# RDS-TASKS

| **Field** | **Details** |
| --- | --- |
| **Title** | End-to-End Setup of EC2 MariaDB/MySQL, RDS Database Creation, Migration, Backup, Snapshot, and Read Replica Operations |
| **Objective** | To install and configure MariaDB/MySQL on an EC2 instance, create local databases and tables, migrate them to RDS MariaDB/MySQL instances, validate connectivity, perform backups/snapshots, and attempt read replica creation. |
| **Prerequisites** | • EC2 instance (Amazon Linux 2023) with SSH access• Security group allowing port 3306 between EC2 ↔ RDS• MySQL/MariaDB installed on EC2• AWS RDS service enabled• IAM permission to create DB instances, snapshots, and secrets• AWS Secrets Manager access (for MariaDB RDS)• Backup file (mybackup.sql) created using mysqldump |
| **Errors Faced** | **1. Database creation syntax error** while executing SQL file because DBName contained a hyphen (my-db-1).**2. Unable to connect RDS initially** because EC2 was not linked as a compute resource and SG rules were incomplete.**3. Free-tier limitation error** when creating a read replica: “You reached the maximum number of instances available with free plan.” |
| **How I Fixed It** | **1. Fixed SQL error** by using a valid DB name (my\_ec2\_db) and recreating the SQL script without hyphens.**2. Resolved connectivity issue** by selecting **“Connect to an EC2 compute resource”** during RDS creation → AWS automatically attached correct SG rules allowing 3306 inbound from EC2.**3. Read replica error** identified as an account limitation; read replica requires quota upgrade or deletion of existing DB instances. |
| **Conclusion** | Successfully installed MariaDB/MySQL on EC2, created databases and tables, generated backup, restored to RDS MariaDB/MySQL using RDS endpoint, validated migrated data, created manual snapshots, and explored creation of read replicas. The full migration workflow works correctly; remaining limitation is AWS free-tier capacity for additional managed instances. |

1. Create MariaDB DB on EC2.

**1️⃣ Modified Security Group Inbound Rules**

* Added/edited **inbound rules** in an EC2 Security Group.
* Allowed:
  + **HTTP (80)** from anywhere (0.0.0.0/0)
  + **SSH (22)** from anywhere
  + **All traffic** from the same security group (**sg-070d0bcbe8b88bed5**)
  + **MySQL/Aurora (3306)** allowed from **Anywhere (0.0.0.0/0)** ← This is a major security risk.

**2️⃣ Installed MariaDB Server on Amazon Linux 2023**

Commands shown:

dnf install mariadb105-server mariadb105 -y

Output shows:

* MariaDB packages installed
* Dependencies resolved successfully

**3️⃣ Enabled and Started MariaDB Service**

Commands used:

systemctl enable --now mariadb

systemctl status mariadb

Service started successfully  
 mysqld (mariadbd) running

**4️⃣ Verified That MariaDB is Listening on Port 3306**

Used:

lsof -i tcp:3306

Output:

* mariadbd is listening on \*:mysql (port 3306)

**5️⃣ Set Environment Variables for DB Creation**

executed:

DBName=my\_ec2\_db

DBPassword=seema1716

DBRootPassword=seema1716

DBUser=ec2dbuser

Then verified:

echo $DBName

echo $DBPassword

echo $DBRootPassword

echo $DBUser

All variables printed correctly.

**6️⃣ Prepared SQL Commands in a Temporary File**

User created a file /tmp/db.setup with:

CREATE DATABASE $DBName;

CREATE USER '$DBUser' IDENTIFIED BY '$DBPassword';

GRANT ALL PRIVILEGES ON \*.\* TO '$DBUser'@'%';

FLUSH PRIVILEGES;

But the image shows an error:

The file contained:

CREATE DATABASE my-db-1;

**7️⃣ Tried Executing SQL File**

Command:

mysql -u root -p${DBRootPassword} < /tmp/db.setup

Error returned:

ERROR 1064 (42000): You have an error in your SQL syntax near '-db-1' at line 1

This is due to invalid DB name.

**8️⃣ Removed the Broken Script File**

deleted the file:

rm /tmp/db.setup

A screenshot of a computer

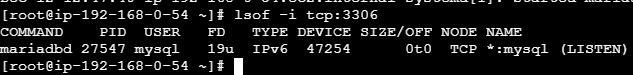
AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

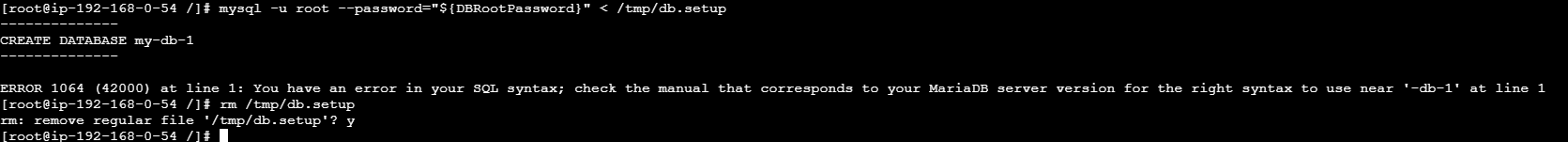


A computer screen with white text

AI-generated content may be incorrect.

A computer screen with text on it

AI-generated content may be incorrect.



1. Insert some dummy data.

A screenshot of a computer program

AI-generated content may be incorrect.

A computer screen shot of white text

AI-generated content may be incorrect.

1. Take the backup of dummy data on EC2.

**Logged into MariaDB Using Root Credentials**

ran:

mysql -u root --password="${DBRootPassword}"

Successfully logged into MariaDB.

**Tried to Use the Database (Before Creating It)**

attempted:

use my\_ec2\_db;

Result:

ERROR 1049 (42000): Unknown database 'my\_ec2\_db'

This confirms the DB was not created earlier.

**Viewed Existing Databases**

Used:

SHOW DATABASES;

Existing DBs:

* information\_schema
* mysql
* performance\_schema

**Created the Database Successfully**

executed:

CREATE DATABASE my\_ec2\_db;

Output:

Query OK, 1 row affected

Then verified:

SHOW DATABASES;

Now the new DB is visible.

**Switched to the Database**

USE my\_ec2\_db;

Database changed successfully.

**Created First Table (table1)**

CREATE TABLE table1 (id INT, name VARCHAR(45));

Table created.

**Inserted Multiple Records into table1**

Command:

INSERT INTO table1 VALUES

(1, 'IMRAN'),

(2, 'MUJJU'),

(3, 'HAJI'),

(4, 'GAYAZ');

Insert successful.

**Selected Data From table1**

SELECT \* FROM table1;

Output:

| **id** | **name** |
| --- | --- |
| 1 | IMRAN |
| 2 | MUJJU |
| 3 | HAJI |
| 4 | GAYAZ |

**Created Second Table (table2)**

CREATE TABLE table2 (id INT, name VARCHAR(45));

Table created.

**Inserted Records into table2**

INSERT INTO table2 VALUES

(1, 'MUJJU'),

(2, 'SEEMA');

Insert successful.

**Viewed Data in table2**

SELECT \* FROM table2;

Output:

| **id** | **name** |
| --- | --- |
| 1 | MUJJU |
| 2 | SEEMA |

A screenshot of a computer

AI-generated content may be incorrect.

1. Launch MariaDB RDS instance.

**Selecting DB Creation Method & Engine**

* Chose **Full configuration** instead of Easy Create.
* Selected **MariaDB** as the engine.
* Chose the supported **engine version (MariaDB 11.4.8)**.
* Picked the **Free tier** template.

**Configuring Basic Settings**

* Entered a **DB instance identifier** → database-1.
* Chose **Managed by AWS Secrets Manager** for credential management.
  + AWS automatically generated and stored master username/password in Secrets Manager.
* Encryption kept as **default (aws/secretsmanager key).**

**Choosing Instance Class & Storage Settings**

* Selected **db.t4g.micro** instance class (Free Tier eligible).
* Storage type selected: **General Purpose SSD (gp2)**.
* Allocated storage: **20 GB default (visible in screenshot as 400 but automatically scalable)**.
* Enabled **Storage autoscaling** so storage can grow as needed.

**Setting Up Connectivity**

* Selected **Connect to an EC2 compute resource**.
  + Attached to the EC2 instance shown in the screenshot (i-0477c29c5aa8b8f5).
* AWS automatically:
  + Created required VPC settings.
  + Attached a DB security group that allows the EC2 instance to connect.
* Chose **Automatic setup** for DB subnet group.

**Additional Network Settings**

* Database port left as **3306** (default for MariaDB).
* Did not add any tags.
* Chose **Password authentication**.

**Monitoring Configuration**

* Selected **Database Insights – Standard**.
* Did **not** enable Enhanced Monitoring.
* No log exports selected.

**Finalizing Database Creation**

* Created the MariaDB RDS instance successfully.
* On the final page:
  + DB status shows **Available**.
  + Endpoint and port (3306) displayed.
  + Security group details visible.
  + Instance is in **us-east-1a**.
  + Engine shown as **MariaDB**.
  + Public accessibility: **No** (the instance is private).

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

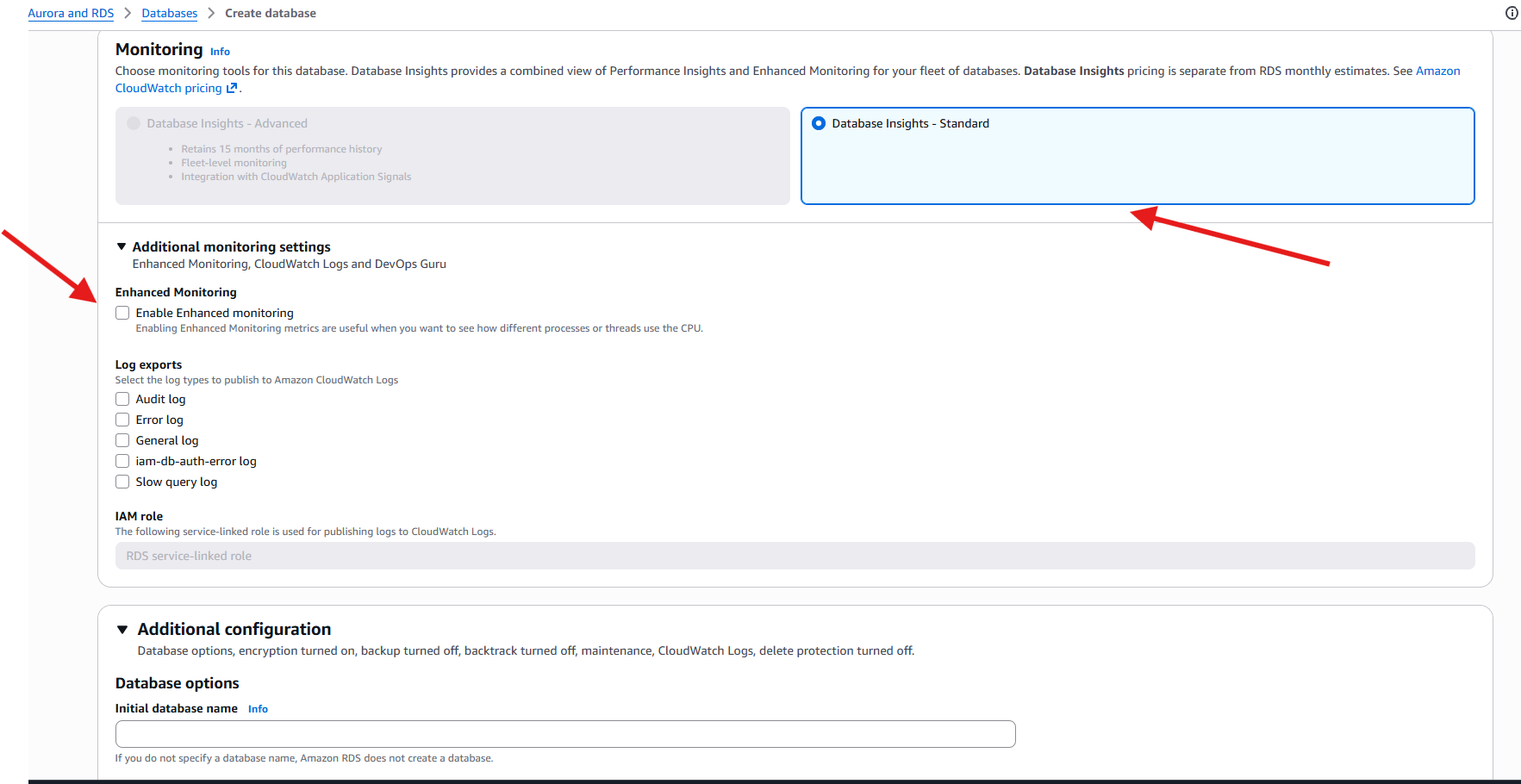
AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.



A screenshot of a computer

AI-generated content may be incorrect.

1. Migrate database from EC2 to RDS.

**Viewing the RDS Endpoint Information**

* opened the **RDS console** and viewed the details of the newly created MariaDB RDS instance.
* Observed:
  + **Endpoint URL** (used to connect from EC2).
  + **Port 3306**.
  + Database engine: **MariaDB**.
  + Availability zone and networking configuration.

**Checking RDS Configuration (Master Username & Secrets Manager Integration)**

* Viewed the **Configuration** tab of the RDS instance.
* Confirmed:
  + Master username → admin
  + Storage, instance class, monitoring, etc.
  + The RDS instance uses **AWS Secrets Manager** for storing credentials.
* Validated that Secrets Manager automatically generated the DB password.

**Opening the Secret in AWS Secrets Manager**

* The user opened AWS Secrets Manager.
* Viewed:
  + Stored username → admin
  + Stored password → autogenerated secure string.
* This secret is automatically attached to the RDS instance and used for authentication.

**Logging in to MariaDB on EC2 to Verify Existing Database Contents**

Before migrating to RDS, the user logged in to the **EC2 MariaDB local database**:

mysql -u root --password="${DBRootPassword}"

Inside MariaDB, they checked:

* Existing databases:
* SHOW DATABASES;
* Selected the EC2-created database:
* USE my\_ec2\_db;
* Displayed existing table data:
* SELECT \* FROM table1;
* Exited the MariaDB CLI.

This confirms that the EC2 MariaDB contains valid data to migrate.

**Restoring the EC2 Backup Into the RDS MariaDB Instance**

The user ran a restore command from EC2 to the RDS endpoint:

mysql -h database-2.c7comnecieho.us-east-1.rds.amazonaws.com -P 3306 -u admin -p my\_ec2\_db < /mybackup.sql

* This imports the EC2 backup into the RDS MariaDB database.
* Password was entered when prompted.
* RDS server accepted the import request.

**Logging into RDS to Validate the Migrated Data**

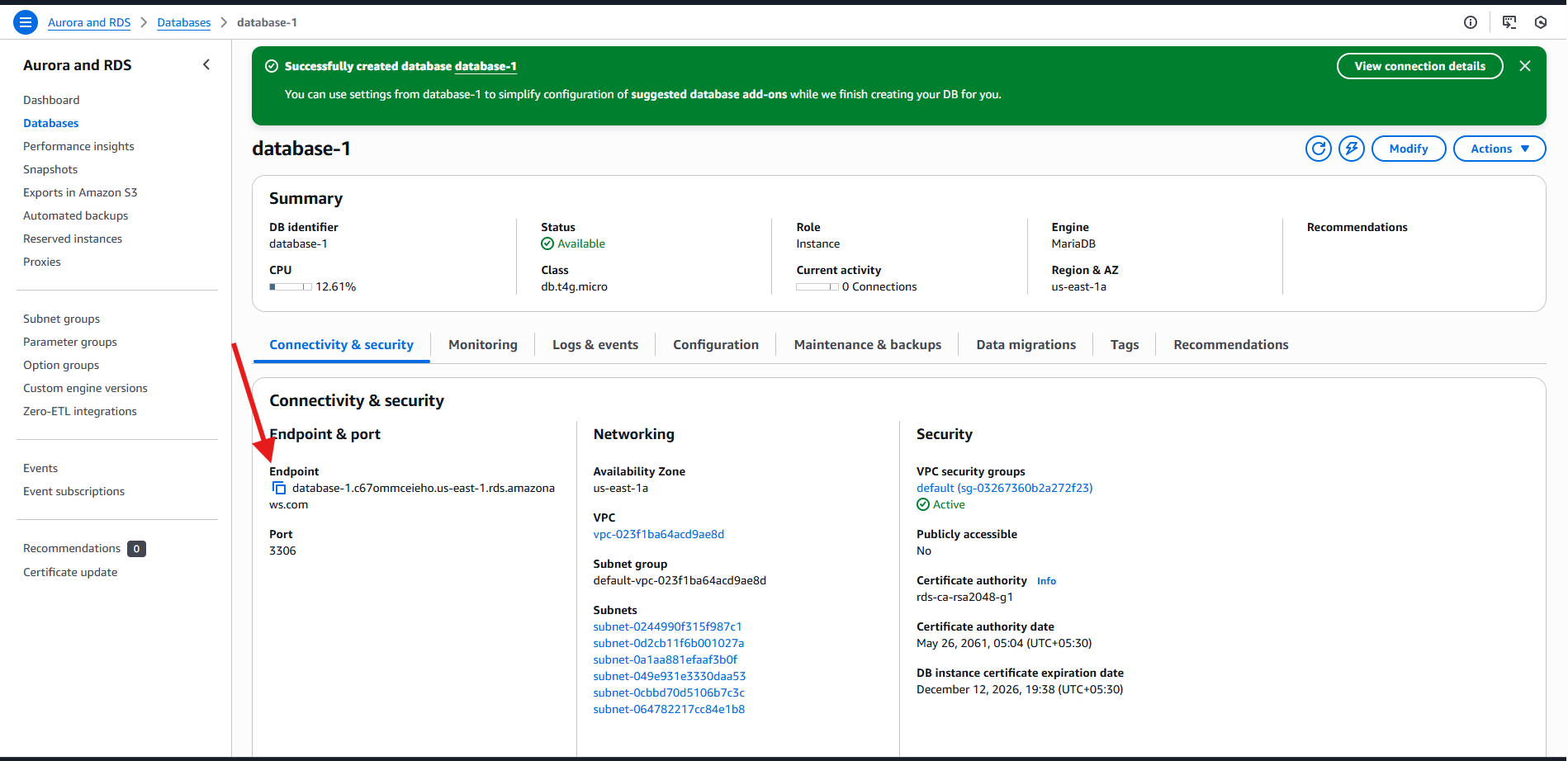
connected to RDS MariaDB using:

mysql -h database-2.c7comnecieho.us-east-1.rds.amazonaws.com -u admin -p

Inside RDS:

* Switched to the restored database:
* USE my\_ec2\_db;
* Verified tables exist:
* SHOW TABLES;
* Queried table contents:
* SELECT \* FROM table1;

The output shows the **same data previously stored in the EC2 MariaDB instance**, confirming successful migration.



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

1. Install MySQL DB on EC2.

**Installing the MySQL Repository Package**

downloaded and installed the official **MySQL Community repository** on the EC2 instance:

dnf install https://dev.mysql.com/get/mysql80-community-release-el9-1.noarch.rpm -y

This action:

* Added the **MySQL 8.0 Yum repository** to the EC2 instance.
* Enabled package access for MySQL Community Server and dependencies.

**Installing MySQL Community Server**

After enabling the MySQL repo, the user installed the MySQL Server package:

dnf install mysql-community-server -y

This triggered the installation of:

* mysql-community-server
* mysql-community-client
* mysql-community-common
* mysql-community-libs
* mysql-community-client-plugins

and replaced:

* mariadb-connector-c-config

This means the EC2 instance transitioned from **MariaDB-based client libraries** to **MySQL client libraries**.

**Starting the MySQL Service**

started the MySQL service using:

systemctl start mysqld

Followed by checking its status:

systemctl status mysqld

The output confirms:

* MySQL server (mysqld) is **active and running**
* It is using MariaDB service naming conventions (mariadb.service) due to EL9 compatibility  
  (AWS AL2023 identifies MariaDB as the default SQL engine)

A black screen with white text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

1. Launch MySQL RDS image.

**Choosing Database Creation Method & Engine**

creating a **new RDS database**, choosing:

* **Full configuration** (to manually configure all settings)
* **MySQL** engine (this time not MariaDB)
* Edition: **MySQL Community**

This indicates the user is now setting up a MySQL RDS instance (separate from the earlier MariaDB RDS instance).

**Configuring Master Credentials**

For the MySQL RDS instance, the user:

* Entered database-1 as the **DB instance identifier**
* Used the default master username → admin
* Selected **Self-managed** credentials
* Entered and confirmed a **strong master password**

This means:

* Secrets Manager is **not** being used to store the MySQL credentials
* will manually manage and remember the password

**Setting Up Connectivity for RDS**

selected:

* **Connect to an EC2 compute resource**
  + The selected EC2 instance: i-0477c29c5aa8bbf3 (same EC2 used earlier)

By selecting this option, AWS automatically:

* Configured the **VPC**
* Created/attached a **DB subnet group**
* Added a new **RDS security group** with inbound permission from the EC2 instance
* Ensured the EC2 instance can reach the RDS instance via port **3306**

This avoids the need for manually configuring SG rules.

**RDS MySQL Instance Successfully Created**

reached the final RDS dashboard page:

* DB instance identifier: **database-1**
* Engine: **MySQL Community**
* Class: **db.t4g.micro**
* Availability Zone: **us-east-1a**
* Status: **Available**

Under **Connectivity & Security**, the user observed:

* Endpoint URL
* Port: **3306**
* Subnet groups used
* Security groups attached
* Public accessibility: **No**  
  (meaning the DB is private and only accessible from EC2)

This confirms that the MySQL RDS instance is running and ready for use.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. Configure Multi-AZ.

* As I have free account I don’t have access to create multi-az

A screenshot of a computer

AI-generated content may be incorrect.

1. Take backup of DB and restore the DB.

**Triggering a Manual DB Snapshot from the RDS Console**

went to the AWS RDS console and selected the RDS instance:

* **database-2** (MariaDB engine)

From the **Actions** dropdown, the user chose:

**Take snapshot**

This action initiates the process of creating a **manual backup** of the RDS instance.

A green banner at the top confirms:

* EC2 connection between the DB and the instance (i-0477c729c3aab8bf3) was already set up earlier
* This is unrelated to snapshot creation, but AWS shows all system events at the top

**Configuring the Snapshot**

On the **Take DB Snapshot** page, the user selected:

* **Snapshot type:** DB instance
* **DB instance:** database-2
* **Snapshot name:** db-snapshot

The name follows AWS requirements:

* Lowercase
* No spaces
* No leading/trailing hyphens

Then clicked:

**Take snapshot**

**Viewing Snapshot Creation Status**

AWS redirected to the **Snapshots** section, showing details of the newly created snapshot:

Snapshot information includes:

* Snapshot name: **db-snapshot**
* Snapshot type: **manual**
* Instance source: **database-2**
* Engine: **MariaDB**
* Engine version: **11.4.8**
* Storage: **20 GB gp2**
* Port: **3306**
* Status: **Creating** (snapshot still in progress)

This confirms the snapshot creation request was successfully initiated

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. Create read replica.

**Opening the RDS Actions Menu**

selected the RDS instance **database-1** (MySQL Community Engine) from the list of databases.

From the **Actions** dropdown, the user clicked:

**Create read replica**

This opens the configuration page to set up a **read-only replica** of the primary RDS instance.

**Configuring the Read Replica**

On the Create read replica page, the user configured:

**Replica source**

* Selected the source database → **database-1**

**New replica identifier**

* Entered the name:
  + replica1

This will become the DB instance identifier of the read replica.

**Instance Class & Storage Configuration**

chose:

* DB instance class → **db.t4g.micro**
* Storage type → **General Purpose SSD (gp2)**
* Allocated storage → **20 GB**

These settings match the primary DB instance.

**Region Selection**

selected:

* Destination region → **US East (N. Virginia)**

This means the replica will be created **in the same region** as the source DB.

**Maintenance Configuration**

enabled:

**Auto minor version upgrade**

This allows AWS to automatically update the replica with minor engine patches.

Deletion protection remained unchecked.

**Account Limitation Error**

When attempted to finalize creation, AWS displayed:

**“You reached the maximum number of instances available with free plan. To remove all limitations, upgrade your account plan.”**

This indicates:

* The current AWS Free Tier account allows only **one free RDS instance**.
* The user already has **database-1** and **database-2** running.
* Read replicas count as additional RDS instances.
* AWS is blocking creation because free plan limits have been exceeded.

Therefore:

**The read replica could NOT be created due to Free Tier limitations**A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.