Name: THARUN R

Date: 05.12.2024

Mail ID: tharunr0219@gmil.com

Project Question:

here are the details of demo Project assessment:

Create your own website

Question:

Create a solution to migrate data from an on-premises database to AWS:

1. Use AWS DMS to replicate a MySQL database into an RDS MySQL instance.

2. Write a Python script to monitor the replication process and log the status into CloudWatch.

3. After migration, ensure that the RDS instance is encrypted and publicly inaccessible.

Answers:

1. Use AWS DMS to replicate a MySQL database into an RDS MySQL instance.

   Here's a detailed **step-by-step guide** to migrate your on-premises MySQL database to an Amazon RDS MySQL instance using **AWS Database Migration Service (DMS)**.

   - **Step 1: Prepare the Target (RDS MySQL) Database**

   - Log in to the **AWS Management Console**

   - Go to **RDS > Databases > Create Database**.

   - **Select Database Engine**: Choose **MySQL**.

   - **Instance Settings**:
     - Select the version compatible with your source MySQL database.
     - Choose the instance type and storage based on your data requirements.

   - **Networking**:
     - ❖ Ensure your RDS instance is in a VPC that can communicate with your on-premises database.

- ❖ Allow **public access** if needed, or configure private connectivity via **AWS Direct Connect** or **VPN**.

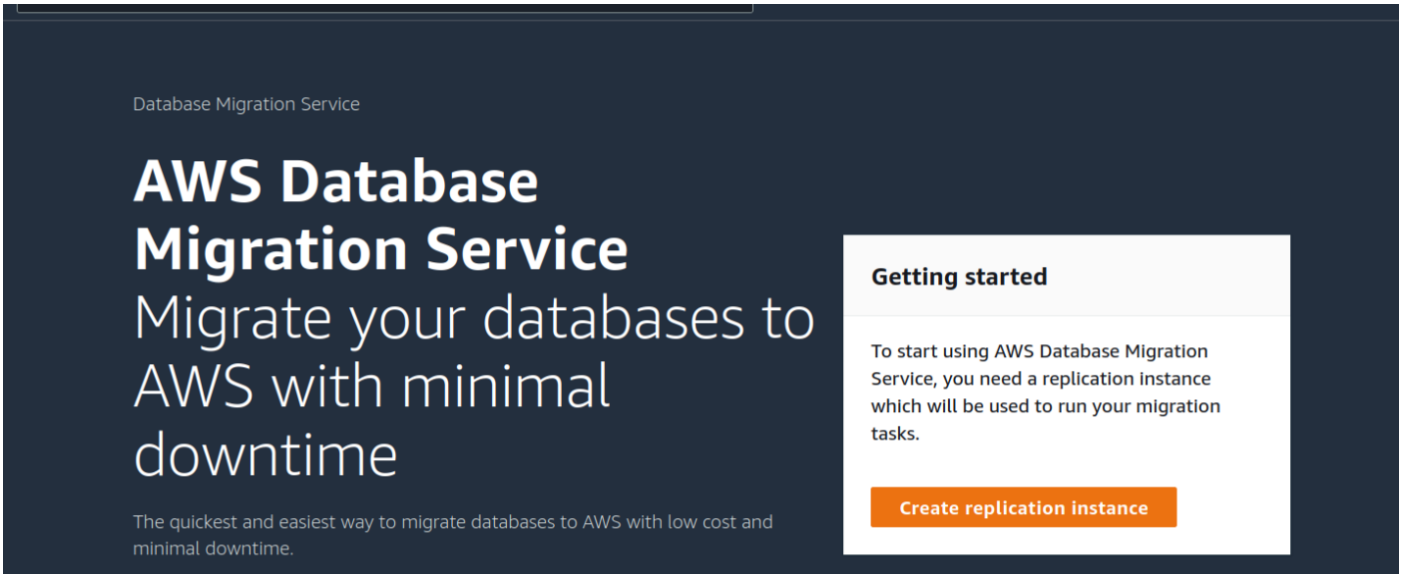
- **Security**:
  - ❖ Set up a security group to allow inbound traffic from your on-premises database server on port **3306**.

- **Create the Database** and note the endpoint, username, and password.

Setup a Replication Instance

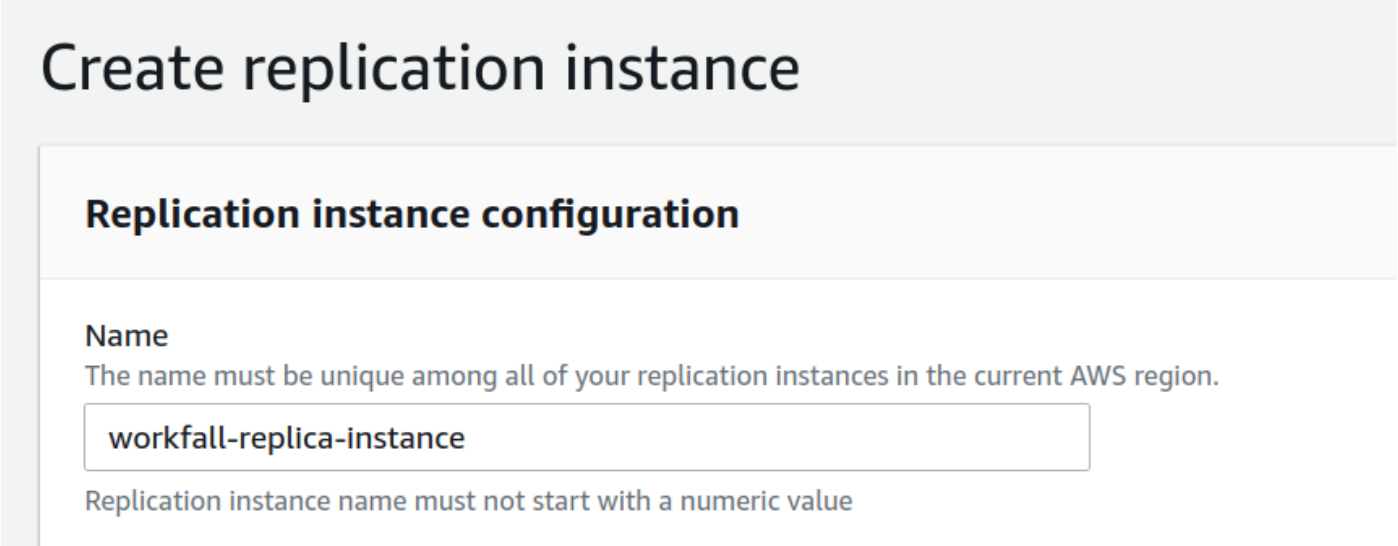
To setup a replication instance, go to AWS DMS console

Click create replication instance



We have to configure the replication instance

Enter the name for the replication instance.



For the Instance class, choose the preferred instance type and the Engine Version be a default.

Instance class Info

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

dms.t3.medium
2 vCPUs 4 GiB Memory

Include previous-generation instance classes

Engine version

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

3.4.4

Include Beta DMS versions

Enter the size of the storage required for the replication instance.

Allocated storage (GiB) Info

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

Choose the VPC where the replication instance should be created.

If you want the replication instance to be deployed in Multi AZ mode, select it.

VPC

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

vpc-089fb4e8acddc309b - workfall-vpc

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

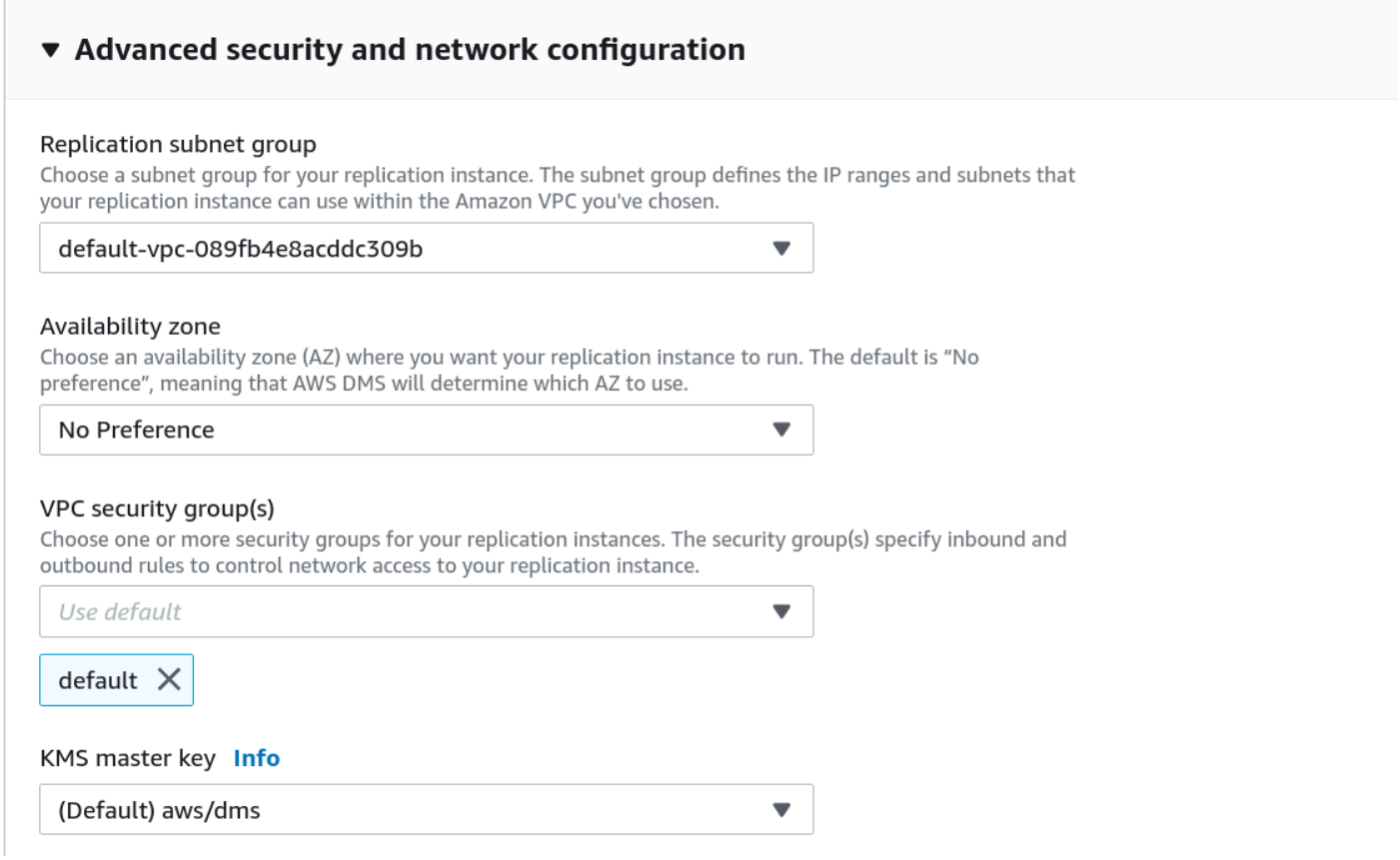
If you choose this option, AWS DMS will assign a public IP address to your replication instance, and you'll be able to connect to databases outside of your Amazon VPC.

Under advanced security and network configuration,

Select the default VPC subnet group.

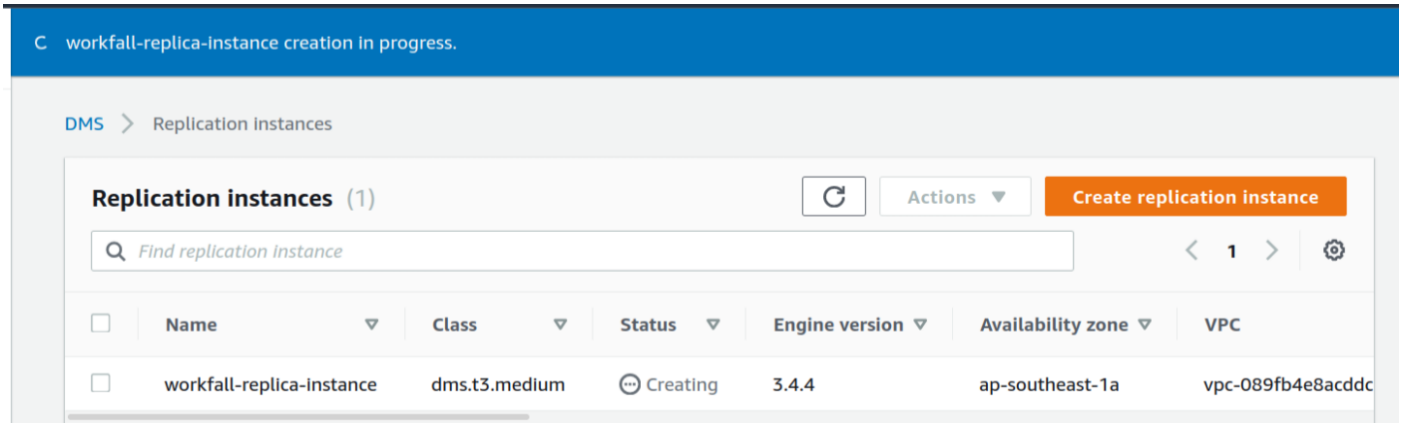
For the VPC security group(s), choose the default security group.

The default KMS master key will be used.



And finally click create.

The Replication instance has been created.



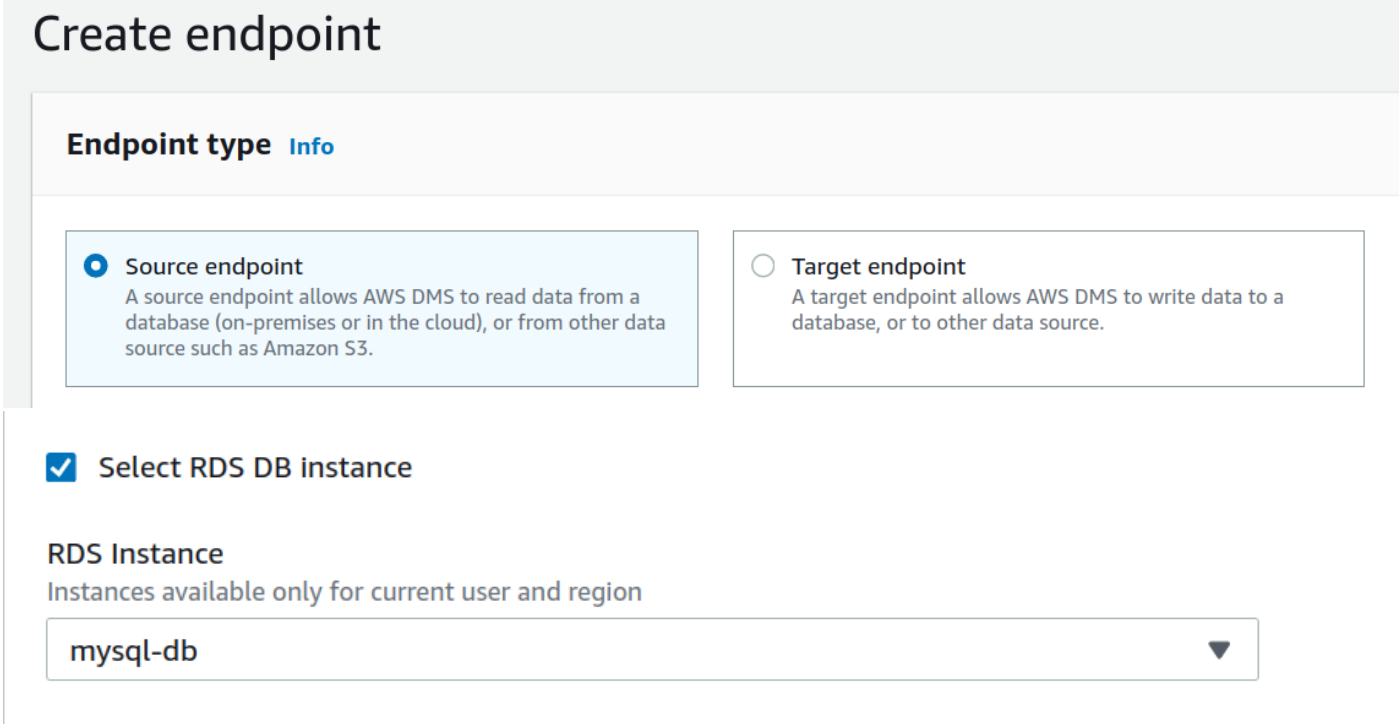
The Replication instance is successfully created and is available.

Creating Endpoint for Source Database (MySQL)

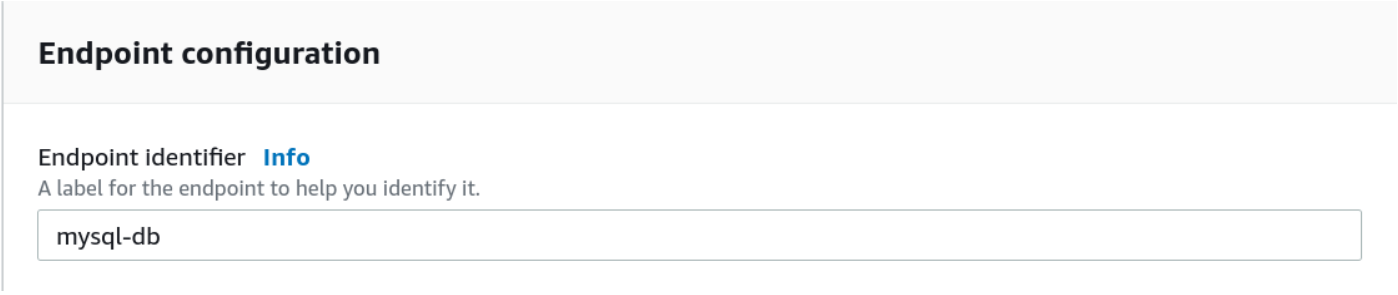
In the navigation pane, click endpoints and then choose Create endpoint.

For the endpoint type, choose Source endpoint.

Check, select RDS DB Instance and then choose the Source RDS DB Instance which is MySQL.



For the Endpoint configuration, enter the Endpoint identifier (It is fetched by default after choosing the RDS DB instance).



For Source Engine, choose MySQL.

For access to the endpoint database, select Provide access information manually.

Which will automatically fetch the source DB endpoint, Port, and the User name.

Make the changes if required and Enter the Password of the MySQL RDs DB instance.

Access to endpoint database

- ○ Choose AWS Secrets Manager
- ◉ Provide access information manually

Server name

mysql-db.cpwh4tcmgzd7.ap-southeast-1.rds.amazonaws.com

Port
The port the database runs on for this endpoint.

3306

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

none

User name Info

admin

Password Info

••••••••

We can test the connectivity to the RDS MySQL DB instance, by doing so.

Expand test endpoint connection (optional)

Choose the VPC and select the Replication instance then click Run test.

If all the provided information is correct, then the status should be successful.

Test endpoint connection (optional)

VPC

vpc-089fb4e8acddc309b - workfall-vpc

Replication instance
A replication instance performs the database migration

workfall-replica-instance

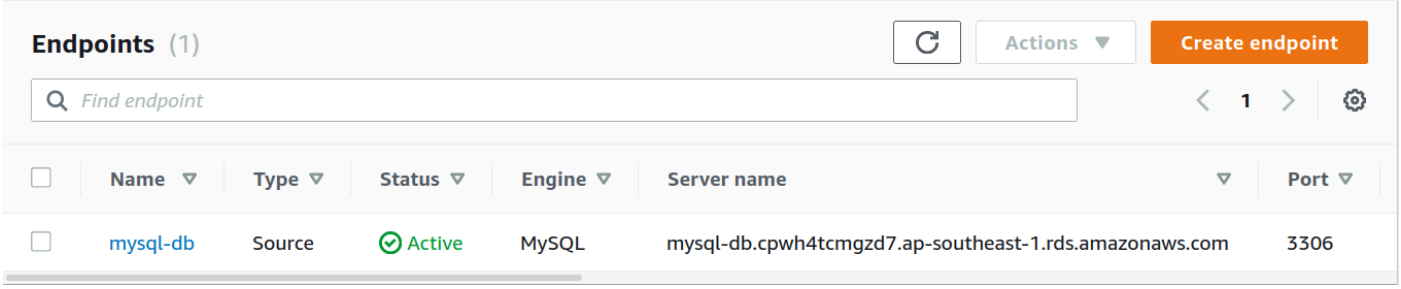
Your endpoint will always be created even if the connection fails
After clicking 'Run test', DMS creates the endpoint with the details you provided and attempts to connect to it. If the connection fails, you can edit the endpoint definition and test the connection again. You can also delete the endpoint manually.

Run test

Endpoint identifier	Replication instance	Status	Message
mysql-db	workfall-replica-instance	successful	

And finally click create Endpoint.

The Endpoint for the RDS MySQL instance is created and is Active.

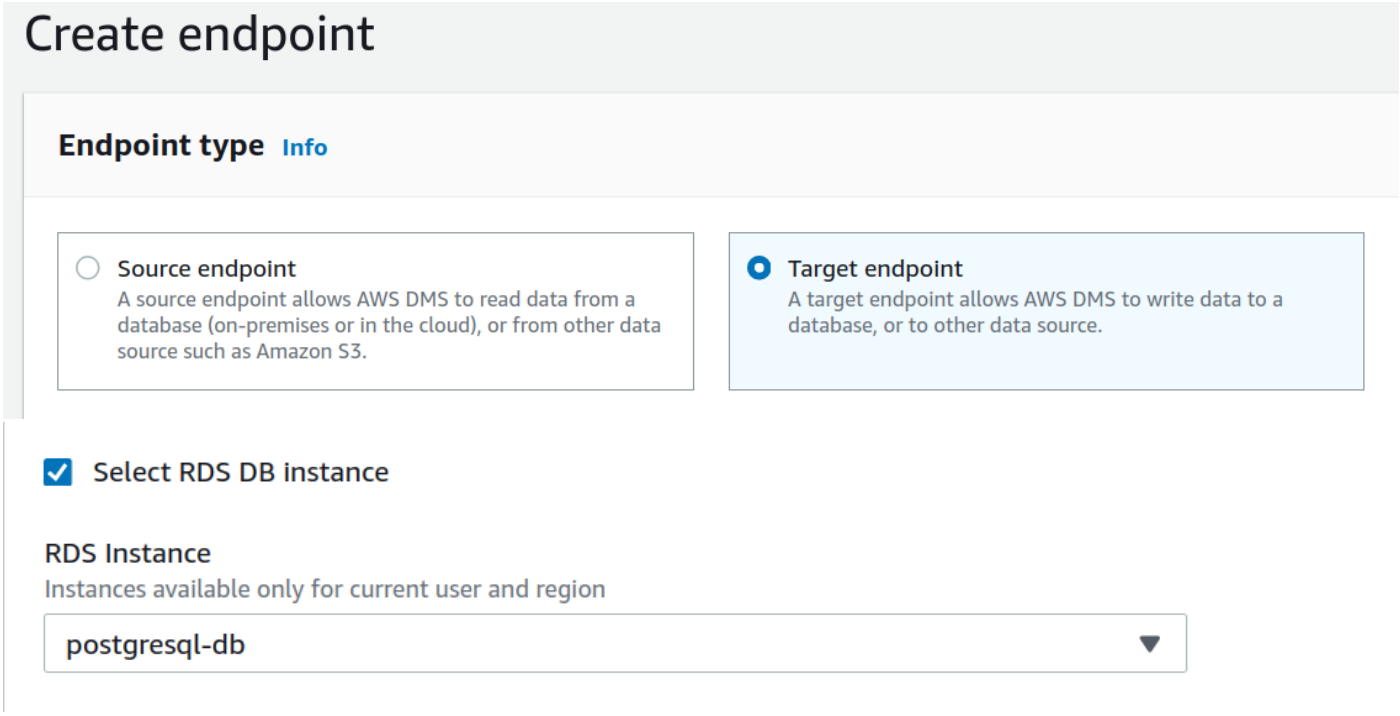


Creating Endpoint for Target Database (PostgreSQL)

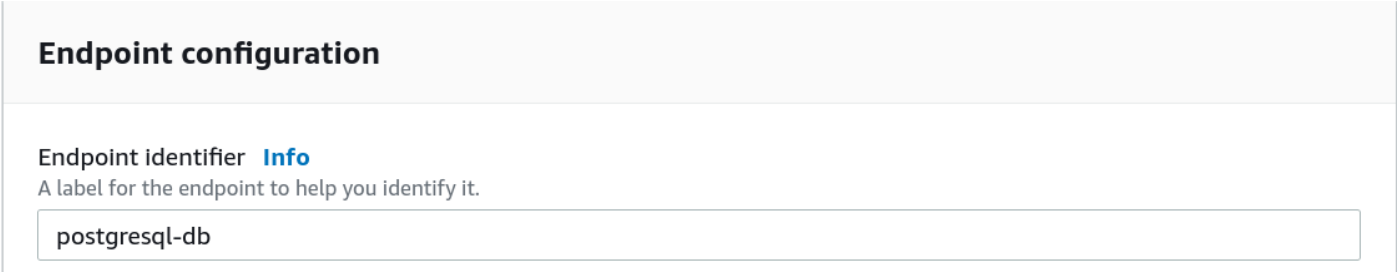
Click Create endpoint.

For the Endpoint type, choose Target endpoint.

Select RDS DB Instance and then select the Target RDS DB Instance which is PostgreSQL.



For the Endpoint configuration, enter the Endpoint identifier (It is fetched by default after choosing the RDS DB instance).



For the Target engine, choose PostgreSQL.

Access to the endpoint database, choose to provide access information manually.

It automatically fetches the Target DB endpoint, Port, and User name.

Make the required changes if any and Enter the Password of the PostgreSQL RDS DB instance.

Access to endpoint database

- ○ Choose AWS Secrets Manager
- ◉ Provide access information manually

Server name

mysql-db.cpwh4tcmgzd7.ap-southeast-1.rds.amazonaws.com

Port
The port the database runs on for this endpoint.

3306

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

none

User name Info

admin

Password Info

••••••••

And Enter the Target Database name.

We can test the connectivity to the RDS PostgreSQL DB instance, to expand test endpoint connection (optional).

Choose the VPC and the Replication instance and then click Run test.

If the provided information is correct, the connectivity test should be successful.

▼ **Test endpoint connection (optional)**

VPC

vpc-089fb4e8acddc309b - workfall-vpc

Replication instance
A replication instance performs the database migration

workfall-replica-instance

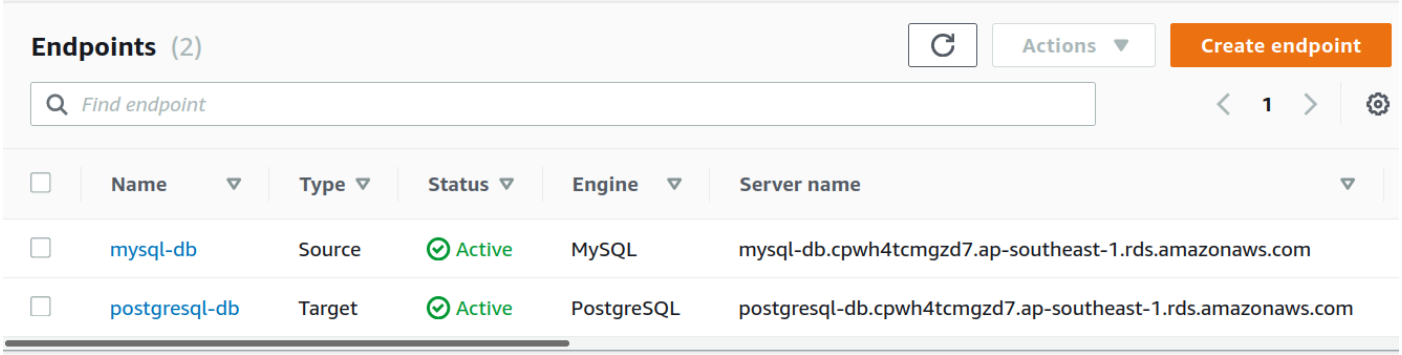
⚠ **Your endpoint will always be created even if the connection fails**
After clicking 'Run test', DMS creates the endpoint with the details you provided and attempts to connect to it. If the connection fails, you can edit the endpoint definition and test the connection again. You can also delete the endpoint manually.

Run test

Endpoint identifier	Replication instance	Status	Message
postgresql-db	workfall-replica-instance	successful	

And at last click Create endpoint.

The Endpoint for the RDS PostgreSQL instance is created successfully and is Active.



Set Up AWS DMS

1. Create a Replication Instance

1. **Go to the AWS DMS Console.**
2. **Navigate to Replication Instances > Create Replication Instance.**
3. **Configure:**
   - **Name: Provide a unique name.**
   - **Instance Class: Choose based on database size (e.g., dms.r5.large).**
   - **VPC: Select the same VPC as your RDS instance.**
4. **Click Create and wait for the replication instance to become available.**

---

2. Create Endpoints

Source Endpoint (On-Premises MySQL):

1. **In the DMS Console, navigate to Endpoints > Create Endpoint.**
2. **Configure:**
   - **Endpoint Type: Source.**
   - **Database Engine: MySQL.**
   - **Endpoint Identifier: Provide a unique name.**
   - **Server Name: Enter the IP or hostname of your on-premises database.**
   - **Port: 3306.**
   - **Username: dms_user (created earlier).**
   - **Password: The password for dms_user.**
3. **Test the connection.**

Target Endpoint (RDS MySQL):

1. **Repeat the process, selecting Target as the endpoint type.**
2. **Use the RDS endpoint details (host, username, password).**
3. **Test the connection.**

---

3. Create a Migration Task

1. **Navigate to Tasks > Create Task.**
2. **Configure:**
   - **Name:** Provide a task name.
   - **Replication Instance:** Select the instance created earlier.
   - **Source Endpoint:** Choose the on-premises MySQL endpoint.
   - **Target Endpoint:** Choose the RDS MySQL endpoint.
   - **Migration Type:**
     - **Full Load: For one-time migration.**
     - **Full Load + Ongoing Replication: For minimal downtime migration.**
3. **Table Mappings:**
   - Use the default to migrate all tables.
   - Or customize using the Table Mapping Editor.
4. **Start Task on Create: Check this option to start the migration immediately.**
5. **Click Create Task.**

Step 4: Monitor and Validate the Migration

1. **Monitor the Task:**
   - Go to Tasks > Task Monitoring.
   - Check progress and logs for any errors.
2. **Validate Data:**
   - Compare row counts and sample queries between source and target databases to ensure consistency.

Step 5: Finalize the Migration

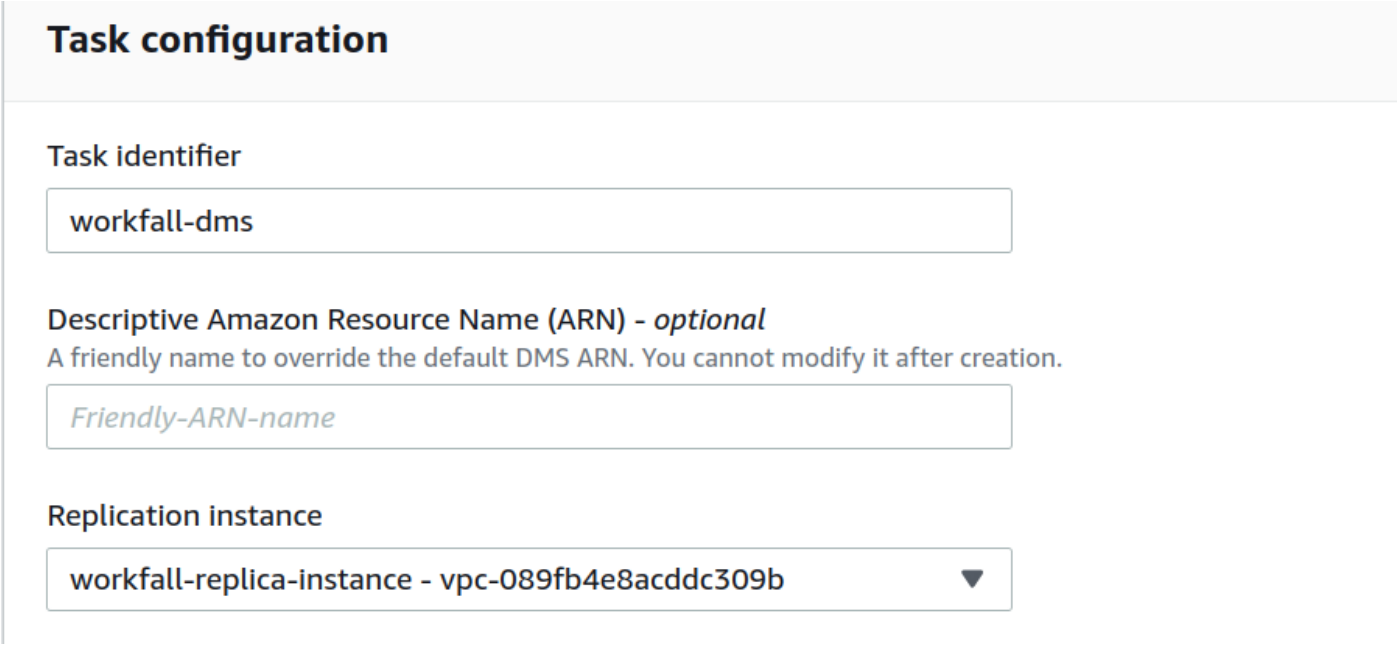
1. **Switch Applications:**
   - Once all data is migrated and validated, redirect your application to the RDS MySQL instance.
2. **Stop Ongoing Replication:**
   - If using ongoing replication, stop the DMS task after completing the final sync.
3. **Optimize the RDS Database:**
   - Review and adjust database parameters.
   - Enable backups, monitoring, and performance insights.

Creating Database Migration Task

To create a migration task, in the navigation pane, click Database migration tasks.

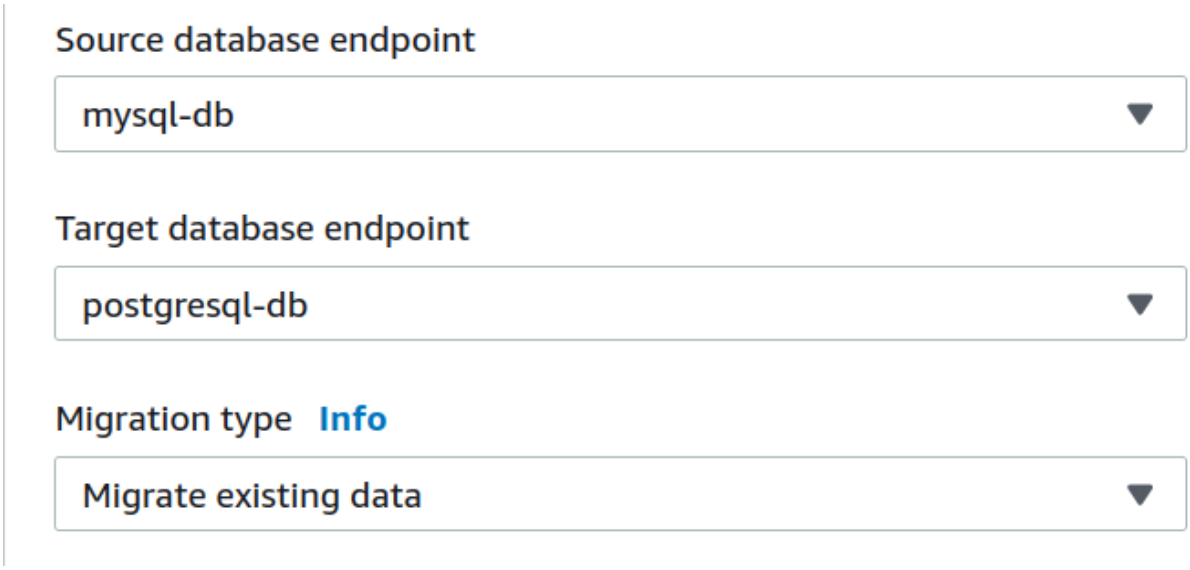
Click create task, enter the name for the task.

Select the Replication instance that you have created.



Select the Source database endpoint and the target database endpoint.

For the Migration type, select Migrate existing data.

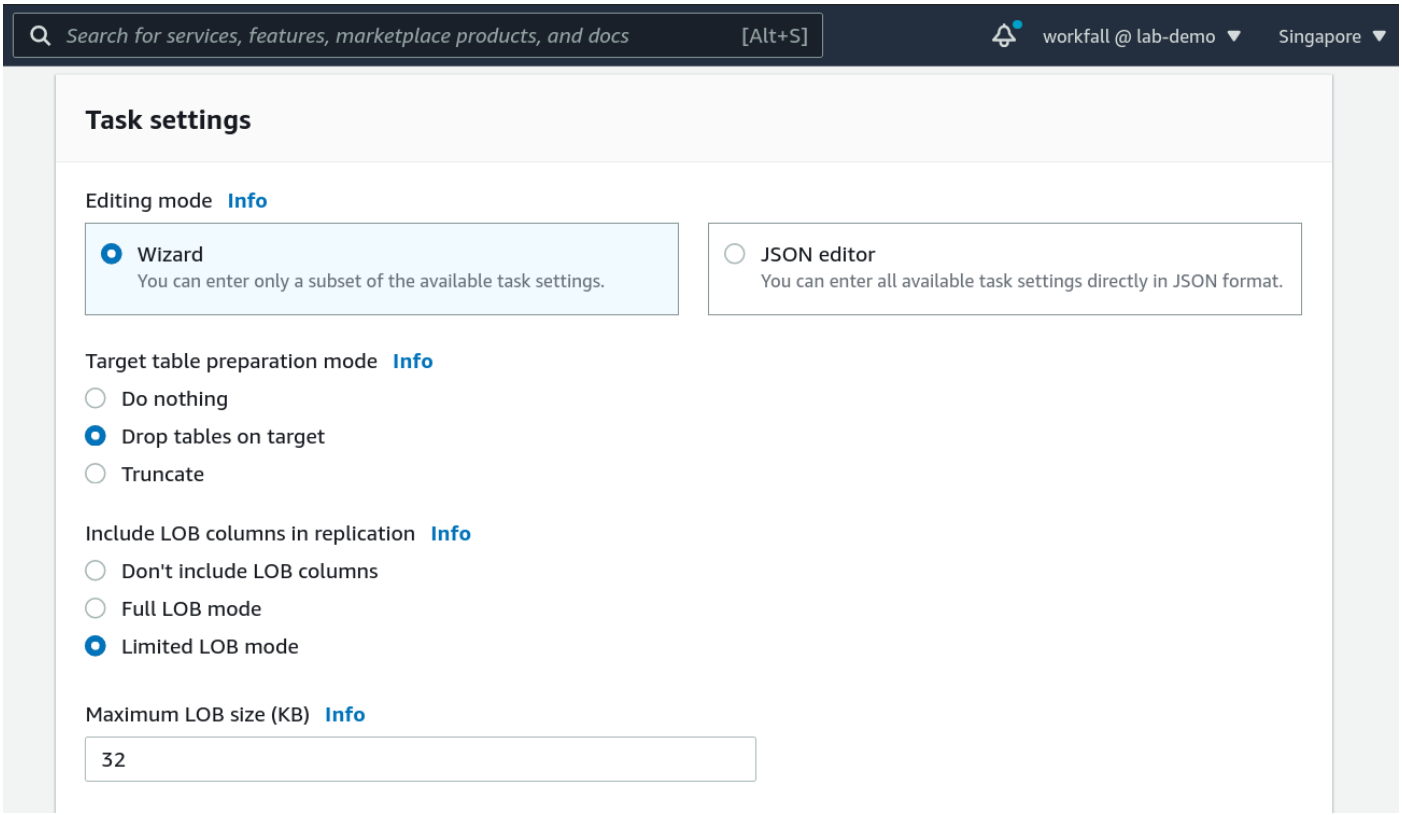


As we perform migration from MySQL to PostgreSQL engine, the AWS schema conversion tool will automatically convert the database scheme.

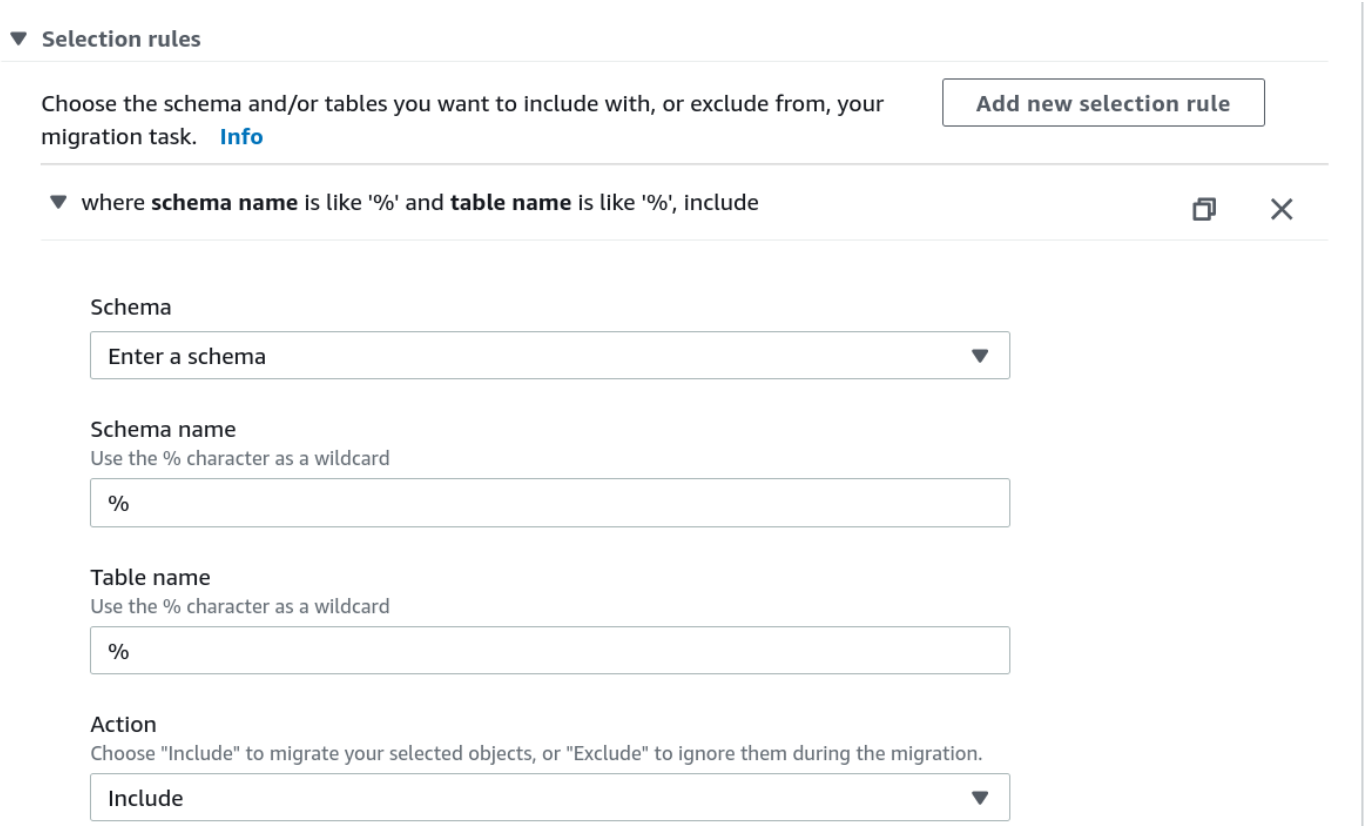
Under Task settings,

For the Target table preparation mode, choose Do nothing

To Include LOB columns in replication, choose Limited LOB mode and the Maximum LOB size be 32KB.

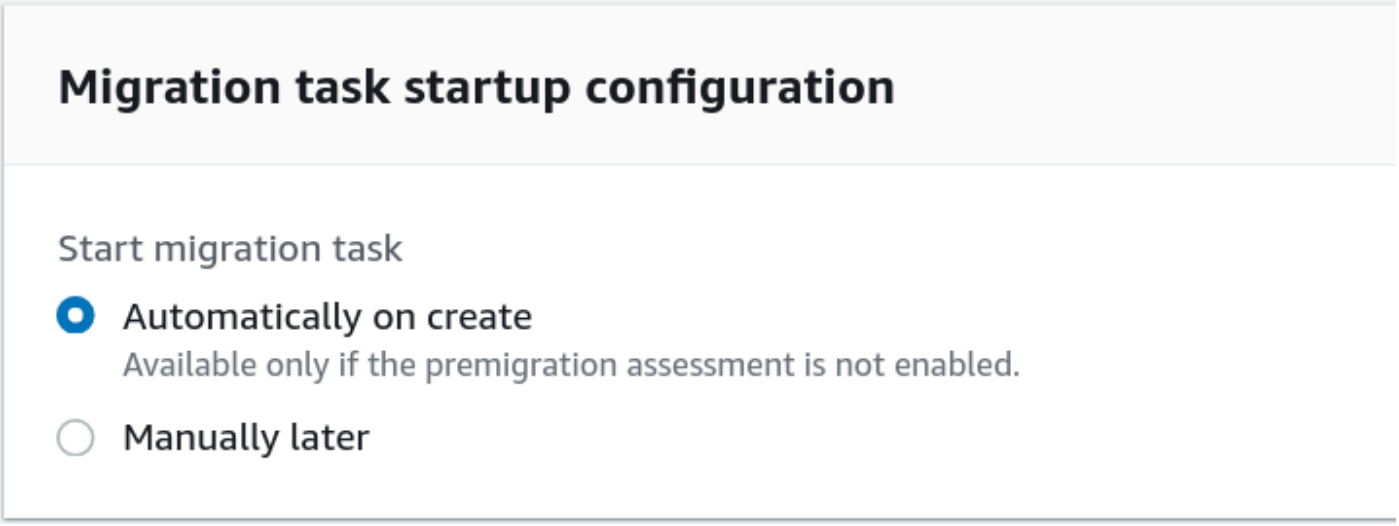


For Table mappings, add a new selection rule.

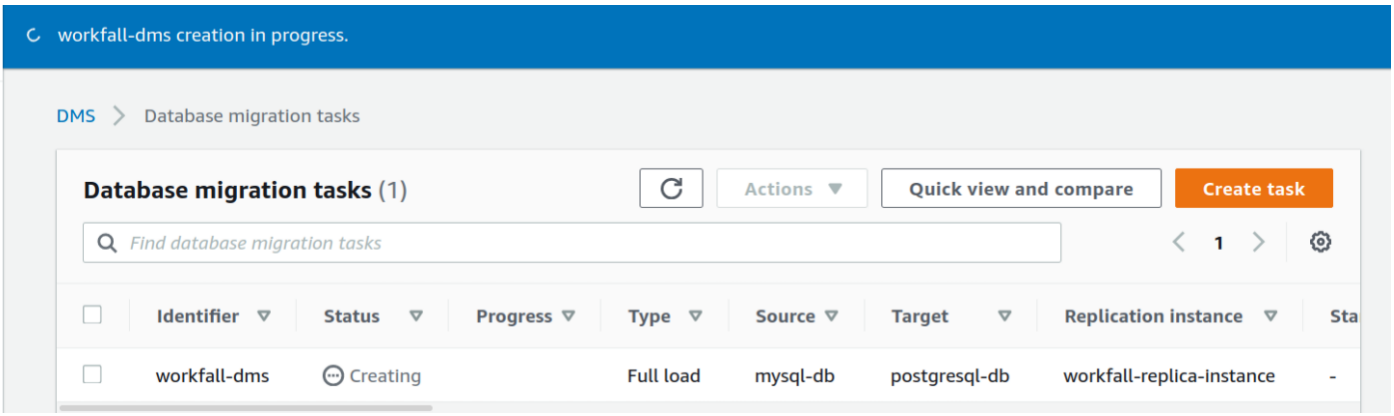


For Migration task startup configuration, choose Automatically on create.

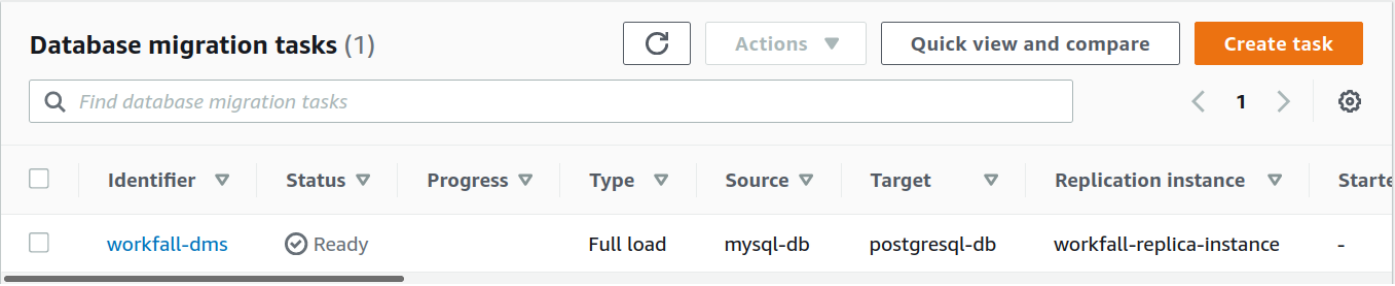
And finally click create task which will start the migration process immediately.



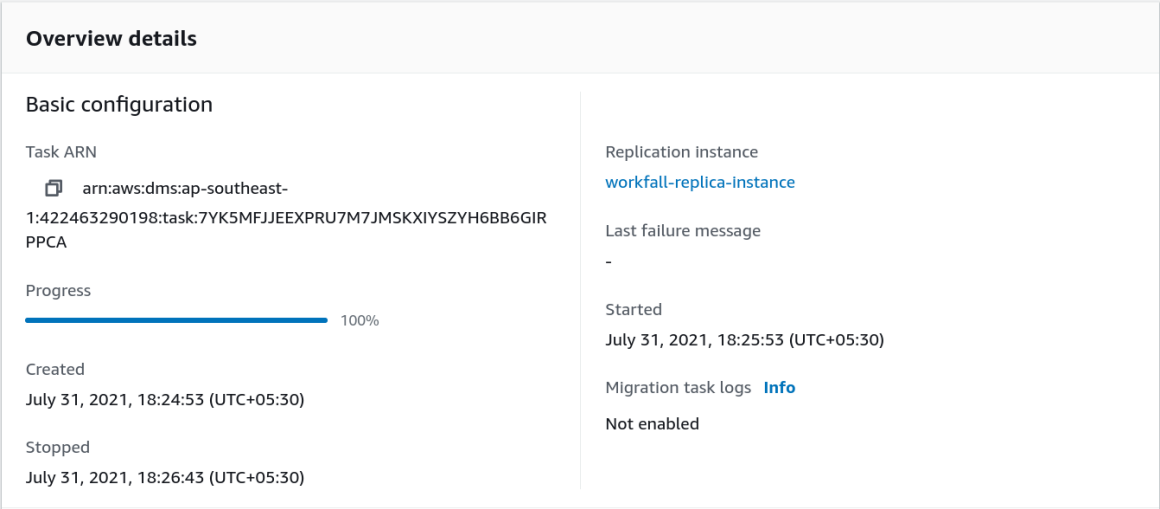
The Database migration task creation is in progress.



The Migration task is ready now.



And you can see the migration process has started immediately.



2. Write a Python script to monitor the replication process and log the status into CloudWatch.

1. Set Up Prerequisites

a. AWS CLI Configuration

Make sure you have AWS CLI installed and configured with credentials that have the necessary permissions:

- Install AWS CLI: AWS CLI Installation Guide
- Configure AWS credentials: bash

```
aws configure
```

Provide your access key, secret key, region, and output format when prompted.

b. IAM Permissions

Ensure the AWS credentials or IAM role used have the following permissions:

- logs:CreateLogGroup
- logs:CreateLogStream
- logs:PutLogEvents
- logs:DescribeLogStreams

c. Install Required Libraries

Install the required Python library boto3:

bash

```
pip install boto3
```

2. Prepare the Script

a. Save the Script

Save the provided Python script into a file, for example, replication_monitor.py.

b. Modify the Script

1. Update the AWS region:

python

```
REGION_NAME = "us-east-1"  # Replace with your AWS region
```

2. Replace the check_replication_status function with logic to check your actual replication process. For example:
    - Query a database.
    - Call an API that tracks replication progress.
    - Parse a file or logs.

Example for a database:

python

```
import psycopg2

def check_replication_status():
    # Example for PostgreSQL
    conn = psycopg2.connect("host=mydbhost dbname=mydb user=myuser password=mypassword")
    cursor = conn.cursor()
    cursor.execute("SELECT status FROM replication_status_table WHERE id=1;")
    status = cursor.fetchone()[0]
    conn.close()
    return status
```

3. Run the Script

a. Execute the Script

Run the script from the command line:

bash

```
python replication_monitor.py
```

b. Monitor the Output

The script will:

1. Check the replication status every CHECK_INTERVAL seconds (default is 60 seconds).
2. Log status updates to the console.
3. Send logs to AWS CloudWatch under the specified log group and stream.

4. Verify in CloudWatch

a. Log Group

1. Go to the CloudWatch Console.
2. Navigate to **Logs > Log Groups**.
3. Find the log group named ReplicationMonitorLogs.

b. Log Stream

1. Click on the log group.
2. Select the log stream named ReplicationStatusStream.
3. View the logs with timestamps and replication status updates.

5. Adjust and Deploy

a. Deployment Options

- **Local Monitoring:** Run the script on your local machine.
- **Cloud/Server Monitoring:** Deploy the script on a server (e.g., EC2, ECS, or Lambda).
    - For AWS Lambda, ensure the script is event-driven.
    - For EC2, use a scheduler like cron for periodic execution.

b. Adjust Check Interval

Modify the CHECK_INTERVAL variable in the script to suit your replication process needs:

python

```
CHECK_INTERVAL = 60 # Interval in seconds
```

c. Add Alerts (Optional)

Set up CloudWatch Alarms to trigger notifications (via SNS) if replication fails:

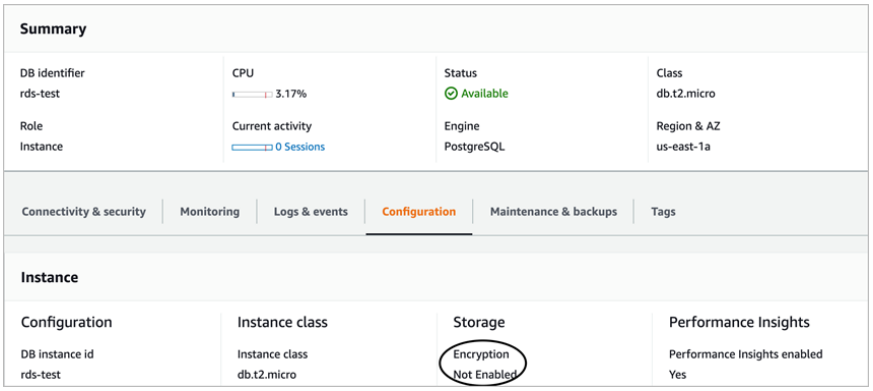
1. Navigate to CloudWatch **Alarms**.
2. Create an alarm for specific log patterns (e.g., "Replication failed").
3. Associate the alarm with an SNS topic for notifications

---

3. After migration, ensure that the RDS instance is encrypted and publicly inaccessible.

Steps:

1. **Verify Encryption**:
    - Navigate to the **Amazon RDS Console**.
    - Select the instance and check the **Encryption** field in the instance details.
    - If not encrypted, proceed to create a new instance.
2. **Create a New Encrypted RDS Instance**:
    - In the RDS Console, choose **Create database**.
    - Enable **Encryption** under the "Additional configuration" section.
    - Select the desired KMS key for encryption.
3. **Migrate Data**:
    - Use tools like **AWS Database Migration Service (DMS)** or manual export/import methods to migrate data from the source to the new encrypted instance.



Ensure the RDS instance is not exposed to the public internet.

Steps:

1. **Verify Public Accessibility**:
    - In the RDS Console, check the **Publicly Accessible** field in the instance details.
    - It should be **No**.
2. **Modify the Instance** (if necessary):
    - In the RDS Console, select your instance and click **Modify**.
    - Set **Public access** to **No** under "Connectivity".
    - Apply changes.
3. **Update Security Groups**:
    - Go to the **EC2 Console** > **Security Groups**.
    - Identify the security group attached to your RDS instance.
    - Ensure there are no **0.0.0.0/0** or **::/0** inbound rules.
    - Restrict access to specific IPs or VPCs as needed.
4. **Use Private Subnets**:
    - Confirm the RDS instance is deployed in private subnets within your VPC.
    - Ensure that the associated **subnet group** contains only private subnets.

Verification

1. Test database connectivity to ensure encryption and private accessibility.
2. Validate with the **AWS Config Rules**:
    - Use predefined rules like:
        - rds-instance-public-access-check: Ensures RDS instances are not publicly accessible.
        - rds-storage-encrypted: Checks if RDS instances have encryption enabled.

Optional Enhancements

- Use **AWS CloudWatch** to monitor DMS and RDS performance.
- Enable **SSL Encryption** for secure communication between the source and target databases.
- Use **AWS Backup** to create automatic backups of your RDS instance.