Migrate On-premises Databases to AWS Using AWS Database Migration Service (DMS)

**SPL-TF-200-MGADMS-1 - Version 1.0.0**

© 2024 Amazon Web Services, Inc. or its affiliates. All rights reserved. This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited. All trademarks are the property of their owners.

Note: Do not include any personal, identifying, or confidential information into the lab environment. Information entered may be visible to others.

Corrections, feedback, or other questions? Contact us at [*AWS Training and Certification*](https://support.aws.amazon.com/#/contacts/aws-training).

**Lab overview**

AnyCompany is migrating its current database to a cloud native database. You have been assigned the migration task, after researching different database options, you choose to use Amazon Aurora. The current architecture includes:

* An application that uses a MariaDB database.

In this lab, you use AWS Database Migration Service to migrate the current database to Amazon Aurora.

OBJECTIVES

By the end of this lab, you should be able to do the following:

* Create an Amazon Aurora database
* Migrate existing database to Aurora using AWS DMS
* Update DNS records to reflect the migration

TECHNICAL KNOWLEDGE PREREQUISITES

This lab requires:

* Access to a computer with Microsoft Windows, Mac OS X, or Linux (Ubuntu, SuSE, or Red Hat)
* A modern internet browser such as Chrome or Firefox

DURATION

This lab requires approximately *75* minutes to complete.

ICON KEY

Various icons are used throughout this lab to call attention to different types of instructions and notes. The following list explains the purpose for each icon:

* **Command:** A command that you must run.
* **Note:** A hint, tip, or important guidance.
* **Caution:** Information of special interest or importance (not important enough to cause problems with equipment or data if you miss it, but it could result in the need to repeat certain steps).
* **Note:** A hint, tip, or important guidance.
* **Task complete:** A conclusion or summary point in the lab.
* **Expected output:**

**Start lab**

1. To launch the lab, at the top of the page, choose **Start lab**.

 You must wait for the provisioned AWS services to be ready before you can continue.

1. To open the lab, choose **Open Console**.

You are automatically signed in to the AWS Management Console in a new web browser tab.

**Do not change the Region unless instructed.**

COMMON SIGN-IN ERRORS

**Error: You must first sign out**



If you see the message, **You must first log out before logging into a different AWS account:**

* Choose the **click here** link.
* Close your **Amazon Web Services Sign In** web browser tab and return to your initial lab page.
* Choose **Open Console** again.

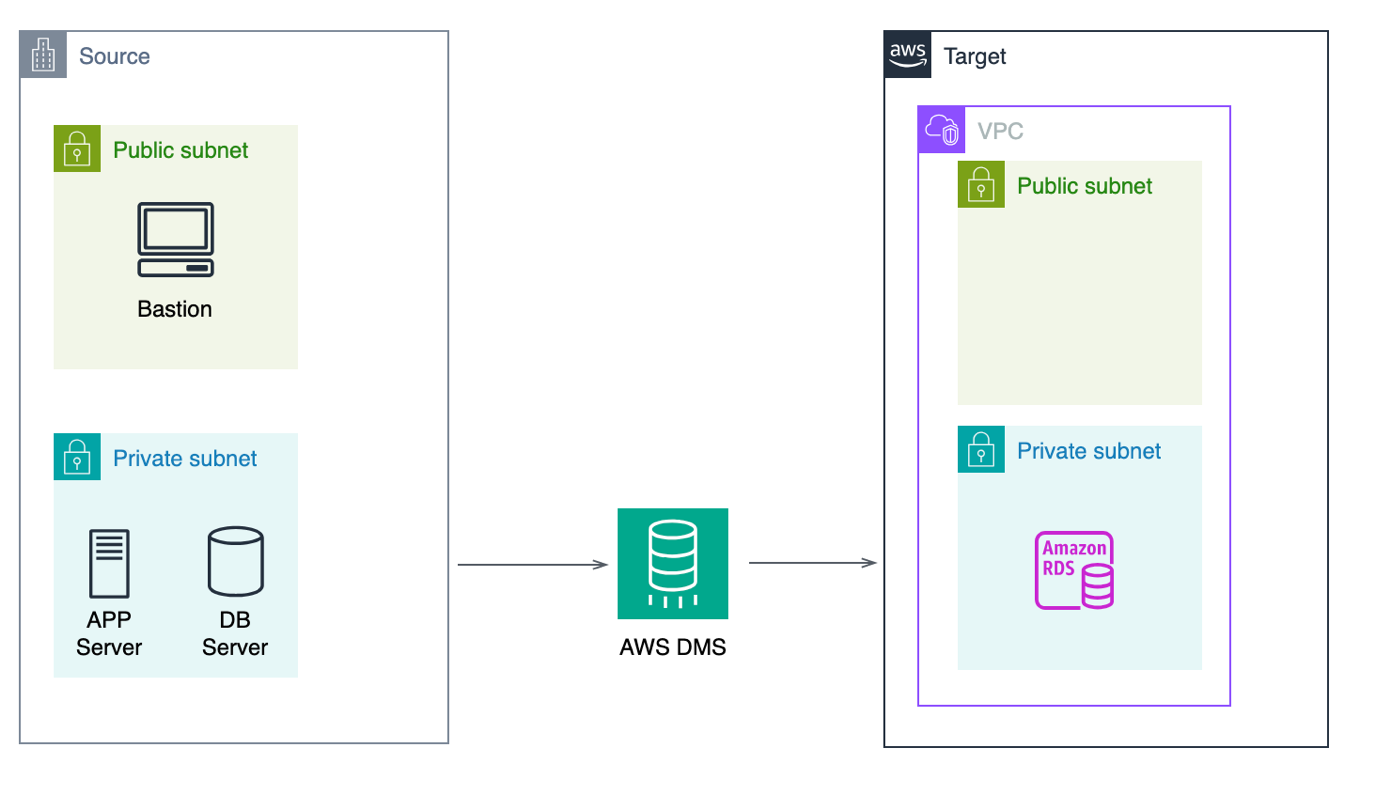
**Error: Choosing Start Lab has no effect**

In some cases, certain pop-up or script blocker web browser extensions might prevent the **Start Lab** button from working as intended. If you experience an issue starting the lab:

* Add the lab domain name to your pop-up or script blocker’s allow list or turn it off.
* Refresh the page and try again.

LAB ENVIRONMENT

The following diagram shows the basic architecture of the lab environment:



*Image description: The lab uses a VPC to simulate an Onpremise data center, this is the Source VPC. The current production environment running in the Source VPC is comprised of 1 application and 2 servers. The application is a Wordpress application that makes use of 1 database. The task is to migrate the database to Aurora located in the Target VPC.*

SERVICES USED IN THIS LAB

**AWS Database Migration Service**

AWS Database Migration Service helps you migrate databases to AWS quickly and securely. The source database remains fully operational during the migration, minimizing downtime to applications that rely on the database. The AWS Database Migration Service can migrate your data to and from most widely used commercial and open-source databases.

AWS Database Migration Service supports homogeneous migrations such as Oracle to Oracle, as well as heterogeneous migrations between different database platforms, such as Oracle or Microsoft SQL Server to Amazon Aurora. With AWS Database Migration Service, you can continuously replicate your data with high availability and consolidate databases into a petabyte-scale data warehouse by streaming data to Amazon Redshift and Amazon S3.

**AWS Aurora**

Amazon Aurora is a MySQL and PostgreSQL-compatible relational database built for the cloud, that combines the performance and availability of traditional enterprise databases with the simplicity and cost-effectiveness of open source databases.

Amazon Aurora is up to five times faster than standard MySQL databases and three times faster than standard PostgreSQL databases. It provides the security, availability, and reliability of commercial databases at 1/10th the cost. Amazon Aurora is fully managed by Amazon Relational Database Service (RDS), which automates time-consuming administration tasks like hardware provisioning, database setup, patching, and backups.

AWS SERVICES NOT USED IN THIS LAB

AWS service capabilities used in this lab are limited to what the lab requires. Expect errors when accessing other services or performing actions beyond those provided in this lab guide.

**Task 1: Create Aurora target database**

AnyCompany wants to migrate to a cloud native database. Your research leads you to choose Amazon Aurora. In this task, you create a new Aurora database that is used as the target database.

**Task 1.1: Create Aurora database**

Create an Aurora database using Amazon Relational Database Service.

1. At the top of the AWS Management Console, in the search bar, search for and choose **RDS**.
2. In the navigation pane at the left of the page, choose Databases, then choose **Create Database**
3. In the **Choose a database creation method** section, select **Standard create**.
4. In the **Engine options** section, configure the following:
   * For **Engine type**, select **Aurora (MySQL Compatible)**.
   * For **Engine version**, select **Aurora (MySQL 5.7) 2.11.2.**
5. In the **Templates** section, select **Dev/Test**.

**Note:** We are using Dev/Test selection on this exercise because there are no active users running the application. For a production environment prefer the Production template.

1. In the **Settings** section, configure the following:
   * For **DB cluster identifier** enter

MID-Wordpress

* + For **Master username** enter

admin

* + For **Master password** enter

AWSmid23

1. In the **Instance configuration** section:
   * For **DB instance class**, select **Burstable classes**.
   * For **instance type**, select **db.t3.medium**.
2. In the **Availability & durability** section, for **Multi-AZ deployment**, select **Don’t create an Aurora Replica**.
3. In the **Connectivity** section:
   * For **Virtual private cloud (VPC)**, select **Target**.
   * For **Public access**, select **No**.
   * For **VPC security group**, select **Choose existing**.
   * For **Existing VPC security groups**:
     + Keep the default security group.
     + Select **xxxxx-TargetSecurityGroup-xxxxx**.
   * For **Availability Zone**, select the **Availability Zone** ending in “a”.
4. In the **Database authentication** section, select **Password authentication**.
5. In the **Monitoring** section, deselect **Enable Enhanced monitoring**.
6. Expand the main  **Additional configuration** section.
   * In the **Database options** section:
     + For **Initial database name**, enter

wordpressdb

* + - In the **Encryption** section, deselect **Enable encryption**.

1. Choose **Create database**
2. On the **Suggested add-ons for mid-wordpress** pop-up window, choose **Close**

**Caution:** The database creation step can take a few minutes to complete. Before continuing to the next steps ensure both Databases status is Available.

You may occasionally choose refresh  - to see the status update.

1. Choose the **wordpressdb Writer** link.
2. In the **mid-wordpress-instance-1** database page, choose the **Connectivity & security** tab.
3. Copy the **Endpoint** value for the **Writer** instance to a text editor for use in a future step.

**Task 1.2: Create database user**

Create a database user for the new target database

1. At the top of the AWS Management Console, in the search bar, search for and choose **EC2**.
2. In the left navigation pane, under **Instances**, choose **Instances**.

**Note:** If the menu is collapsed, choose the menu  icon.

1. Select **MID-Wordpress-DB**.
2. Choose **Connect** from the navigation bar.
3. With **Session Manager** tab selected, choose **Connect**.

* **Note:** Session Manager is a fully managed AWS Systems Manager capability. With Session Manager, you can manage your Amazon Elastic Compute Cloud (Amazon EC2) instances, edge devices, on-premises servers, and virtual machines (VMs). You can use either an interactive one-click browser-based shell or the AWS Command Line Interface (AWS CLI). Session Manager provides secure and auditable node management without the need to open inbound ports, maintain bastion hosts, or manage SSH keys. Session Manager also allows you to comply with corporate policies that require controlled access to managed nodes, strict security practices, and fully auditable logs with node access details, while providing end users with simple one-click cross-platform access to your managed nodes.

1. **Command:** On the **MID-Wordpress-DB** instance, run the following command.

* Replace the **ENDPOINT** placeholder value with the database Writer **Endpoint** value you copied earlier, and run the command.

mysql -u admin -h ENDPOINT -p

1. Enter password

AWSmid23

1. **Command:** Once connected to the mysql console, run the following commands to create the user wordpress:

CREATE USER 'wordpress'@'%' IDENTIFIED BY 'AWSmid23';

GRANT ALL ON wordpressdb.\* TO 'wordpress'@'%';

FLUSH PRIVILEGES;

QUIT

**Task complete:** You have successfully created a Amazon Aurora DB.

**Task 2: Configure Database Migration Services (DMS)**

To be able to launch a DMS Replication instance, it is necessary to specify what subnet group in the VPC the Replication instance was created. A subnet is a range of IP addresses in your VPC in a given Availability Zone. These subnets can be distributed among the Availability Zones for the AWS Region where your VPC is located. DMS Replication instance requires at least two Availability Zones. A replication instance performs the actual data migration between source and target endpoints. Your instance needs enough storage and processing power to perform the tasks that migrate data from your source database to your target database. How large this replication instance should be depends on the amount of data to migrate and the tasks your instance needs to do.

**Task 2.1: Configure WordPress (MySQL) replication subnet group**

Create a replication subnet group.

1. Navigate to the AWS console browser tab with EC2 service.
2. At the top of the AWS Management Console, in the search bar, search for and choose **DMS**.
3. In the left navigation pane, choose **Subnet groups**
4. Choose **Create subnet group**.
5. In the **Subnet group configuration** section:
   * For **Name** enter

WP-SubnetGroup

* + For **Description** enter

Migration Immersion Day - WordPress Subnet Group

* + For **VPC** select **Source**

1. In the **Add subnets** section:
   * Select **xxxx-SourcePrivate** and **xxxx-SourcePrivateDB**
2. Choose **Create subnet group**

**Task 2.2 Create replication instance**

Create a replication instance that has sufficient storage and processing power to perform the tasks you assign and migrate data from your source database to the target database. The required size of this instance varies depending on the amount of data you need to migrate and the tasks that you need the instance to perform.

1. In the left navigation pane, under **Migrate data** choose **Replication instances**.
2. Choose **Create Replication Instance**
3. In the **Settings** section:
   * For **Name** enter

MID-REPINST-WP

* + For **Description - optional** enter

Migration Immersion Day - Rep Inst WordPress

1. In the **Instance configuration** section:
   * For **Instance class** select **dms.t3.medium**
   * For **Engine Version** select **3.4.7**
   * For **High Availability** select **Dev or Test workload (Single-AZ)**
2. In the **Connectivity and security** section:
   * Fort **Virtual private cloud(VPC) for IPv4** select **Source**
   * For **Replication subnet group** select **WP-SubnetGroup**
   * Uncheck **Publicly accessible**
   * Expand **Advanced settings**
     + For **Availability zone** select the one that ends with **a**
     + For **VPC Security groups** select **default** and **xxxx-SourceSecurityGroup-xxxx**

All other settings can be used as the default values.

1. Choose **Create Replication Instance**

 The Replication instance creation can take several minutes to complete. Wait for the Replication instance status to update to Available before continuing to the next steps.

You may occasionally choose refresh  - to see the status update.

**Task 2.3: Create source endpoint.**

Create the endpoint to connect to the source database.

First, you need to get the EC2 private IP address where WordPress database is running.

1. At the top of the AWS Management Console, in the search bar, search for and choose **EC2**
2. In the left navigation pane, under **Instances**, choose **Instances**.

**Note:** If the menu is collapsed, choose the menu  icon.

1. Select **MID-Wordpress-DB**.
2. With **Details** tab selected, copy the **Private IPv4 addressess** value to a text editor.
3. At the top of the AWS Management Console, in the search bar, search for and choose **DMS**.
4. In the left navigation pane, under **Migrate data** choose **Endpoints**.
5. Choose **Create endpoint**
6. In the **Endpoint type** section:
   * Select **Source endpoint**
7. In **Endpoint configuration** section:
   * For **Endpoint identifier** enter

SourceWordpress

* + For **Source engine** select **Mariadb**
  + For **Access to endpoint database**, select **Provide Access Information Manually**
  + For **Server name** enter **the IP address that you copied earlier.**
  + For **Port** enter

3306

* + For **User name** enter

wordpress

* + For **Password** enter

AWSmid23

1. Expand **Test endpoint connection**:
   * For **VPC** select **Source**
   * For **Replication Instance** select **mid-repinst-wp**
   * Choose **Run test**. Wait for the test result. In case of error, review the parameters again.
2. After the successful status, choose **Create endpoint**

**Task 2.4: Create target endpoint**

Create the endpoint to connect to the target database.

1. Choose **Create endpoint**
2. In the **Endpoint type** section:
   * Select **Target endpoint**
3. Select **Select RDS DB instance** and for **RDS Instance** select **mid-wordpress-instance-1**.
4. In **Endpoint configuration** section:
   * Endpoint identifier:

TargetWordpress

* + For **Access to endpoint database**, select **Provide Access Information Manually**
  + For **Password** enter

AWSmid23

1. All other settings can be used as the default values.
2. Expand **Test endpoint connection**:
   * For **VPC** select **Target**
   * For **Replication Instance** select **mid-repinst-wp**
   * Choose **Run test**. Wait for the test result. In case of error, review the parameters again.
3. After the successful status, choose **Create endpoint**

**Task 2.5: Create migration task**

Create a migration task. An AWS Database Migration Service (AWS DMS) task is where all the work happens. You specify what tables (or views) and schemas to use for your migration and any special processing, such as logging requirements, control table data, and error handling.

1. In the left navigation pane, under **Migrate data** choose **Database migration tasks**.
2. Choose **Create task**
3. In the **Task configuration** section:
   * For **Task identifier** enter

WordPress-MySQL-to-Aurora

* + For **Replication instance** select **mid-repinst-wp**
  + For **Source database endpoint** select **sourcewordpress**
  + For **Target database endpoint** select **targetwordpress**
  + For **Migration Type** select **Migrating existing data**

All other settings can be used as the default values.

1. In the **Table mappings** section:
   * Choose **Wizard**.
   * Expand **Selection rules** and choose **Add new selection rule**
     + For **Schema** select **Enter a schema**
2. In the **Premigration assessment** section:
   * Uncheck **Turn on premigration assessment**
3. Choose **Create task**

Your replication from the EC2 database instance to Aurora MySQL is now running, and the data is being replicated.

**Monitor your task**

 The database replication can take several minutes to complete. Wait for replication status to update to Load complete before continuing to the next steps.

You may occasionally choose refresh  - to see the status update.

**Task 2.6: Update DNS records**

Now that all databases have been migrated to Aurora, it’s time to update the DNS information so the application server can connect to the related Aurora database server. Both apps are using a DNS entry as a connection hostname. In a real-world application migration, once you have completed all of your testing and are ready to fully transition your databases to Aurora, you should perform the shutdown of the source servers and update the DNS records properly to reflect the new database servers running in Aurora.

1. At the top of the AWS Management Console, in the search bar, search for and choose **EC2**.
2. In the left navigation pane, under **Instances**, choose **Instances**.

**Note:** If the menu is collapsed, choose the menu  icon.

1. Select **MID-Wordpress-DB**.
2. Choose **Connect** from the navigation bar.
3. With **Session Manager** tab selected, choose **Connect**.

First, check the current database DNS record running:

1. **Command:** On the **MID-Wordpress-DB** instance, run the following command, to get the current DNS configuration for wordpress-db:

nslookup wordpress-db

**Expected output:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* EXAMPLE OUTPUT \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Server: 192.168.1.250

Address: 192.168.1.250#53

Name: wordpress-db.onpremsim.env

Address: 192.168.1.238

1. **Command:** Run the following command.

* Replace the **ENDPOINT** placeholder value with the database Writer **Endpoint** value you copied earlier, and run the command. This creates a variable to store the endpoint of the target database:

ADDR="ENDPOINT"

1. **Command:** Run the following command to update DNS, to update the DNS configuration of wordpress-db to point to the target database:

HOST="wordpress-db.onpremsim.env"

sudo touch /tmp/nsupdate.txt

sudo chmod 666 /tmp/nsupdate.txt

echo "server dns.onpremsim.env" > /tmp/nsupdate.txt

echo "update delete $HOST A" >> /tmp/nsupdate.txt

echo "update delete $HOST PTR" >> /tmp/nsupdate.txt

echo "update add $HOST 86400 CNAME $ADDR." >> /tmp/nsupdate.txt

echo "send" >> /tmp/nsupdate.txt

sudo nsupdate /tmp/nsupdate.txt

Verify the DNS name resolution again and check if it was updated to a CNAME pointing to your Aurora database (compare the output with the previous DNS lookup):

1. **Command:** Run the following command, to check that the DNS configuration of wordpress-db was updated succesfully:

nslookup wordpress-db

**Expected output:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* EXAMPLE OUTPUT \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Server: 192.168.1.250

Address: 192.168.1.250#53

wordpress-db.onpremsim.env canonical name = mid-wordpress-instance-1.c0rsfcvnchuj.us-west-2.rds.amazonaws.com.

Name: mid-wordpress-instance-1.c0rsfcvnchuj.us-west-2.rds.amazonaws.com

Address: 10.0.0.68

Shutdown the source database server as the database replication task is complete.

1. **Command:** Run the following command to shutdown the source server:

sudo shutdown -h now

**Task complete:** You have successfully created a replication subnet, replication instance, source endpoint, target endpoing, a migration task, migrated the database and finish the switch over.

**Conclusion**

 Congratulations! You now have successfully:

* Created an Amazon Aurora database
* Migrated the existing database to Aurora with AWS DMS
* Updated the DNS records to reflect the migration

**End lab**

Follow these steps to close the console and end your lab.

1. Return to the **AWS Management Console**.
2. At the upper-right corner of the page, choose **AWSLabsUser**, and then choose **Sign out**.
3. Choose **End lab** and then confirm that you want to end your lab.

**Additional resources**

* [AWS DMS Documentation](https://docs.aws.amazon.com/dms/).

For more information about AWS Training and Certification, see [*https://aws.amazon.com/training/*](https://aws.amazon.com/training/).

*Your feedback is welcome and appreciated.*  
If you would like to share any feedback, suggestions, or corrections, please provide the details in our [*AWS Training and Certification Contact Form*](https://support.aws.amazon.com/#/contacts/aws-training).