

☒ **Publicly accessible**
If you choose this option, AWS DMS will assign a public IP address to your replication instance, and you'll be able to connect to databases outside of your Amazon VPC.

Next, open the **Advanced security and network configuration** section. For the **VPC security group(s)** configuration, choose the same security group that you attached to your Amazon RDS database. This allows your replication instance to access your Amazon RDS database.

▼ **Advanced security and network configuration**

Replication subnet group
Choose a subnet group for your replication instance. The subnet group defines the IP ranges and subnets that your replication instance can use within the Amazon VPC you've chosen.

default-vpc-0-.....e ▼

Availability zone
Choose an availability zone (AZ) where you want your replication instance to run. The default is "No preference", meaning that AWS DMS will determine which AZ to use.

No Preference ▼

VPC security group(s)
Choose one or more security groups for your replication instances. The security group(s) specify inbound and outbound rules to control network access to your replication instance.

Use default ▼

mysql-database X ←

KMS master key [Info](#)
(Default) aws/dms ▼

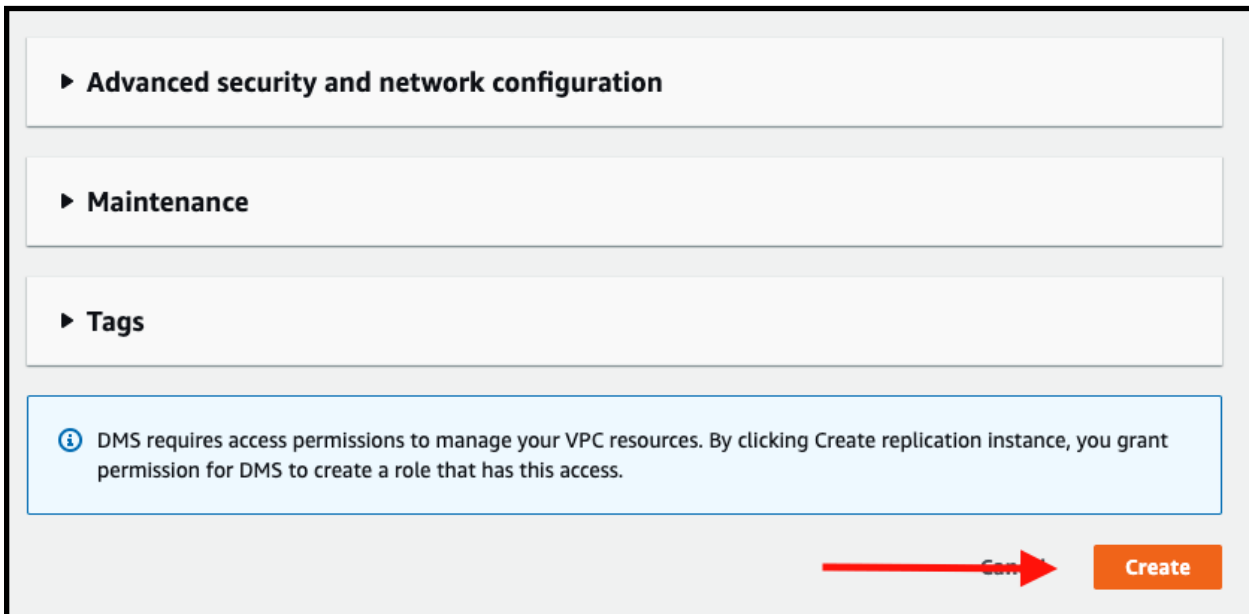
Account

Description

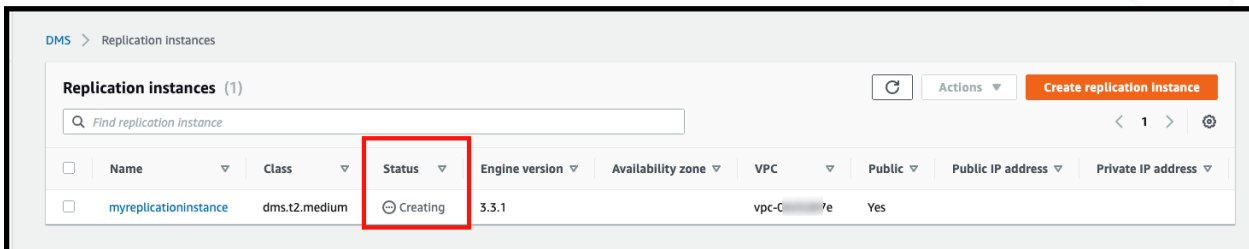
Key ARN

You can also edit the Maintenance and Tags settings.

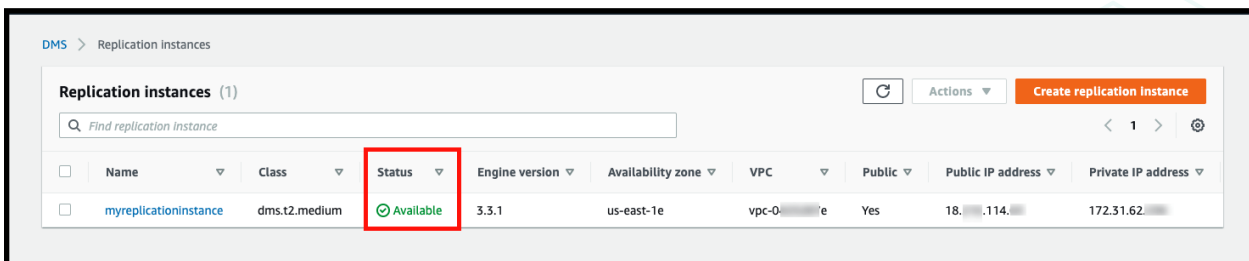
When you're ready, choose **Create** to create your replication instance in AWS DMS.



While AWS provisions and initializes your instances, the **Status** is *Creating*.

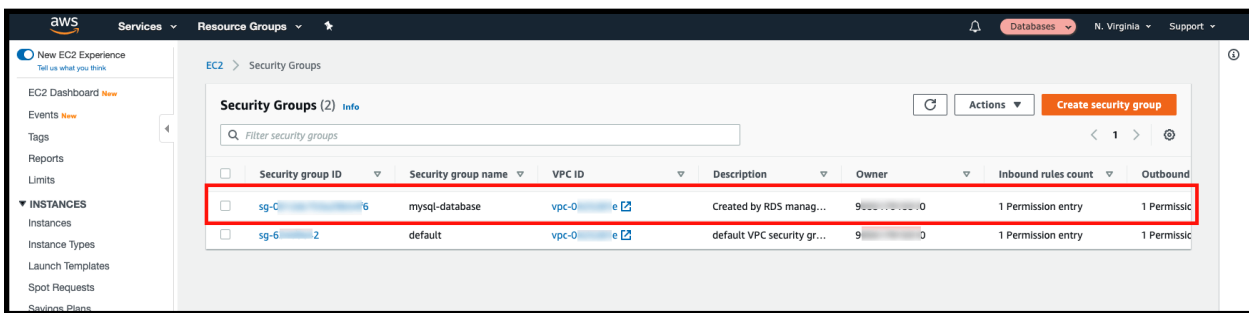


When your replication instance is ready to go, its **Status** is *Available*.

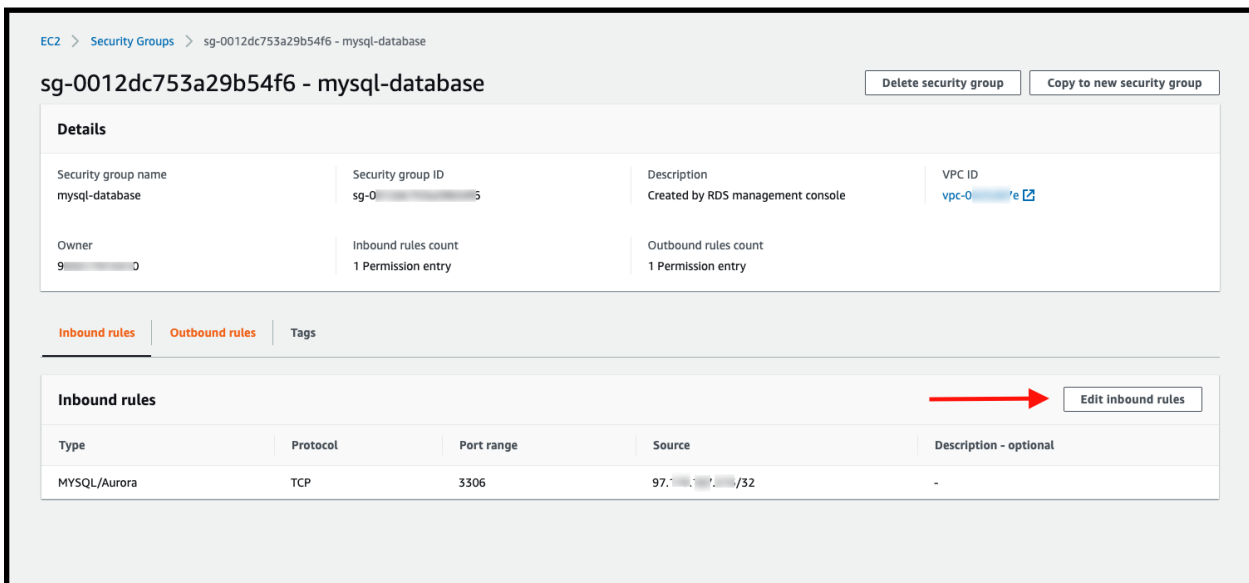


While you are waiting for your replication instance to be available, go to the [Security Groups section](#) in the Amazon EC2 console. You need to add a rule to your security group to allow your replication instance to access your database.

In the **Security Groups** section, find the security group you attached to your MySQL database instance and your replication instance, and choose it.



Choose to **Edit inbound rules** for your security group.



Your security group has an existing rule that allows for access to your MySQL instance from the IP address you used to create the database. Remove the existing IP address and enter the name of the security group used for your Amazon RDS database instance and replication instance.

Your screen should look like the following.

EC2 > Security Groups > sg-0012dc753a29b54f6 - mysql-database > Edit inbound rules

Edit inbound rules [Info](#)

Inbound rules control the incoming traffic that's allowed to reach the instance.

Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info	
MySQL/Aurora	TCP	3306	Custom		Delete

NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.

Choose **Save rules** to save the updated rules for your security group.

When your replication instance is available and you have updated the rules for your security group, you can move to the next step.

In this step, you created a replication instance in AWS DMS. The replication instance is used to host the replication tasks that migrate data from an existing database to a fully managed database in Amazon RDS. You also updated a security group to allow access from your replication instance to your MySQL database instance in Amazon RDS.

In the next step, you will create endpoints for your source and target databases in Amazon RDS.

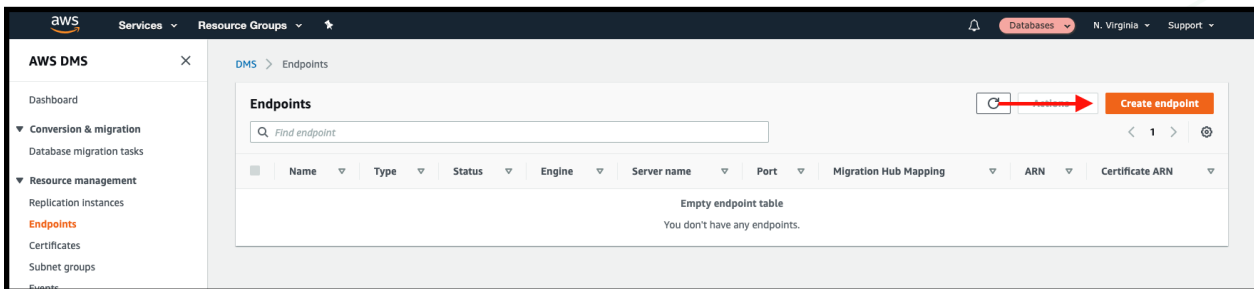
3. Create endpoints in AWS DMS

In this step, you will create source and target endpoints for a replication task in AWS DMS.

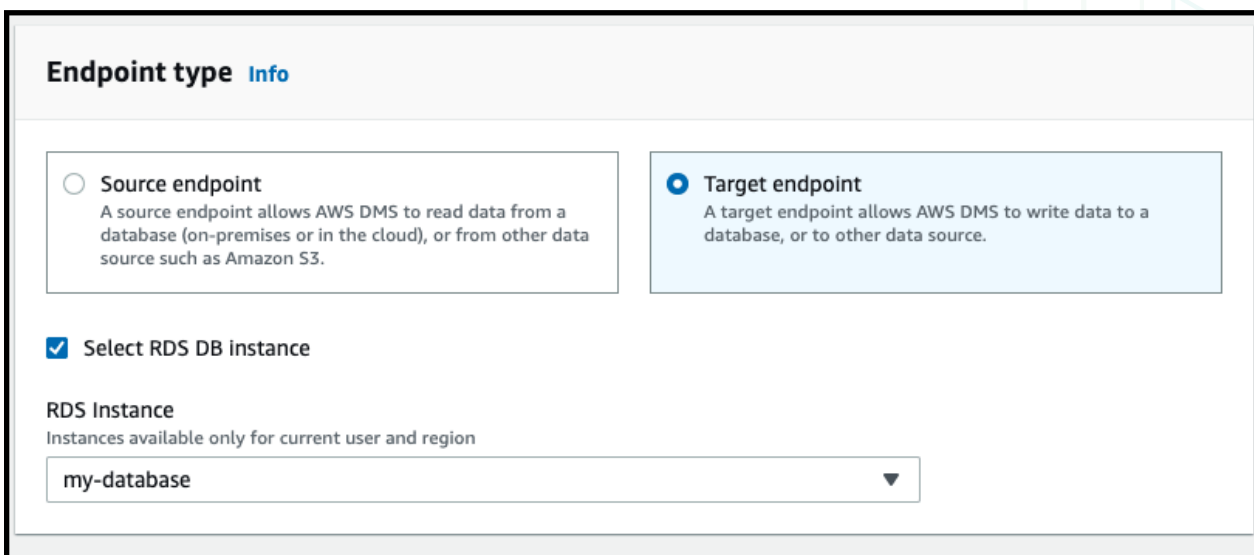
A replication task is a job that migrates data from one database to another by using AWS DMS. Before creating a replication task, you must register endpoints for your source and target databases. An endpoint describes the connection address, credentials, and other information required to connect to a database.

First, create the endpoint for your target database. This is the database you created in Amazon RDS.

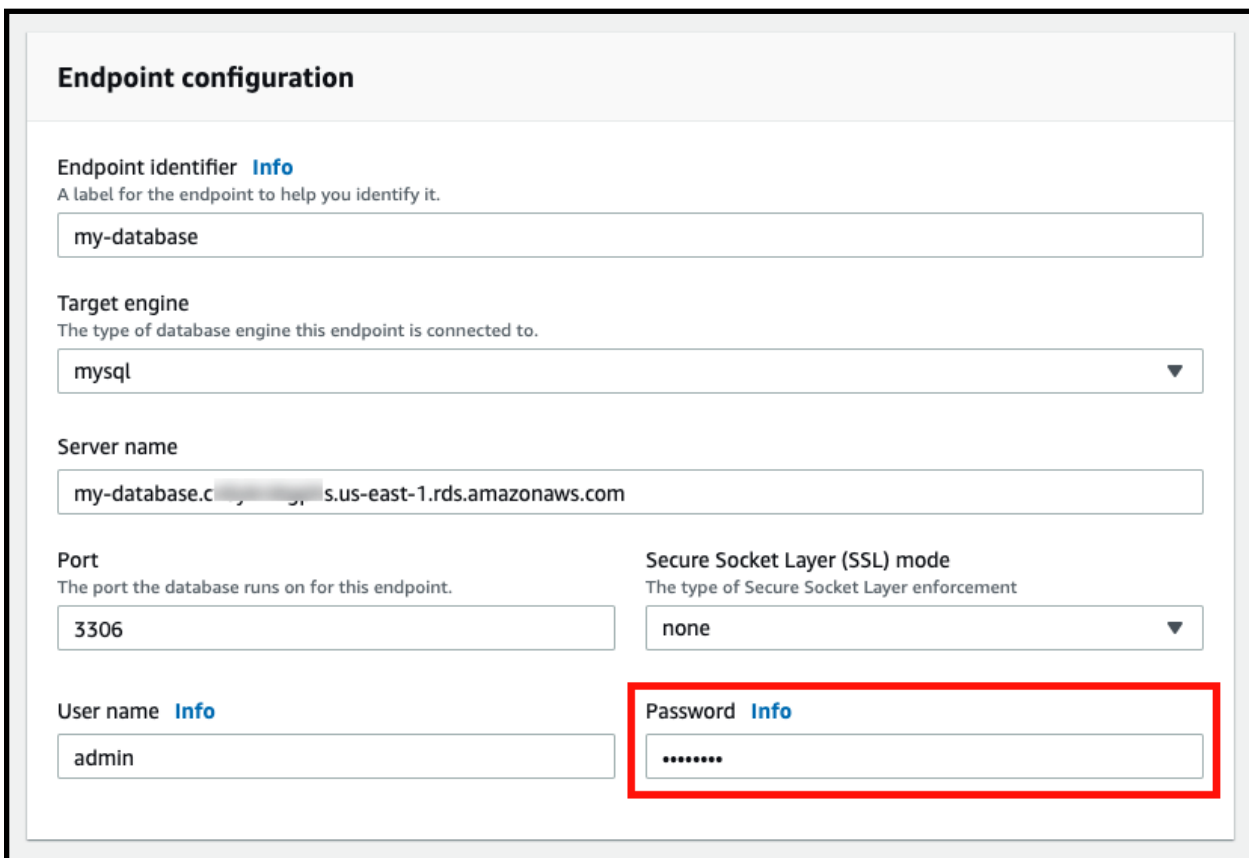
Navigate to the [Endpoints section](#) of the AWS DMS console. Choose **Create endpoint** to create a new endpoint.



In the endpoint creation wizard, choose to create a **Target endpoint**. Select **Select RDS DB instance**, and choose your newly created Amazon RDS database in the dropdown.



This fills in most of the **Endpoint configuration** details for you. You still need to enter your password near the bottom.



Endpoint configuration

Endpoint identifier [Info](#)
A label for the endpoint to help you identify it.

Target engine
The type of database engine this endpoint is connected to.

Server name

Port
The port the database runs on for this endpoint.

Secure Socket Layer (SSL) mode
The type of Secure Socket Layer enforcement

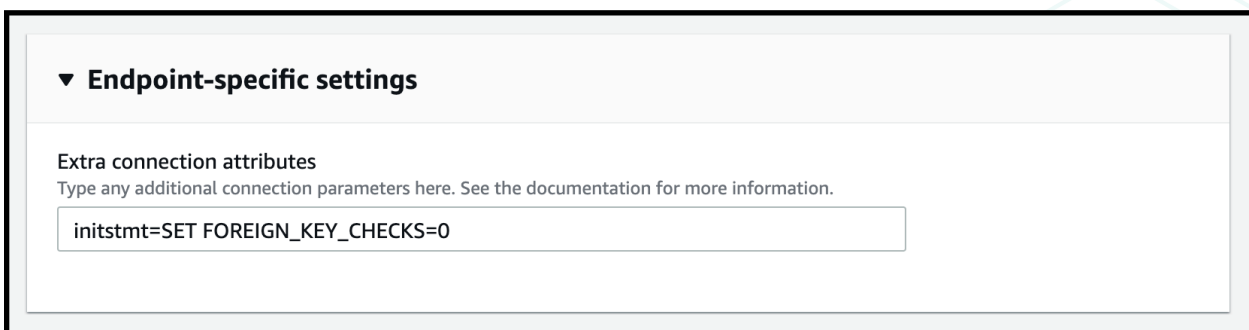
User name [Info](#)

Password [Info](#)

When loading data into a MySQL database by using AWS DMS, you need to disable foreign key checks. You can do this by entering

`initstmt=SET FOREIGN_KEY_CHECKS=0`

in the **Extra connection attributes** box.



▼ **Endpoint-specific settings**

Extra connection attributes
Type any additional connection parameters here. See the documentation for more information.

Before you save your endpoint, you should test the connection to ensure it was configured correctly. Open the **Test endpoint connection** section to test your connection.

Choose the replication instance you want to use, and then choose **Run test**. After a few seconds, you should see a **Status** of *successful*. This indicates that you configured your security group and endpoint correctly. Choose **Create endpoint** to save your endpoint.

► Endpoint-specific settings

► KMS master key

► Tags

▼ Test endpoint connection (optional)

Test your endpoint connection by selecting a replication instance within your desired VPC. After clicking "Run test", an endpoint will be created with the details provided and attempt to connect to the instance. If the connection fails, you can edit and test it again. Endpoints that aren't saved will be deleted.

VPC
vpc-01234567e ▼

Replication instance
A replication instance performs the database migration
myreplicationinstance ▼

Run test ←

After clicking "Run test", an endpoint will be created with the details provided and attempt to connect to the instance. If the connection fails, you can edit and test it again. Endpoints that aren't saved will be deleted.

Endpoint identifier	Replication instance	Status	Message
my-database	myreplicationinstance	successful	

→ **Create endpoint**

Follow these same steps again to create an endpoint for your source database. Unlike the target database, you need to fill out the connection endpoint, port, and credentials yourself.

You also need to ensure that your replication instance has network access to your source database. If your source database is hosted on Amazon EC2, allow traffic from your replication instance security group into the source database security group. If your source database is not hosted on Amazon EC2, you need to configure the network settings according to the location of your source database.

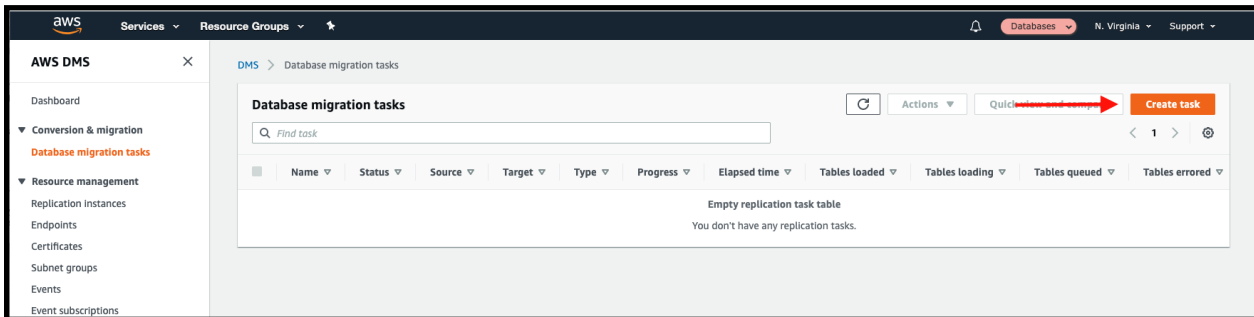
Before moving to the next step, you should have two endpoints configured: one for your source database and one for your target database. Make sure that you have tested both endpoints and can successfully connect to both databases. Then move to the next step.

In this step, you created endpoints to connect to your databases. In the next step, you will use those endpoints to create a replication task that copies data from your source database to your target database.

4. Create a replication task in AWS DMS

In this step, you will create a replication task in AWS DMS. A replication task is responsible for migrating data from a source database to a target database. In your case, you are moving data from an existing database to your newly created database in Amazon RDS.

To get started, navigate to the [Database migration tasks section](#) of the AWS DMS console. Choose **Create task** to create a new replication task.



In the **Task configuration** section, set up the parameters of your replication task. Give your task a name, and choose the replication instance you created in an earlier step. Then choose the source endpoint for your existing database and your target endpoint for your fully managed database in Amazon RDS.

You need to choose a migration type. There are two migration types:

1. **Migrate existing data**, which performs a one-time process to copy data from your source database to your target database
2. **Replicate data changes**, which copies all ongoing operations from your source database to your target database

For the migration type, choose **Migrate existing data and replicate ongoing changes**. Note that this requires you to have the [MySQL binary log](#) enabled on your source database.

Task configuration

Task identifier

mysql-migration

Replication instance

myreplicationinstance - vpc-0455281e

Source database endpoint


self-managed-database


Target database endpoint


my-database

Migration type [Info](#)

Migrate existing data and replicate ongoing changes

 When switching database engines, the AWS Schema Conversion Tool can automatically convert your database schema and code to the engine of your choice. Click here to find out more. [Learn more](#)



 Your source database is MySQL. Replicating ongoing changes requires the MySQL binary log to be enabled and set to row.
Please ensure your binary logs are retained on the server for a sufficient amount of time, (24 hours is usually enough.) To set your binary log retention time on RDS instances you can use the following command: call mysql.rds_set_configuration("binlog retention hours", 24);

☒ Start task on create

In the **Table mappings** section, select the tables you want to copy over. Enter the names of the schemas and tables you want to copy. You can use % as a wildcard character to copy multiple tables or schemas.

▼ Table mappings

Editing mode

☒ Guided UI
Set up your table mapping rules using a step-by-step guided interface.

☐ JSON editor [Learn more](#)
Enter your table mapping rules directly, in JSON format.

Specify at least one selection rule with an include action. After you do this, you can add one or more transformation rules.

▼ Selection rules

Choose the schema and/or tables you want to include with, or exclude from, your migration task. [Info](#)

Add new selection rule

▼ where **schema name** is like '%' and **table name** is like '%', include

Schema

Enter a schema

Schema name

Use the % character as a wildcard

public

Table name

Use the % character as a wildcard

%

Action

Choose "Include" to migrate your selected objects, or "Exclude" to ignore them during the migration.

Include

Source filters [Info](#)

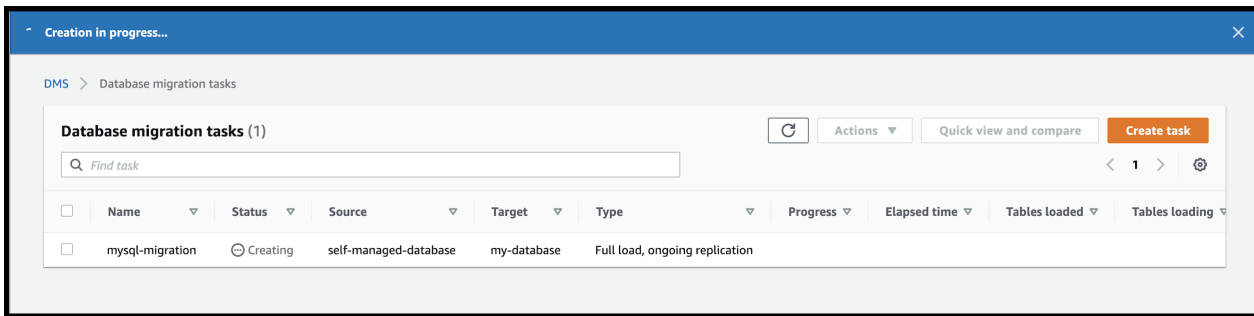
Add column filter

When you are ready, choose **Create task** to start your replication task.

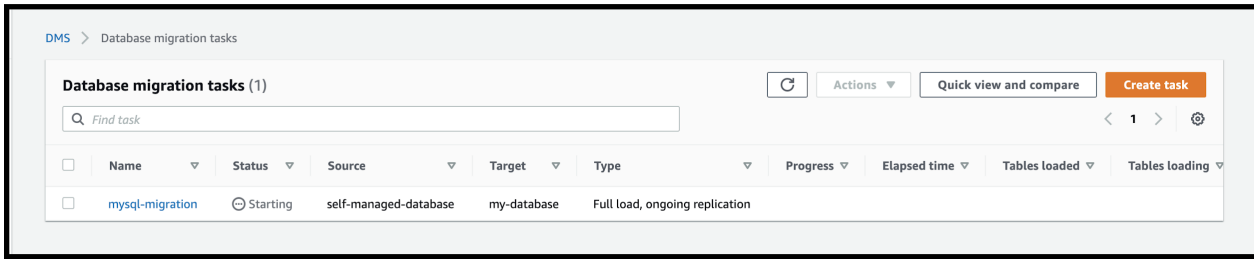
After you create the task, it is shown in the **Database migration tasks** section with a **Status** of *Creating*.

aws training and
certification

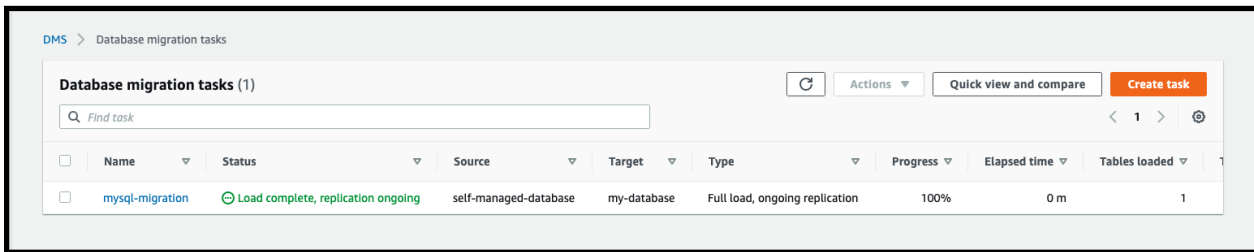
Page 22



After the task is initialized, its **Status** is *Starting*.



After the migration of existing data is complete, it shows a **Status** of *Load complete, replication ongoing*. Any updates to your source database at this point are copied to your target database.



The screenshot shows the AWS DMS console interface for 'Database migration tasks'. At the top, there's a breadcrumb 'DMS > Database migration tasks'. Below it, a section titled 'Database migration tasks (1)' contains a search bar labeled 'Find task' and buttons for 'Actions', 'Quick view and compare', and 'Create task'. A table lists the tasks with columns: Name, Status, Source, Target, Type, Progress, Elapsed time, and Tables loaded. One task is listed: 'mysql-migration' with a status of 'Load complete, replication ongoing', source 'self-managed-database', target 'my-database', type 'Full load, ongoing replication', progress '100%', elapsed time '0 m', and '1' table loaded.

	Name	Status	Source	Target	Type	Progress	Elapsed time	Tables loaded
<input type="checkbox"/>	mysql-migration	Load complete, replication ongoing	self-managed-database	my-database	Full load, ongoing replication	100%	0 m	1

In this step, you created a replication task in AWS DMS to migrate your existing data and sync ongoing changes from your previous database to your new database in Amazon RDS.

In the next step, you will learn about next steps and cleaning up the resources you created.

5. Complete the migration and clean up resources

If you followed all the steps in this lesson, you created a new, fully managed MySQL database in Amazon RDS and created a migration task to copy data from your source database to your new database. In this final step, you will learn the steps to complete your migration and clean up your AWS DMS resources.

When your initial migration is complete and all data is synced to your new database, you are ready to use your new database as your primary database.

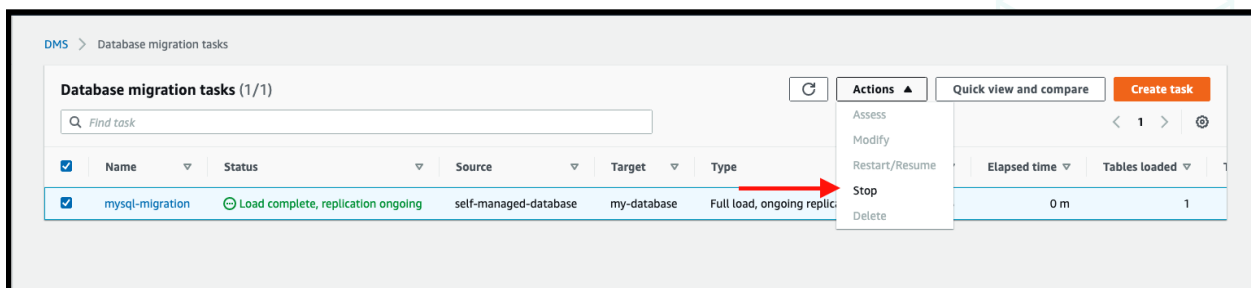
There are two different ways you can handle this:

1. If you feel confident in the correctness of the migration, you can change the database configuration in your application to use your new database. This helps ensure that all reads and writes go to your new database.
2. If you want to follow a more cautious approach, you can read from and write to both databases for a period of time. This enables you to compare the results from each database for correctness while still maintaining the correct data in your existing database.

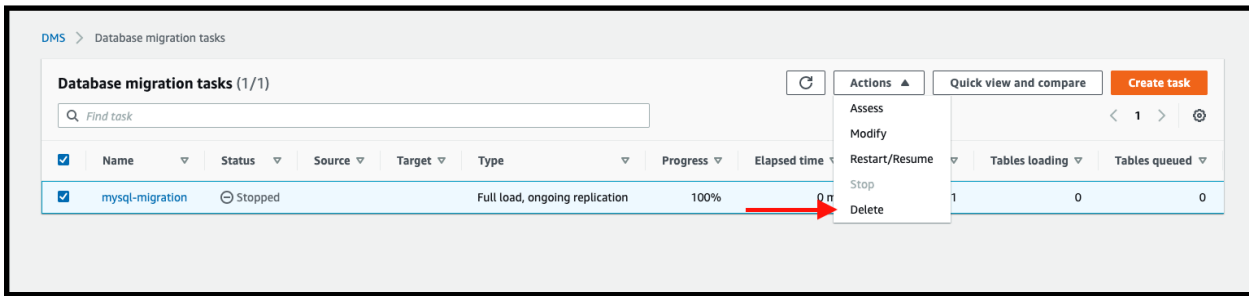
Whichever method you choose, you should thoroughly test your new database for correctness before making it your primary database.

After you have switched to using your primary database and are confident in the results, you might want to delete your AWS DMS infrastructure.

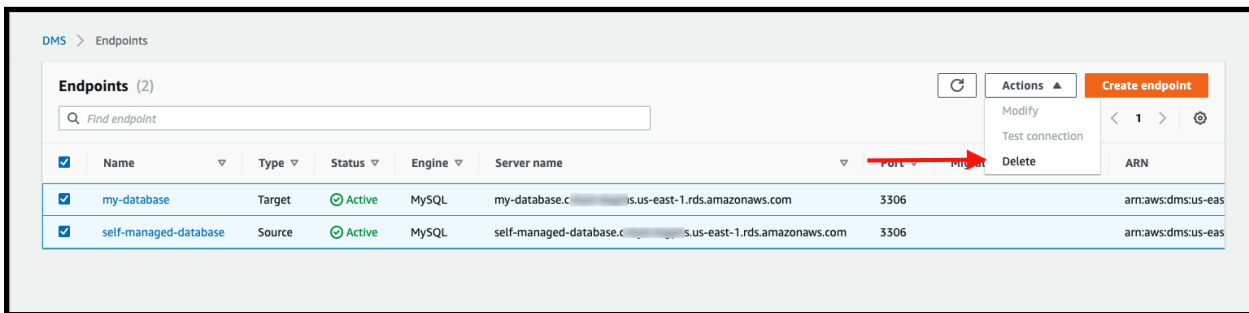
First, stop and delete the database migration task to replicate your data. Navigate to the [Database migration tasks](#) section of the AWS DMS console. Choose the task you want to remove, and then choose **Stop**.



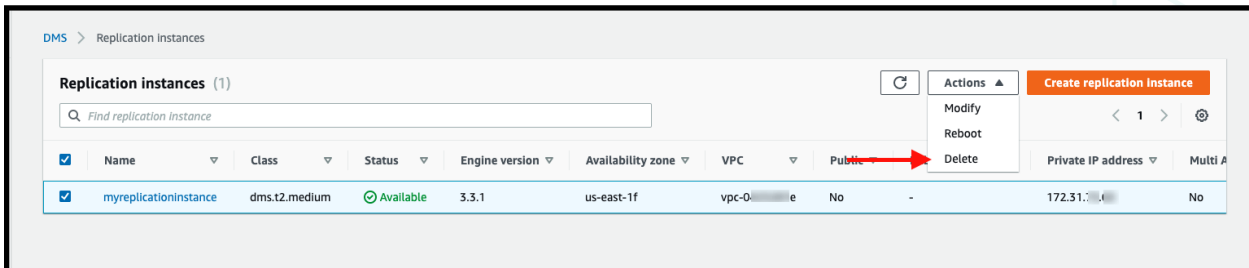
When the task has stopped, choose it again, and then choose **Delete**.



Next, navigate to the [Endpoints section](#) of the AWS DMS console. Choose your source endpoint and your target endpoint, and then choose **Delete**.



Then go to the [Replication instances section](#) of the AWS DMS console. If your replication instance is not being used for any other replication tasks, choose it, and then choose **Delete**.



Finally, you might want to terminate your source database because it is no longer being used. If your source database is running on Amazon EC2, you can terminate the Amazon EC2 instance. If your source database is running elsewhere, follow the proper procedures to terminate it.

In this step, you learned how to migrate your application to use your new database. You also saw how to clean up AWS DMS resources after you are done using them.

In this lesson, you migrated an existing MySQL database to a fully managed MySQL database in Amazon RDS by using AWS DMS. By using Amazon RDS, you can free your developers to focus on innovation that's core to your business. By using AWS DMS, you can automate the delicate task of migrating data to a new database.

