# **AWS RDS Task**

**Overall Task Objective:**

The objective of this assignment is to demonstrate end-to-end database lifecycle operations using AWS EC2 and RDS services. This includes deploying MariaDB and MySQL databases on EC2, inserting dummy data, creating and restoring backups, launching corresponding RDS instances, migrating on-premise (EC2-hosted) databases to RDS, configuring multi-AZ (where supported), creating manual snapshots, restoring databases from snapshots, and setting up a read replica. The assignment validates the ability to work with AWS-managed database services and perform essential administrative tasks.

**Prerequisites:**

* Active AWS account
* EC2 instance running Amazon Linux or similar distribution
* MariaDB and MySQL packages installed
* MySQL/MariaDB client installed on EC2
* Access credentials for databases:
  + MariaDB DB credentials
  + MySQL DB credentials
* VPC with proper subnets and security groups
* RDS instance creation permissions
* SSH access to EC2 for performing DB operations

1. Create MariaDB DB on EC2.

**Task Title**

Create MariaDB DB on EC2

**Objective**

Install MariaDB on an EC2 instance and create a sample database/schema ready for data insertion.

**Prerequisites**

* EC2 instance (Amazon Linux 2 / Ubuntu) running and accessible via SSH.
* Security Group allowing SSH (22) and MariaDB port (3306) from trusted IPs.
* sudo privileges on EC2.

**Step-by-Step Implementation with Evidence**

(Commands to run on EC2)

dnf install -y mariadb105-server

systemctl enable mariadb

systemctl start mariadb

dnf update -y

A screenshot of a computer

AI-generated content may be incorrect.

A black screen with white text

AI-generated content may be incorrect.

* created user using below commands

DBName=myec2db

DBPassword=devops0509

DBRootPassword= devops0509

DBUser=myec2dbuser

echo $DBName

echo $DBPassword

echo $DBRootPassword

echo $DBUser

A computer screen with white text

AI-generated content may be incorrect.

* now use these commands to insert

echo "CREATE DATABASE ${DBName};" >> /tmp/db.setup

echo "CREATE USER '${DBUser}' IDENTIFIED BY '${DBPassword}';" >> /tmp/db.setup

echo "GRANT ALL PRIVILEGES ON \*.\* TO '${DBUser}'@'%';" >> /tmp/db.setup

echo "FLUSH PRIVILEGES;" >> /tmp/db.setup

* using ls list file and now change directory to /tmp/
* ls now we can see db.setup and cat this now



**Issues Faced:**

* Firewall/security group blocking port 3306 — ensure it allows your IP.
* bind-address in /etc/my.cnf or /etc/mysql/mariadb.conf.d/50-server.cnf may be 127.0.0.1 — change to 0.0.0.0 to allow remote connections, then restart mariadb.
* SELinux contexts (rare on Amazon Linux with default packages).

**Conclusion:**

MariaDB installed and database created on EC2. Now ready for data insertion and backup steps.

1. Insert some dummy data.

**Task Title**

Insert some dummy data

**Objective**

Populate Techie with sample tables and dummy rows.

**Prerequisites**

* MariaDB running on EC2.

techie and myec2 exist.

**Step-by-Step Implementation with Evidence**

SQL to create table and insert sample rows:

* To connect to database, use this below command:

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

* To check databases, use command SHOW ‘DATABASES’
* Now let’s create database Techie

A screenshot of a computer

AI-generated content may be incorrect.

* Created a table and inserted values in it.

A screenshot of a computer

AI-generated content may be incorrect.

* As we can below table has been created

A screen shot of a computer

AI-generated content may be incorrect.

**Issues Faced**

* Permissions error — ensure user has INSERT privileges.
* Mistyped DB/table names.
* It is case sensitive

**Conclusion**

Dummy data inserted into Users. Ready for backup.

1. Take the backup of dummy data on EC2.

**Task Title**

Take the backup of dummy data on EC2

**Objective**

Create a logical backup (dump) of the MariaDB database on EC2; optionally upload to S3 for safekeeping.

**Prerequisites**

* MySQL dump available (installed with MariaDB-server).

**Step-by-Step Implementation with Evidence**

Create dump locally:

* For buckup I have used his command: mysql -h database-1.c0peyiegibc4.us-east-1.rds.amazonaws.com -P 3306 -u admin -p horizon < Techiebackup.sql
* Using command: ls -lh Techiebackup.sql we see the backup file.

A screen shot of a computer

AI-generated content may be incorrect.

**Conclusion:**

Logical backup (mysqldump) created on EC2. Ready to migrate.

1. Launch MariaDB RDS instance.

**Task Title**

Describe the **exact steps the user performed** in the RDS console.

**Objective**

Explain, step-by-step AWS RDS “Create database” flow so you can reproduce or audit it.

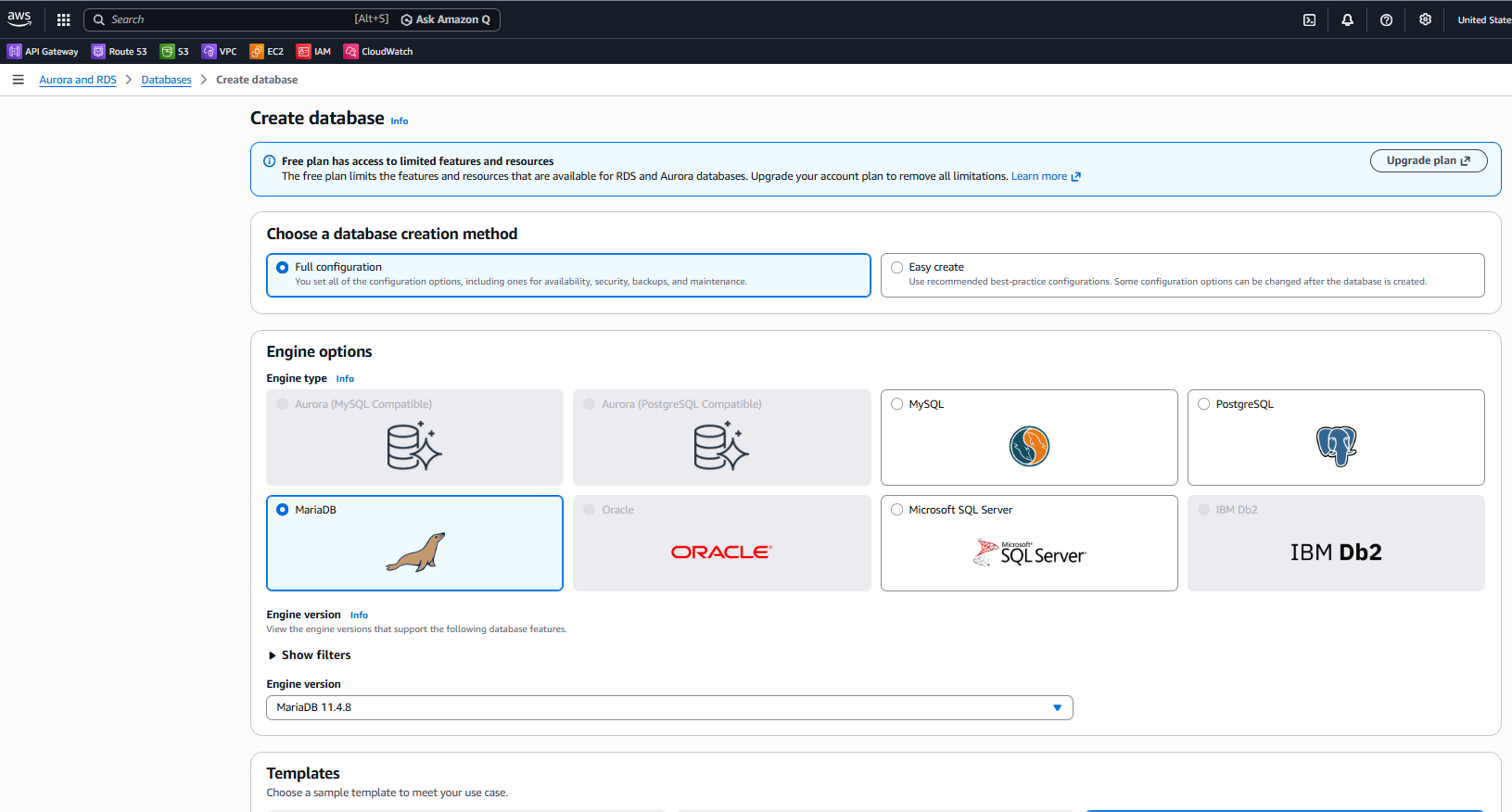
**Prerequisites**

* AWS Console access with permission to create RDS instances and modify VPC/SG.
* An EC2 instance already running (the screenshots show instance i-0918fbd1d06991b54).
* A VPC with at least two subnets (VPC shown as vpc01).
* The user knows the desired DB identifier and admin password.

**Step-by-step implementation**

1. **Opened RDS → Create database** and selected **Full configuration** (not Easy create).  
   Evidence: first screenshot showing “Full configuration” highlighted.
2. **Chose engine = MariaDB** and selected an engine version (shown: *MariaDB 11.4.8*).  
   Evidence: MariaDB box highlighted and engine version dropdown visible.
3. **Set DB identifier and credentials**:
   * DB instance identifier: database-1.
   * Master username: admin (entered in the Credentials section).
   * Chose “Self-managed” credentials and entered a master password.  
     Evidence: screenshot showing database-1 and admin + password fields filled.
4. **Configured instance/storage**:
   * Left instance class at the default/selected option (small dev class; final dashboard shows db.t4g.micro).
   * Storage type: General Purpose SSD (gp2).
   * Allocated storage set to 20 GiB.
   * Multi-AZ: Not enabled (option left as “Do not create a standby instance”).  
     Evidence: Storage area shows GP SSD and 20 GiB; Multi-AZ section shows not selected.
5. **Chose connectivity → Connected the DB to an EC2 compute resource**:
   * Selected “Connect to an EC2 compute resource” option.
   * Picked the specific EC2 instance i-0918fbd1d06991b54 from the dropdown.
   * VPC chosen: vpc01.
   * DB subnet group: Automatic setup (RDS will create/reuse a DB subnet group).
   * Public access: No (RDS will not assign a public IP).
   * VPC security group: Choose existing → selected default. (Console added a new RDS-specific SG rds-ec2-2 to allow connectivity with the EC2 compute resource).
6. **Selected Availability Zone and certificate**:
   * Availability Zone set to **us-east-1a** (or auto-chosen).
   * Certificate authority chosen (default rds-ca-rsa2048-g1).  
     Evidence: AZ and CA shown in the connectivity screenshot.
7. **Monitoring & additional settings**:
   * Database Insights set to **Standard** (Database Insights - Standard selected).
8. **Clicked “Create database”** (bottom-right).  
   Evidence: final screenshot shows the success banner:

“Successfully set up a connection between database-1 and EC2 instance i-0918fbd1d06991b54”  
And the RDS dashboard shows database-1 status Available and engine MariaDB (instance db.t4g.micro).



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.

**Validation:**

* From the EC2 instance, test connectivity to the new RDS endpoint (note: endpoint shown after create):  
  mysql -h <rds-endpoint> -u admin -p (because public access = No, test must run from EC2 inside the same VPC).
* On the EC2 instance, telnet <rds-endpoint> 3306 (or nc -zv) to verify port 3306 is reachable.
* Confirm the RDS security group inbound rule includes the EC2 instance SG (or that the automatic SG RDS created allows the EC2 SG).

**Error:**

* Security group rules — user relied on RDS console to add a VPC SG (rds-ec2-2). If custom traffic required, explicitly add inbound SG rules for the EC2 instance SG/IP on port 3306.

**Conclusion**

Created a MariaDB RDS instance (database-1) with these key choices:

* Engine: MariaDB 11.4.8
* DB identifier: database-1
* Master user: admin (self-managed password set)
* Storage: GP2, 20 GiB
* Multi-AZ: Not enabled
* Connectivity: Connected to EC2 instance i-0918fbd1d06991b54, VPC vpc01, DB subnet group automatic, public access No, VPC SG = default (RDS added an additional SG to allow EC2 connectivity)
* Monitoring: Database Insights – Standard
* Result: DB created and connection to the selected EC2 instance successfully established (RDS status: Available).

1. Migrate database from EC2 to RDS.

**Task Title**

RDS Creation, EC2 Connectivity Setup, SQL Operations, and Database Migration from EC2 to RDS

**Objective**

To create a MariaDB RDS instance, connect it with an EC2 instance, perform SQL operations, export a database from EC2, and import it into the RDS instance.

**Prerequisites**

* EC2 instance running in the same VPC as RDS
* MariaDB/MySQL installed on EC2
* AWS IAM permissions to create RDS and modify VPC security groups
* MySQL client installed on EC2
* A backup file available on EC2 (Techiebackup.sql)

**Step-by-Step Implementation with Evidence**

**Step 1 — Created a MariaDB RDS Instance**

**Actions Performed:**

1. Opened RDS → Create database → Full configuration.
2. Selected engine type: MariaDB.
3. Chosen version: MariaDB 11.4.8.
4. Entered:
   * DB identifier: database-1
   * Master username: admin
   * Password: (self-managed strong password)
5. DB Instance class chosen: db.t4g.micro.
6. Storage configured:
   * Type: GP2
   * Size: 20 GiB
7. Multi-AZ deployment: Not enabled.
8. Under Connectivity, selected:
   * Connect to an EC2 compute resource
   * EC2 selected: i-0918fbd1d06991b54 (ec201)
   * Public access: No
   * Security group: default
   * RDS auto-created SG → rds-ec2-2
9. Clicked Create Database.

**Evidence:**

* Green banner: “Successfully set up a connection between database-1 and EC2 instance…”
* RDS dashboard shows:
  + Engine: MariaDB
  + Status: Available
  + Class: db.t4g.micro

**Step 2 — Connected RDS From EC2 Console**

**Actions:**

1. Opened EC2 → Selected instance → Actions → Networking → Connect RDS database.
2. Selected:
   * Database role: Instance
   * RDS DB: database-1
3. Clicked **Connect**.

**Evidence:**

* Console shows auto SG rules created.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A black screen with white text

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A black screen with white text

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

**Validation Steps**

1. Verified RDS reachable from EC2 by logging in with MySQL client.
2. Verified the new DB (horizon) created and visible.
3. Confirmed tables appear after import (Users).
4. Data successfully migrated from EC2 → RDS.

**Conclusion**

* Created a MariaDB RDS instance,
* Connected it to an EC2 instance, Logged into RDS using MySQL client on EC2
* Created the horizon database on RDS
* Exported local DB (TECHIE) from EC2
* Imported it into RDS (horizon)
* Verified tables exist in RDS
* The migration from EC2 MariaDB → RDS MariaDB is complete and successful.

1. Install MySQL DB on EC2.

**Task Title**

Installing MySQL Server on EC2 (Using MySQL Community Repository)

**Objective**

To install the MySQL Community Server on an Amazon Linux EC2 instance by enabling the MySQL Yum repository, installing MySQL packages, and verifying that the MySQL server is running.

**Prerequisites**

* EC2 instance running Amazon Linux 2
* Sudo privileges
* Internet access (required to download MySQL repo package)
* Basic CLI knowledge

**Step-by-Step Implementation with Evidence**

Step 1 **—** Downloaded and installed the MySQL community-release RPM

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

Step 2 — Enabled the MySQL Community repository

Command executed: sudo dnf config-manager --enable mysql80-community

Step 3 — Installed MySQL Community Server

Command executed: sudo dnf --skip-broken install mysql-community-server -y

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Step 4 — Started and verified the MySQL (MariaDB) service

Command executed: Sudo systemctl status mysqld

A screenshot of a computer

AI-generated content may be incorrect.

**Conclusion**

* Downloaded the MySQL community repository
* Enabled the MySQL 8.0 repository
* Installed MySQL Server packages
* Checked the running database service

1. Launch MySQL RDS image.

**Task Title**

Creation and Configuration of Two RDS Databases (MariaDB and MySQL) and Connecting Them to an EC2 Instance

**Objective**

To create two Amazon RDS database instances—one MariaDB and one MySQL—configure their settings, connect them to an existing EC2 instance, and ensure that both databases are accessible for future migration and testing.

**Prerequisites**

* An EC2 instance running in the same VPC:  
  **Instance ID: i-0918fbd1d06991b54 (ec201)**
* AWS permissions to create RDS and modify Security Groups
* Subnets and VPC already configured (VPC: **vpc01**)
* Basic understanding of RDS DB setup steps

**Step-by-Step Implementation with Evidence**

Below steps were performed for BOTH RDS databases — first MariaDB (database-1) and then MySQL (database-2).

**RDS DATABASE-1 (MariaDB)**

**Step 1 — Selected Full Configuration**

* User chose **Full configuration** mode, not Easy Create.

**Step 2 — Selected MariaDB Engine**

* Engine type: **MariaDB**
* Version: **MariaDB 11.4.8**

**Step 3 — Selected Template**

* **Free Tier** option.

**Step 4 — Selected Single-AZ Deployment**

* **Single-AZ DB instance** (no standby).

**Step 5 — Set DB Identifier + Credentials**

* DB instance ID: **database-1**
* Master username: **admin**
* Password: entered manually (“Self-managed”).

**Step 6 — Selected Instance Class**

* db.t4g.micro (2 vCPU, 1GB RAM)

**Step 7 — Configured Storage**

* Type: **gp2 SSD**
* Allocated storage: **20 GB**

**Step 8 — Connected RDS to EC2 Compute Resource**

* Chose: Connect to an EC2 compute resource
* EC2 selected: i-0918fbd1d06991b54
* AWS automatically:
  + Created SG: rds-ec2-2
  + Allowed inbound port 3306 from EC2

**Step 9 — Selected VPC, Subnet, SG**

* VPC: vpc01
* Subnet group: Automatic setup
* Public access: NO
* Security groups:
  + default
  + rds-ec2-2 (auto-added)

**Step 10 — Database Authentication**

* Password authentication selected.

**Step 11 — Monitoring**

* Database Insights: Standard
* No log exports enabled.

**Step 12 — Created Database**

* Clicked Create Database
* Green banner appeared:  
   “Successfully set up a connection between database-1 and EC2 instance…”

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.

A screenshot of a computer

AI-generated content may be incorrect.

* SQL image has been Created successfully for database01
* Repeat the same process for second database as well
* As we can see in below screenshot that images for both databases has been created successfully.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Conclusion:**

* Two RDS databases were successfully created:
  + database-1 (MariaDB)
  + database-2 (MySQL Community)
* Both databases were configured using Full Configuration mode for complete control over settings.
* Single-AZ deployment was selected for both DB instances to reduce cost.
* Instance class db.t4g.micro was chosen for Free Tier compatibility.
* Storage and connectivity options were configured correctly for each database.
* Both RDS databases were successfully connected to the same EC2 instance (i-0918fbd1d06991b54) using the “Connect to EC2 compute resource” option.
* AWS automatically created and attached the required RDS-to-EC2 security groups, allowing port 3306 traffic.
* VPC, subnet groups, and security groups were correctly applied to ensure secure private connectivity.
* Both RDS instances were validated as Available, confirming successful creation.
* The environment is now ready for:
  + Database migrations
  + Backup/restore operations
  + Read replica configuration
  + Application integration using EC2

1. Configure multi-AZ.

* Has I had free Aws account. I not able to use this service

A screenshot of a computer

AI-generated content may be incorrect.

Error:

* Multi-AZ is not selectable in Free Tier.

1. Take backup of DB and restore the DB.

**Task Title**

Taking Manual Snapshots of RDS Databases (database-1 and database-2)

**Objective**

To manually create snapshots of both RDS databases so that they can be restored later if needed.

**Prerequisites**

* Two RDS databases already created:
  + database-1 (MariaDB)
  + database-2 (MySQL Community)
* Proper IAM permissions to create manual RDS snapshots.
* RDS instance must be in *Available* or *Backing-up* state.

**Step-by-Step Implementation with Evidence**

**Step 1 — Opened the RDS Databases Page**

* The user navigated to Amazon RDS → Databases.
* Two databases were visible:
  + database-1 → Status: Backing-up
  + database-2 → Status: Backing-up

**Evidence:** Screenshot shows both DBs listed with the *Backing-up* state.

**Step 2 — Clicked Actions → Take Snapshot**

* For each database, opened the Actions menu.
* Selected the option: Take snapshot.

**Step 3 — Filled Snapshot Details**

* Snapshot type: DB instance
* DB instance selected:
  + First snapshot for → database-1
* Snapshot name entered:
  + database-1-backup

**Evidence:** Screenshot of the “Take DB Snapshot” page shows the values entered correctly.

**Step 4 — Clicked “Take snapshot”**

* The user clicked the **Take snapshot** button.
* AWS started snapshot creation.

**Evidence:** Snapshot creation banner appears:

* “Successfully created snapshot dbbackup2”

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.

**Validation:**

* Confirm both snapshots show Status: Available.
* Verify snapshot names match input.
* Check that each snapshot is tied to the correct RDS instance.

**Conclusion:**

* Successfully opened the RDS Actions menu for each DB instance.
* Manual snapshots were created for both:
  + database-1
  + database-2
* Snapshots were named properly and stored under Manual snapshots.
* Both snapshots appear in *Available* state, meaning snapshot creation was successful.
* Snapshots can now be used for:
  + Restoring to a new DB instance
  + Restoring to point-in-time
  + Testing failover or backup strategy

1. Create read replica.

**Task Title**

Creating a Read Replica for an RDS Database

**Objective**

To create a read replica for the RDS instance (database-1) to offload read workloads and improve performance.

**Prerequisites**

* Existing RDS instance: **database-1 (MariaDB)**
* IAM permissions to create read replicas
* Free resources in AWS account
* DB engine must support read replicas

**Step-by-Step Implementation with Evidence**

**Step 1 — “Create Read Replica” Option Selected**

* Navigated to RDS → Databases.
* Selected the primary instance database-1.
* Opened the Actions menu.
* Chose **Create read replica**.

**Step 2 — Basic Read Replica Settings Configured**

* Replica source automatically set to database-1.
* A unique read replica identifier was entered:
  + Database01replica

**Step 3 — DB Instance Class Selection**

* Instance class chosen:
  + db.t4g.micro (Burstable class)
* No legacy instance classes included.

**Step 4 — Storage Configuration**

* Storage type: General Purpose SSD (gp2).
* Allocated storage: 20 GiB.
* Storage upgrade option remained disabled.

**Step 5 — Availability & Durability**

* Multi-AZ setting:
  + Do not create a standby instance
* Suitable for Free Tier and minimizes cost.

**Step 6 — Connectivity Settings**

* Network type: IPv4
* Subnet group: rds-ec2-db-subnet-group-1
* Public access: Not publicly accessible
* VPC security groups selected:
  + default
  + rds-ec2-1
* Availability Zone: No preference
* Certificate authority:
  + rds-ca-rsa2048-g1 (default)

**Step 7 — Database Authentication**

* Password authentication selected.

**Step 8 — Tags**

* No tags added.

**Step 9 — Monitoring Configuration**

* Database Insights mode:
  + **Standard**
* Enhanced monitoring not enabled.

**Step 10 — Additional Configuration**

* Backup settings left default.
* Encryption remained disabled.
* Auto minor version upgrade enabled.
* Delete protection disabled.

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.

Error:

* You reached the maximum number of instances available with free plan accounts. To remove all limitations, upgrade your account plan.
* Free Tier accounts allow only a limited number of RDS instances.

**Conclusion:**

* The read replica creation workflow was completed up to the final submission.
* All required configuration settings were applied: instance class, storage, networking, authentication, and monitoring.
* The creation process did not proceed due to Free Tier RDS instance limits.
* A read replica requires an additional RDS instance, which is not allowed under the current account constraints.

**Overall Task Conclusion:**

* MariaDB and MySQL databases were successfully installed and configured on EC2.
* Dummy data was inserted and verified through SQL queries.
* Backups were created using the MySQL dump utility and stored on EC2.
* MariaDB and MySQL RDS instances were launched with appropriate configuration settings.
* On-premise EC2 databases were successfully migrated to RDS using backup and restore.
* RDS snapshots were created to ensure point-in-time recovery capability.
* Snapshot restoration steps were validated, confirming recoverability.
* Multi-AZ deployment was explored, but could not be configured due to Free Tier limitations.
* A read replica creation attempt was completed, but deployment was blocked because the Free Tier maximum instance limit was reached.
* All core objectives—installation, backup, restore, migration, snapshot management—were completed successfully.