



AWS RDS

**Relational Database
Service**

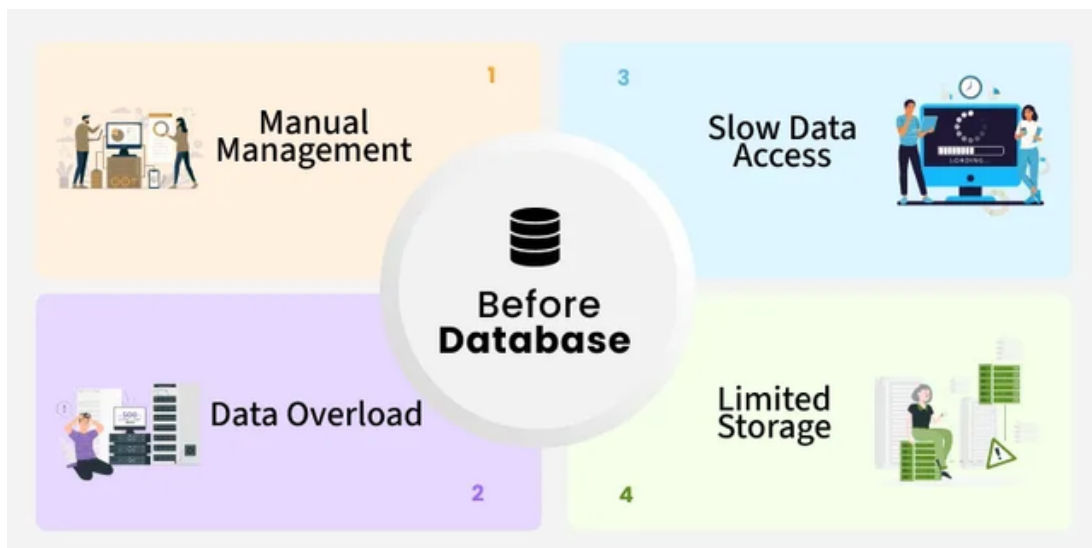


Concept Overview:

What is Database	1
Relational Database	2
Non-Relational Database	3
Introduction RDS	4
RDS Features	5
Work process of RDS	6
Hands-on with (RDS + MySQL + EC2)	7

What is Database:

A database is an electronically stored, systematic collection of data. It can contain any type of data, including words, numbers, images, videos, and files. You can use software called a database management system (DBMS) to store, retrieve, and edit data. In computer systems, the word database can also refer to any DBMS, to the database system, or to an application associated with the database.



Relational Database:

Relational database is a database that stores data in tables (rows & columns) with relationships between them.

- Data organized into tables
- Each table has rows (records) & columns (fields)
- Uses SQL (Structured Query Language) for queries
- Supports relationships (Primary Key, Foreign Key)
- Ensures data integrity & consistency

Relational databases:

- MySQL
- PostgreSQL
- Oracle Database
- Microsoft SQL Server
- MariaDB
- IBM Db2
- Amazon Aurora
- SQLite

Non-Relational Database:

Non-relational database is a database that stores data in flexible formats (not only tables), designed for scalability

- Data stored as documents, key-value pairs, graphs, or wide-columns
- No fixed schema (schema-less or flexible schema)
- Optimized for scalability & high performance
- Great for big data & real-time applications

Non-Relational databases:

- MongoDB
- Cassandra
- Redis
- DynamoDB

About RDS:

Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the AWS Cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks.

Benefits of it:

Amazon RDS provides the following principal advantages over database deployments that aren't fully managed:

- You can use database engines that you are already familiar with: IBM Db2, MariaDB, Microsoft SQL Server, MySQL, Oracle Database, and PostgreSQL.
- Amazon RDS manages backups, software patching, automatic failure detection, and recovery.
- You can turn on automated backups, or manually create your own backup snapshots. You can use these backups to restore a database. The Amazon RDS restore process works reliably and efficiently.
- You can get high availability with a primary DB instance and a synchronous secondary DB instance that you can fail over to when problems occur. You can also use read replicas to increase read scaling.
- In addition to the security in your database package, you can control access by using AWS Identity and Access Management (IAM) to define users and permissions. You can also help protect your databases by putting them in a virtual private cloud (VPC).

Features of RDS:

Availability:

- Automated Backups → easy recovery, quick access
- Database Snapshots → user-driven, shareable across AWS accounts

Security:

- Strong password (Admin role by default)
- Encryption with AWS KMS keys

Backups:

- Automated Backups (configured at creation)
- Snapshots (non-editable, for record keeping)

Scalability:

- Horizontal Scaling → add multiple instances to handle high traffic
- Vertical Scaling → upgrade existing resources (CPU, storage)

Performance:

- General Purpose SSD (cost-effective, broad use)
- Provisioned SSD (higher performance needs)

Pricing:

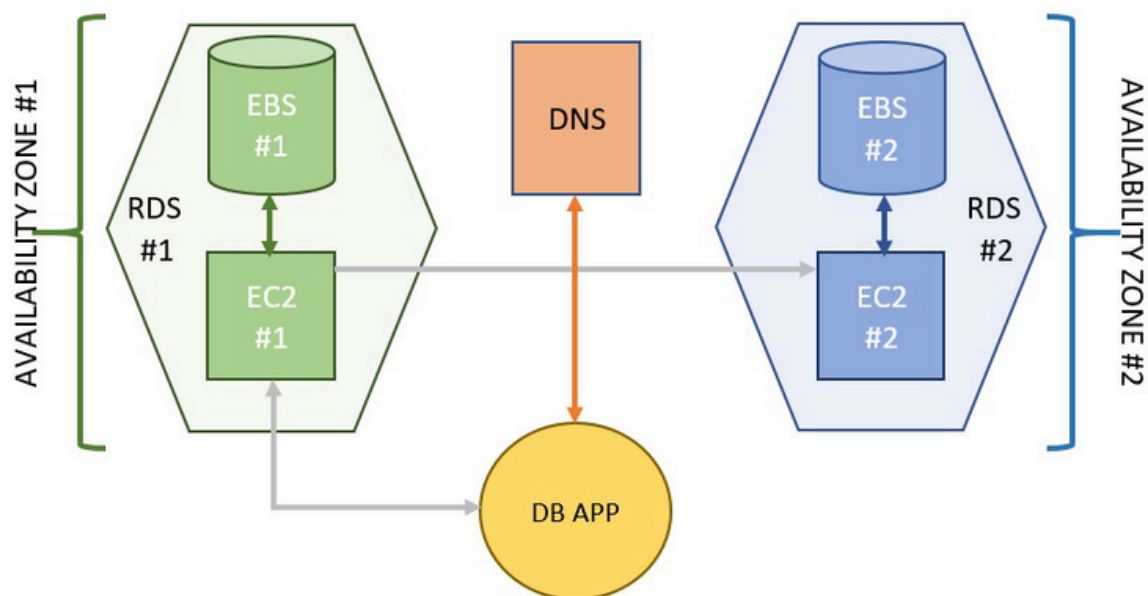
- Pay-as-you-go model
- No minimum charge, free tier available with limits

Work process of RDS:

Traditionally, database management was scattered across multiple layers, the web server, the application server, and finally the database, requiring a dedicated team for maintenance. To simplify this, AWS introduced RDS, an all-in-one managed database service.

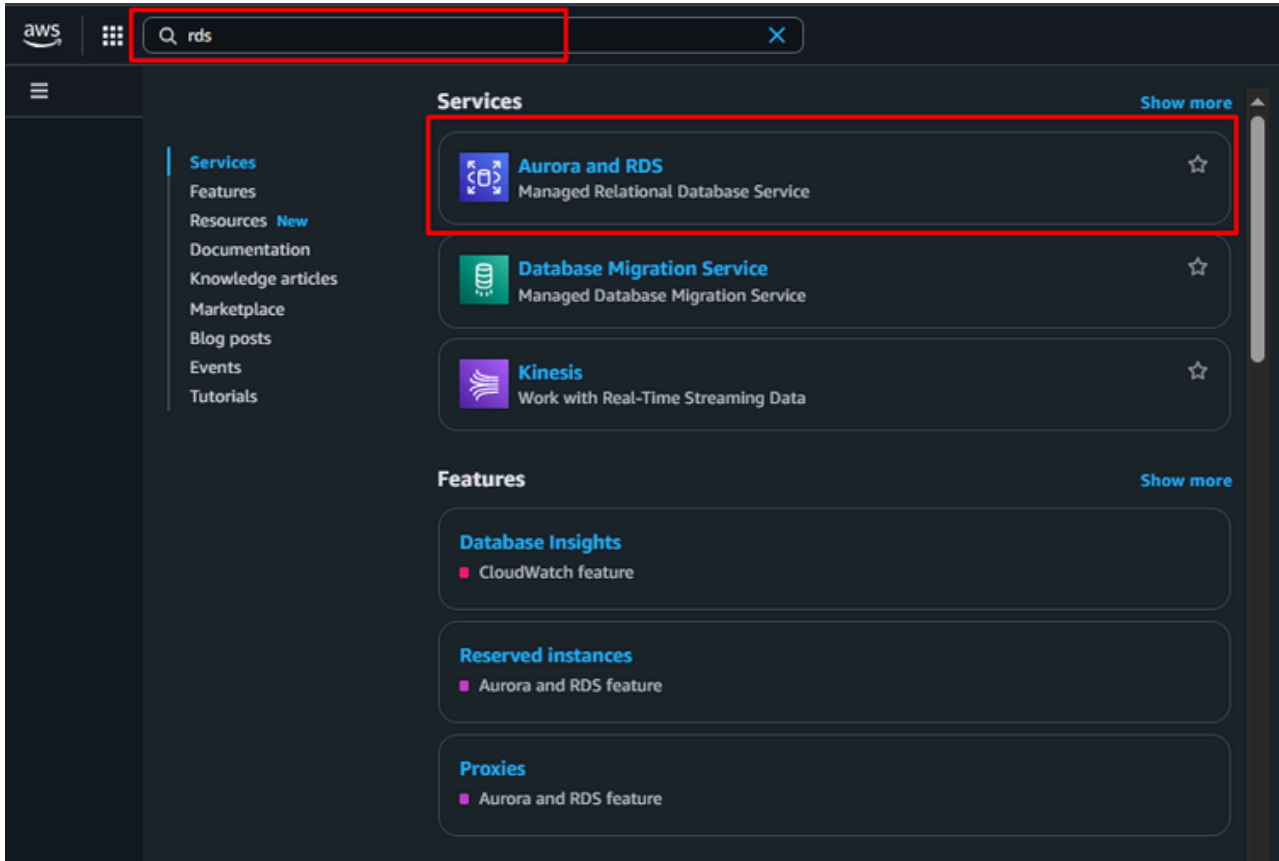
RDS integrates all components of traditional database architecture, including compute (EC2) and networking (DNS), into a single managed solution. Each part of the RDS architecture comes with its own set of features, making database management easier, more efficient, and highly scalable.

A diagram illustrating the RDS architecture is provided below.

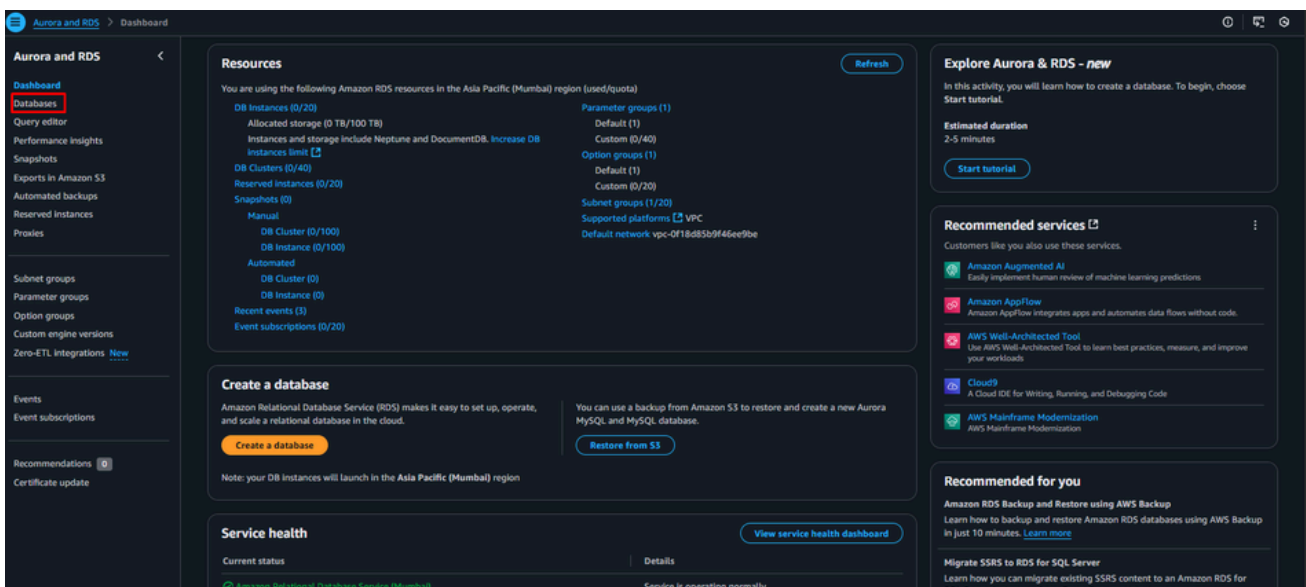


Hands-on with (RDS + MySQL + EC2):

Step 1: Sign in to your AWS account and search for RDS.

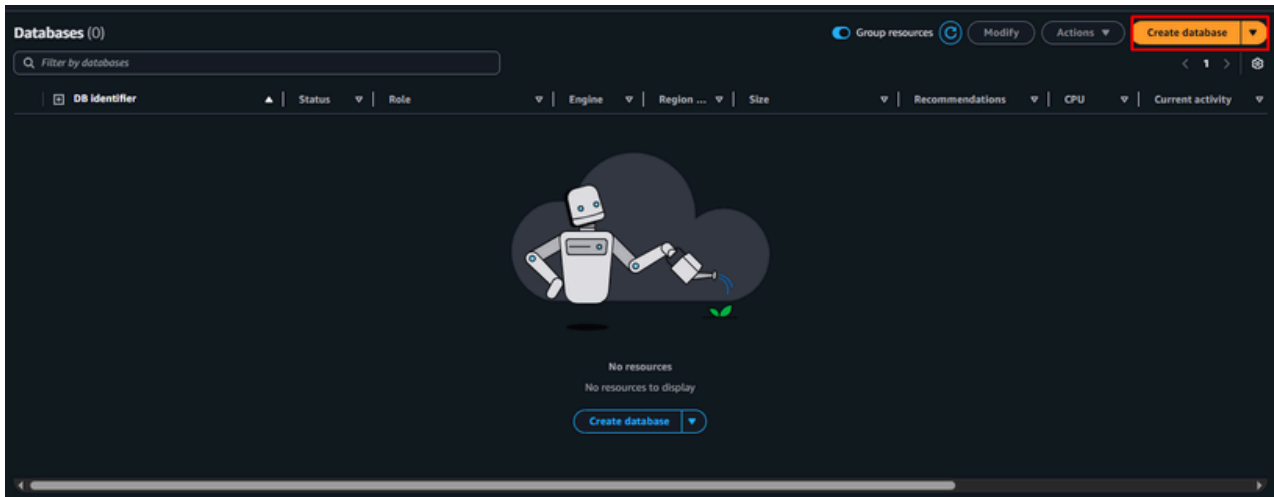


Get this page now click on databases section.

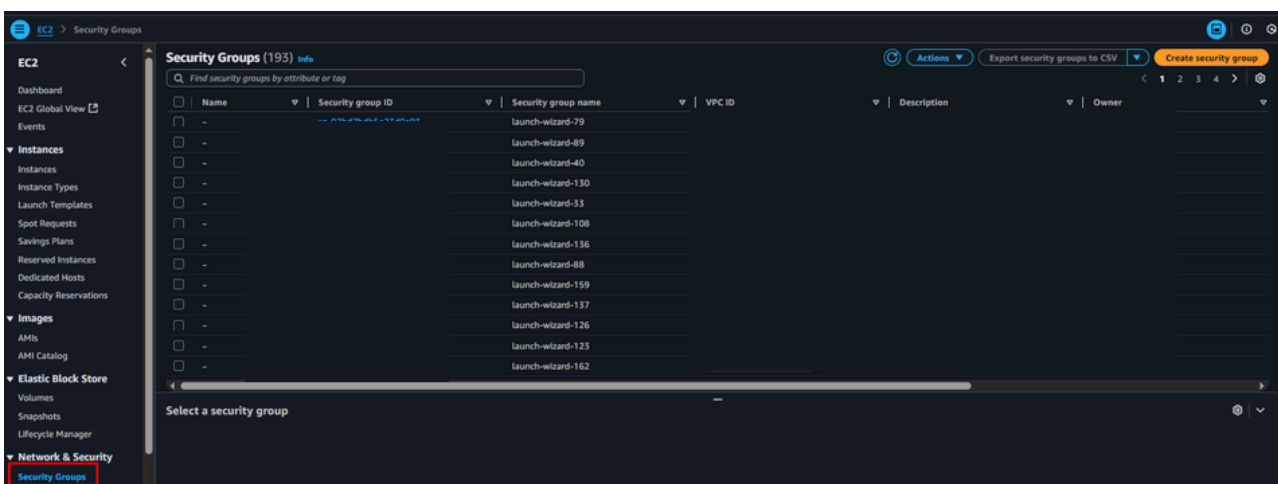
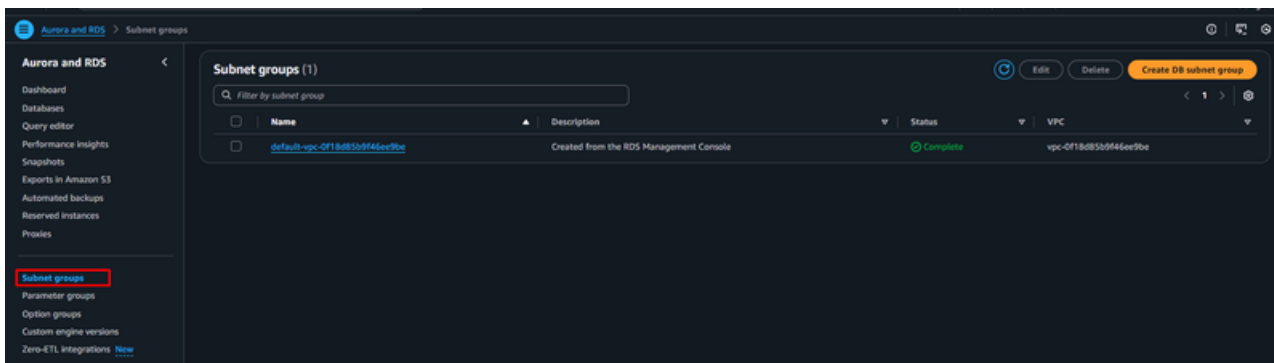


Hands-on with (RDS + MySQL + EC2):

Step 2: After click on database section get this page. Now click on create database button.

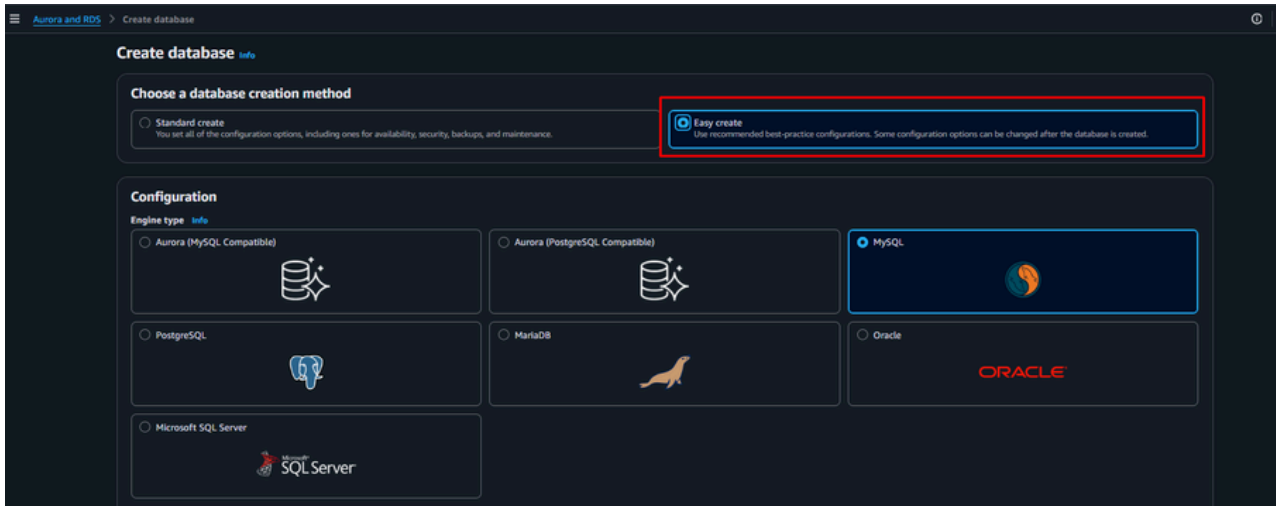


Note: Before creating a new database, you need to create a subnet group and a security group.

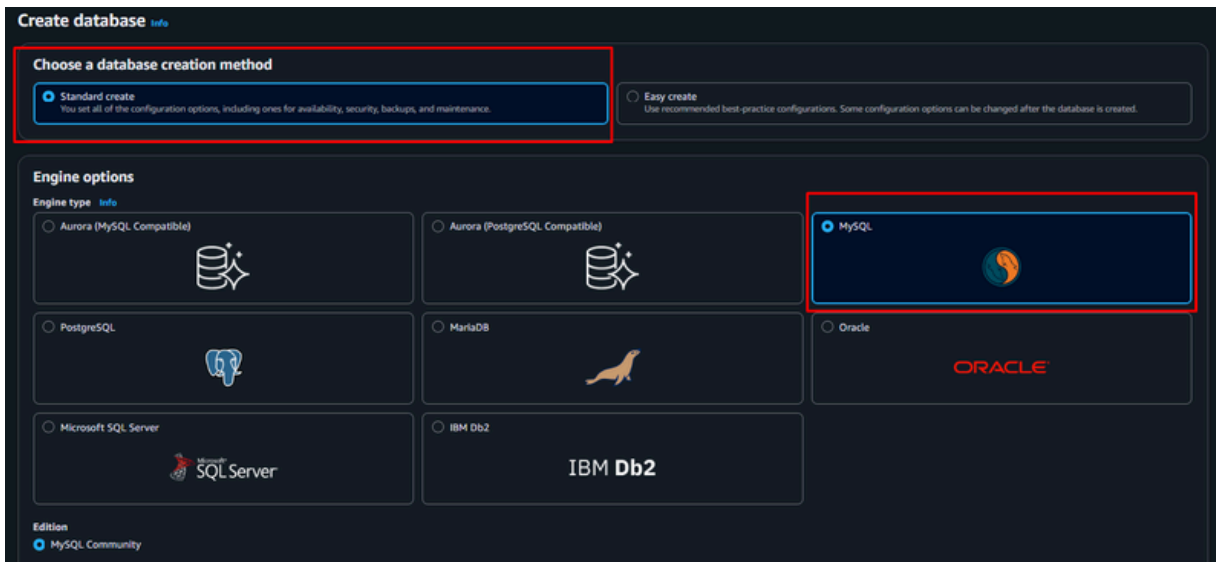


Hands-on with (RDS + MySQL + EC2):

Step 3: You can create a database easily with one click, but I chose to create it manually with detailed configuration.

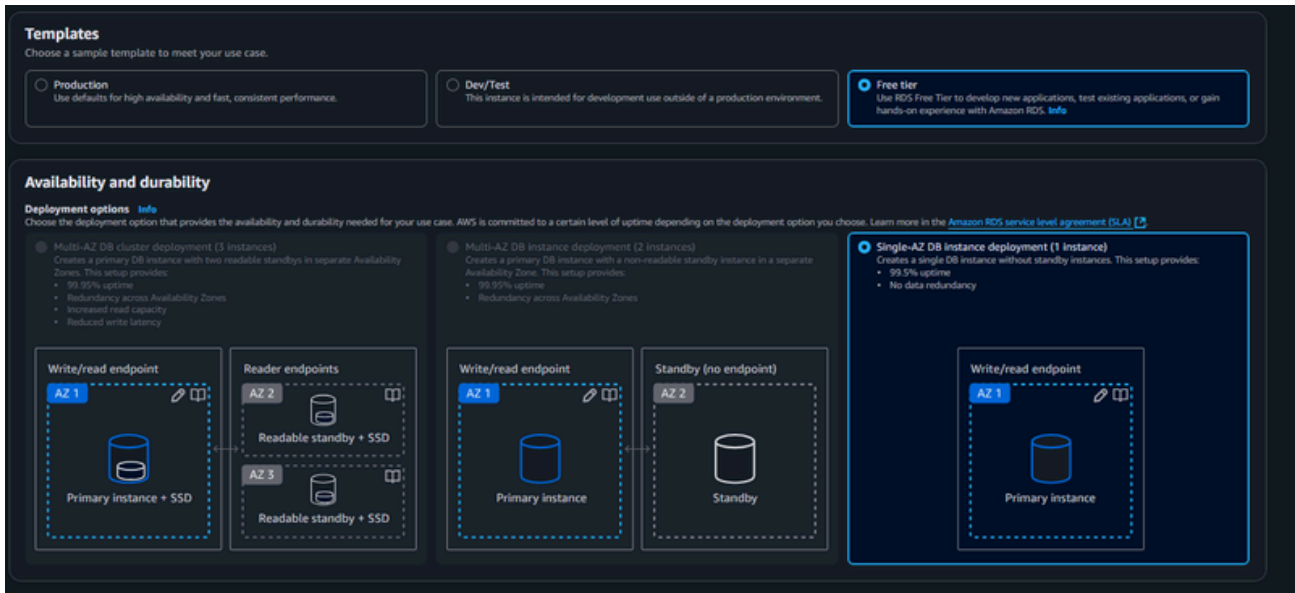


Step 4: Let's create db manually. Choose standard create and database which one you want to use.



Hands-on with (RDS + MySQL + EC2):

In the Template section, I chose the Free Tier option, so it hides paid resources for more configuration.

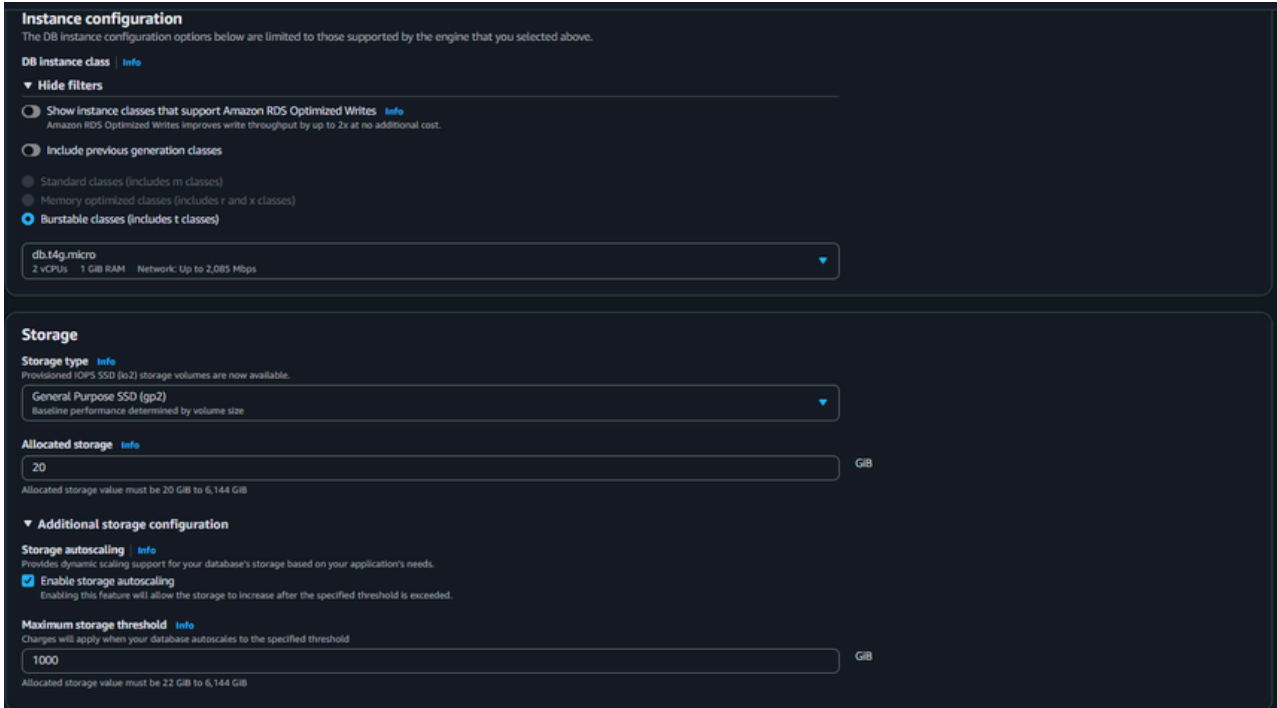


In settings section, type db name, master username and set password for access it later.



Hands-on with (RDS + MySQL + EC2):

In the Instance configuration & Storage section, I keep all as it is, if you need any changes you can do.



Instance configuration
The DB instance configuration options below are limited to those supported by the engine that you selected above.

DB instance class [Info](#)

▼ Hide filters

☐ Show instance classes that support Amazon RDS Optimized Writes [Info](#)
Amazon RDS Optimized Writes improves write throughput by up to 2x at no additional cost.

☐ Include previous generation classes

☐ Standard classes (includes m classes)

☐ Memory optimized classes (includes r and x classes)

☒ Burstable classes (includes t classes)

db.t4g.micro
2 vCPUs 1 GiB RAM Network: Up to 2,085 Mbps

Storage

Storage type [Info](#)
Provisioned IOPS SSD (io2) storage volumes are now available.

General Purpose SSD (gp2)
Baseline performance determined by volume size

Allocated storage [Info](#)
20 GiB
Allocated storage value must be 20 GiB to 6,144 GiB

▼ Additional storage configuration

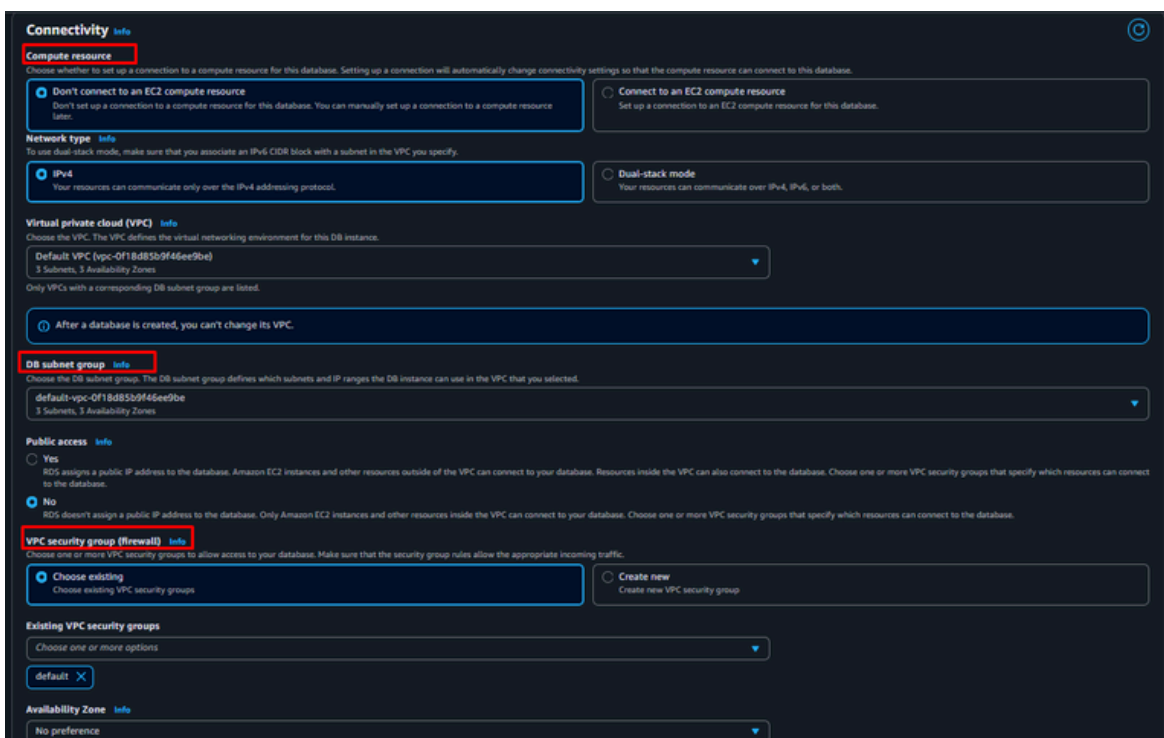
Storage autoscaling [Info](#)
Provides dynamic scaling support for your database's storage based on your application's needs.

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

Maximum storage threshold [Info](#)
Charges will apply when your database autoscales to the specified threshold.

1000 GiB
Allocated storage value must be 22 GiB to 6,144 GiB

In the Connectivity section, configure your subnet groups and security groups, SG will be the same for EC2 instance.



Connectivity [Info](#)

Compute resource

Choose whether to set up a connection to a compute resource for this database. Setting up a connection will automatically change connectivity settings so that the compute resource can connect to this database.

☒ Don't connect to an EC2 compute resource
Don't set up a connection to a compute resource for this database. You can manually set up a connection to a compute resource later.

☐ Connect to an EC2 compute resource
Set up a connection to an EC2 compute resource for this database.

Network type [Info](#)
To use dual-stack mode, make sure that you associate an IPv6 CIDR block with a subnet in the VPC you specify.

☒ IPv4
Your resources can communicate only over the IPv4 addressing protocol.

☐ Dual-stack mode
Your resources can communicate over IPv4, IPv6, or both.

Virtual private cloud (VPC) [Info](#)
Choose the VPC. The VPC defines the virtual networking environment for this DB instance.

Default VPC (vpc-0f18d85b9f46ee9be)
3 Subnets, 3 Availability Zones

Only VPCs with a corresponding DB subnet group are listed.

ⓘ After a database is created, you can't change its VPC.

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

default-vpc-0f18d85b9f46ee9be
3 Subnets, 3 Availability Zones

Public access [Info](#)

☐ Yes
RDS assigns a public IP address to the database. Amazon EC2 instances and other resources outside of the VPC can connect to your database. Resources inside the VPC can also connect to the database. Choose one or more VPC security groups that specify which resources can connect to the database.

☒ No
RDS doesn't assign a public IP address to the database. Only Amazon EC2 instances and other resources inside the VPC can connect to your database. Choose one or more VPC security groups that specify which resources can connect to the database.

VPC security group (firewall) [Info](#)
Choose one or more VPC security groups to allow access to your database. Make sure that the security group rules allow the appropriate incoming traffic.

☒ Choose existing
Choose existing VPC security groups

☐ Create new
Create new VPC security group

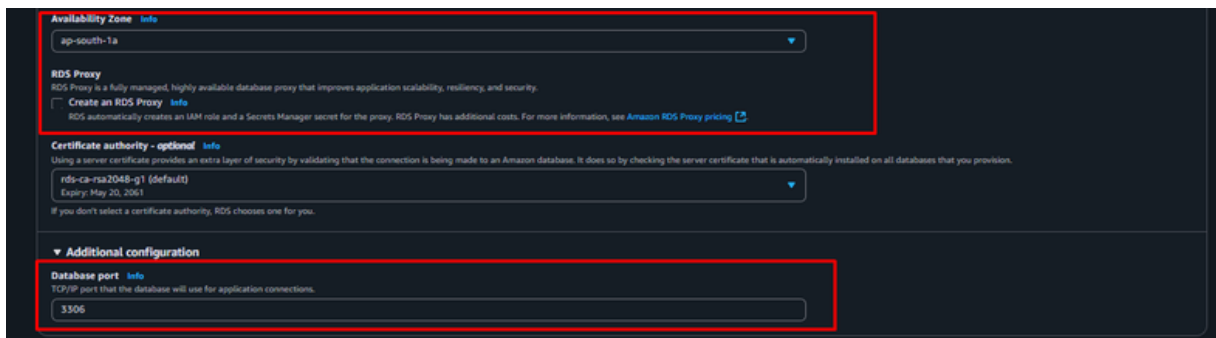
Existing VPC security groups
Choose one or more options

default X

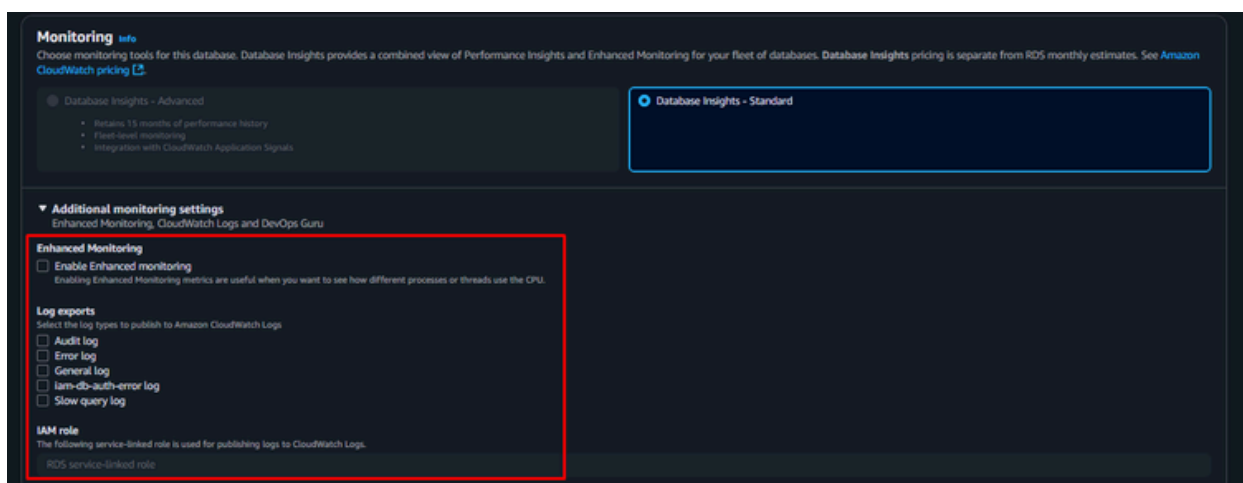
Availability Zone [Info](#)
No preference

Hands-on with (RDS + MySQL + EC2):

Don't forget to choose AZ. If you need you can set a custom port for your MySQL connection.



In the Monitoring section, you can choose the logs option, i keep it as it is.



In the Additional configuration, you can choose the backup, maintance and maintainece window option. I keep it default i don't need an backup.

Hands-on with (RDS + MySQL + EC2):

In the Additional Configuration, you can choose backup, maintenance, and maintenance window options. I kept it default as I don't need a backup.

Additional configuration
Database options, encryption turned off, backup turned off, backtrack turned off, maintenance, CloudWatch Logs, delete protection turned off.

Database options

Initial database name [Info](#)

If you do not specify a database name, Amazon RDS does not create a database.

DB parameter group [Info](#)

default:mysql8.0

Option group [Info](#)

default:mysql-8-0

Backup

☐ Enable automated backups
Creates a point-in-time snapshot of your database.

☐ Enable encryption
Choose to encrypt the given instance. Master key IDs and aliases appear in the list after they have been created using the AWS Key Management Service console. [Info](#)

Maintenance

Auto minor version upgrade [Info](#)

☐ Enable auto minor version upgrade
Enabling auto minor version upgrade will automatically upgrade your database minor version. For limitations and more details, see [Automatically upgrading the minor engine version documentation](#).

Maintenance window [Info](#)

Select the period you want pending modifications or maintenance applied to the database by Amazon RDS.

☐ Choose a window

☒ No preference

☒ Enable deletion protection
Protects the database from being deleted accidentally. While this option is enabled, you can't delete the database.

One important thing: enable deletion protection by checking this option. Your database won't be deleted when it is selected.

Estimated monthly costs

The Amazon RDS Free Tier is available to you for 12 months. Each calendar month, the free tier will allow you to use the Amazon RDS resources listed below for free:

- 750 hrs of Amazon RDS in a Single-AZ db.t2.micro, db.t3.micro or db.t4g.micro instance.
- 20 GB of General Purpose Storage (SSD).
- 20 GB for automated backup storage and any user-initiated DB Snapshots.

[Learn more about AWS Free Tier.](#)

When your free usage expires or if your application use exceeds the free usage tiers, you simply pay standard, pay-as-you-go service rates as described in the [Amazon RDS Pricing page](#).

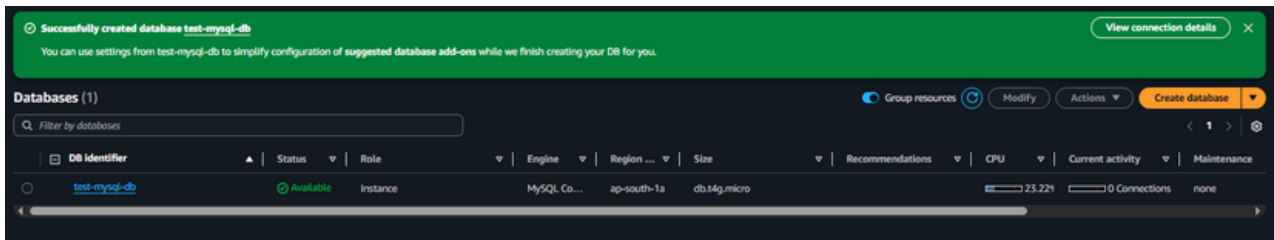
You are responsible for ensuring that you have all of the necessary rights for any third-party products or services that you use with AWS services.

[Cancel](#) [Create database](#)

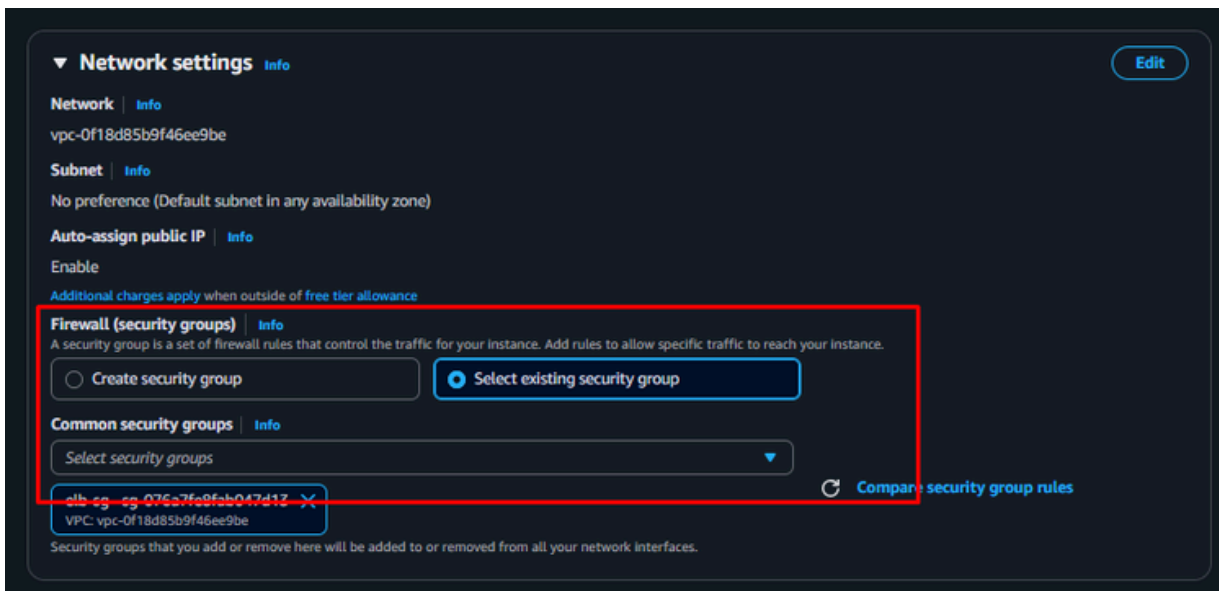
Now, click on create database button and create it.

Hands-on with (RDS + MySQL + EC2):

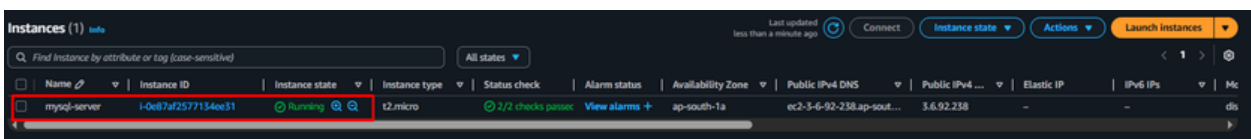
Database created successfully.



Step 5: Now launch an EC2 instance with the same Security Group that is used in the database.



Here is my EC2 instance:



Hands-on with (RDS + MySQL + EC2):

Step 6: Now it's time to connect our database with the EC2 instance. Before doing this, I provide a command-line file that can be used to connect to and test the RDS from EC2.

```
# update packages
sudo apt update && sudo apt upgrade -y

# install mysql
sudo apt install mariadb-client

# test
mysql --version

# retrieved endpoint, username, and password to connect
mysql -h <RDS_Endpoint> -P 3306 -u <RDS_Username> -p

# after connect show db lists
SHOW DATABASES;

# create new database
CREATE DATABASE dbname;

# switch to your db
USE dbname;

# create user table
CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  email VARCHAR(100) UNIQUE NOT NULL
);

# insert sample data
INSERT INTO users (name, email)
VALUES
('Alamgir Hosen', 'alamgir@example.com'),
('Abir Ahmed', 'abir@example.com');

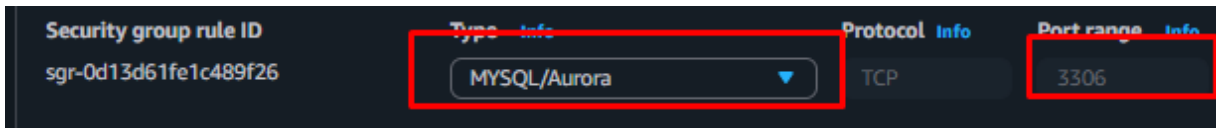
# see table lists
SHOW TABLES;

# show data from specific table
SELECT * FROM users;

# leave from db
exit;
```

Hands-on with (RDS + MySQL + EC2):

Don't forget to set inbound roles of Security Groups for your MySql connection.

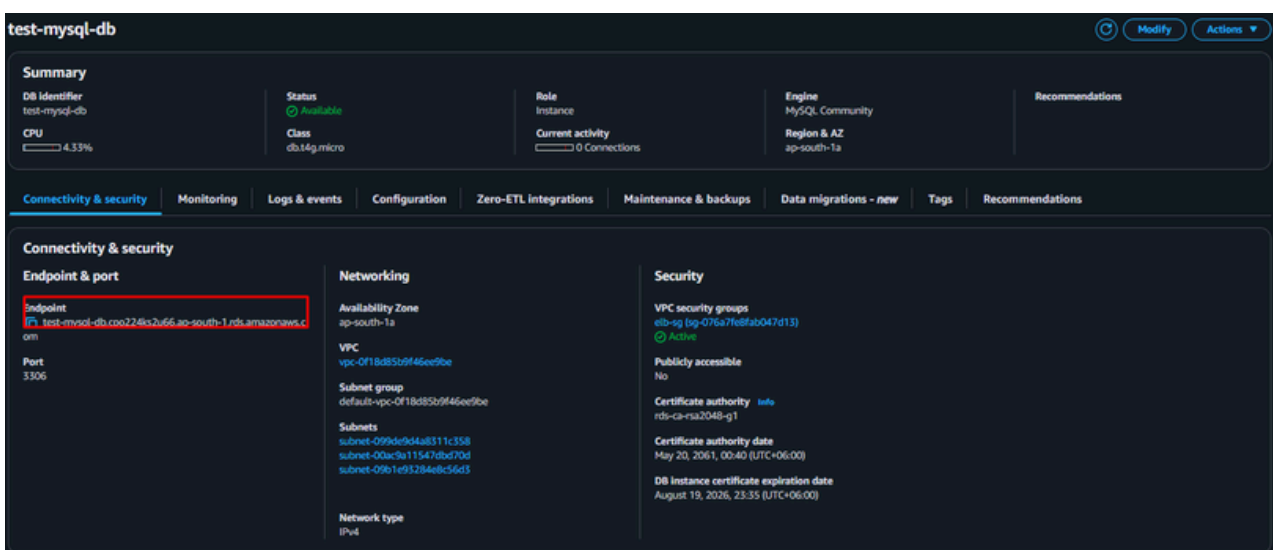


Now, connect to your EC2 instance and install the necessary configurations. You can do this via SSH or directly from the EC2 terminal. In my case, I used the EC2 terminal.

```
ubuntu@ip-172-31-39-126:~$ mysql --version
mysql Ver 15.1 Distrib 10.11.13-MariaDB, for debian-linux-gnu (x86_64) using EditLine wrapper
ubuntu@ip-172-31-39-126:~$
```

MySQL installed successfully.

Now, connect to the RDS MySQL. To do this, you need the endpoint and master username, which you can find in your database details.



Hands-on with (RDS + MySQL + EC2):

Instance			
Configuration	Instance class	Primary storage	Monitoring
DB Instance ID test-mysql-db	Instance class db.t4g.micro	Encryption Not enabled	Monitoring type Database Insights - Standard
Engine version 8.0.42	vCPU 2	Storage type General Purpose SSD (gp2)	Performance Insights Disabled
RDS Extended Support Disabled	RAM 1 GB	Storage 20 GiB	Enhanced Monitoring Disabled
DB name -	Availability	Provisioned IOPS -	
License model General Public License	Master username admin	Storage throughput -	

Now run this command with your configuration details:

```
mysql -h <RDS_Endpoint> -P 3306 -u <RDS_Username> -p
```

```
ubuntu@ip-172-31-39-126:~$ mysql --version
mysql Ver 15.1 Distrib 10.11.13-MariaDB, for debian-linux-gnu (x86_64) using EditLine wrapper
ubuntu@ip-172-31-39-126:~$ mysql -h test-mysql-db.cpo224ks2u66.ap-south-1.rds.amazonaws.com -P 3306 -u admin -p
Enter password: █
```

Type your database password and press enter.

```
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 32
Server version: 8.0.42 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> █
```

Successfully connected to the database. Now, test it by creating a new database and table, and inserting some records as well.

Hands-on with (RDS + MySQL + EC2):

Here are the results of the operations I performed using my command file: First, I created a new database, then a new table called users, inserted some user data, and finally displayed the list of users.

```
MySQL [(none)]> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
4 rows in set (0.009 sec)

MySQL [(none)]> CREATE DATABASE testdb;
Query OK, 1 row affected (0.009 sec)

MySQL [(none)]> use testdb;
Database changed
MySQL [testdb]> CREATE TABLE users (
  -> id INT AUTO_INCREMENT PRIMARY KEY,
  -> name VARCHAR(100) NOT NULL,
  -> email VARCHAR(100) UNIQUE NOT NULL
  -> );
Query OK, 0 rows affected (0.045 sec)

MySQL [testdb]> INSERT INTO users (name, email)
  -> VALUES
  -> ('Alamgir Hosen', 'alamgir@example.com'),
  -> ('Abir Ahmed', 'abir@example.com');
Query OK, 2 rows affected (0.006 sec)
Records: 2 Duplicates: 0 Warnings: 0

MySQL [testdb]> SHOW TABLES;
+-----+
| Tables_in_testdb |
+-----+
| users |
+-----+
1 row in set (0.002 sec)

MySQL [testdb]> SELECT * FROM users;
+----+-----+-----+
| id | name       | email                |
+----+-----+-----+
| 1  | Alamgir Hosen | alamgir@example.com |
| 2  | Abir Ahmed   | abir@example.com    |
+----+-----+-----+
2 rows in set (0.003 sec)

MySQL [testdb]> 
```

Successfully connected the RDS MySQL database with the EC2 instance.

Thank You

Stay Connect:

/in/alamgirweb11

/alamgirweb11

Dynamo DB

Amazon DynamoDB is a serverless, NoSQL, fully managed database with single-digit millisecond performance at any scale.

It is a **fully managed NoSQL database service** provided by AWS.

- **NoSQL** means it doesn't use the traditional table–row–column schema like relational databases (MySQL, PostgreSQL). Instead, it stores data as **key-value pairs** and **documents**.
- It's designed for **fast, consistent performance at any scale**.
- AWS takes care of infrastructure management, scaling, backups, patching, and availability — so developers can focus on building apps instead of managing DB servers.

Capabilities of DynamoDB

1. Query & Scan Operations

- **Query**: Efficient lookup based on primary key or indexes.
- **Scan**: Reads all items in a table (less efficient).

2. Indexes

- **Primary Key**: Partition key (mandatory) + optional sort key.
- **Secondary Indexes**:
 - **Global Secondary Index (GSI)** – query using a different partition key.
 - **Local Secondary Index (LSI)** – same partition key, different sort key.

3. DAX (DynamoDB Accelerator)

- In-memory caching layer.
- Reduces read latency from milliseconds to microseconds.

4. Streams

- Captures table activity (insert/update/delete).
- Used for event-driven applications, analytics, auditing.

5. Transactions

- Supports ACID transactions for multi-item, multi-table operations.

6. TTL (Time to Live)

- Automatically delete expired items (useful for session data, logs).

7. Backup & Restore

- On-demand backups and point-in-time recovery (PITR).

8. Integration

- Works seamlessly with **AWS Lambda, API Gateway, Kinesis, S3, Athena, EMR, SageMaker**, etc.

Key Features of DynamoDB

1. Managed & Serverless

- No need to provision or manage servers.
- Auto-scaling based on traffic.
- Pay only for what you use.

2. High Performance (Single-digit ms Latency)

- Optimized for low-latency reads/writes.
- Suitable for real-time applications.

3. Scalability

- Handles **millions of requests per second**.
- Scales seamlessly without downtime.

4. Flexible Data Model

- **Tables, Items, and Attributes:**
 - A **Table** = collection of items (like a database table).
 - An **Item** = single record (like a row).
 - **Attributes** = fields of an item (like columns).
- Supports nested documents (JSON-like).

5. Primary Key Options

- **Partition Key (hash key):** Ensures data distribution.
- **Partition Key + Sort Key (composite key):** Allows range queries and grouping.

6. Indexes for Queries

- **Global Secondary Index (GSI):** Query on non-primary attributes.
- **Local Secondary Index (LSI):** Query with same partition key but different sort key.

Step 1: - Open AWS Console and search for DynamoDB and click on Create table and insert detail as shown in picture (for demo only).

Create table

Table details

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name
This will be used to identify your table.
user_table

Partition key
The partition key is part of the table's primary key. It is a hash value that is used to partition items from your table and allocate data across hosts for scalability and availability.
user_id

Sort key - optional
You can use a sort key as the second part of a table's primary key. The sort key allows you to sort an search among all items sharing the same partition key.
message

Table settings

☐ Default settings
The fastest way to create your table. You can modify most of these settings after your table has been created. To modify these settings, choose "Customize settings".

☒ Customized settings
Use these advanced features to make DynamoDB work better for your needs.

Table class

Select a table class to optimize your table's cost based on your workload requirements and data access patterns.

☒ DynamoDB Standard

Capacity calculator

Average item size (KB): 1

Item read/second: 1

Item write/second: 1

Read capacity units: 1

Write capacity units: 1

Region: us-east-1

Estimated cost: \$0.59 / month

Read/write capacity settings

☐ On-demand
Simplify billing by paying for the actual reads and writes your application performs.

☒ Provisioned
Manage and optimize your costs by allocating read/write capacity in advance.

Read capacity

☐ Auto scaling
Amazon adjusts provisioned throughput capacity on your behalf in response to actual traffic patterns.

☒ On/Off

Provisioned capacity units: 5

Write capacity

☐ Auto scaling
Amazon adjusts provisioned throughput capacity on your behalf in response to actual traffic patterns.

☒ On/Off

Minimum capacity units: 1

Maximum capacity units: 5

Target utilization (%): 70

Warm throughput

☒ Keep default values

☐ Increase warm throughput

Read units per second: 5

Write operations per second: 5

☒ Keep default values

☐ Increase warm throughput

Write units per second: 1

Secondary indexes

Name	Type	Partition key	Sort key	Projected attributes	Warm throughput
No indexes					

Use secondary indexes to perform queries on attributes that are not part of your table's primary key.

[Create global index](#)

Estimated read/write capacity cost

Here is the estimated total cost of provisioned read and write capacity for your table and indexes, based on your current settings. To learn more, see [Amazon DynamoDB pricing](#) for provisioned capacity.

Total read capacity units	Total write capacity units	Region	Estimated cost
5	2	us-east-1	\$1.46 / month

Encryption at rest

All user data stored in Amazon DynamoDB is fully encrypted at rest. By default, Amazon DynamoDB manages the encryption key, and you are not charged any fee for using it.

Encryption key management

☒ AWS managed key (AWS KMS)
The key is created and managed by DynamoDB. You are not charged additional fees for using this key.

☐ AWS managed key (AWS KMS)
The key is created in your account and managed by AWS Key Management Service (KMS). AWS KMS charges apply.

☐ Customer managed key (AWS KMS)
The key is stored in your account and managed by you. AWS KMS charges apply.

Deletion protection

☒ Deletion protection is turned off by default. Deletion protection protects the table from being deleted unintentionally. You can turn on deletion protection now, and you can also turn it on after the table has been created.

☐ Turn on deletion protection

Resource-based policy

The resource-based policy, written in JSON, helps manage access to this DynamoDB table.

[Policy examples](#)

[Add new statement](#)

[Edit statement](#)

Select a statement

Select an existing statement in the policy or add a new statement.

[+ Add new statement](#)

[+ Add new tag](#)

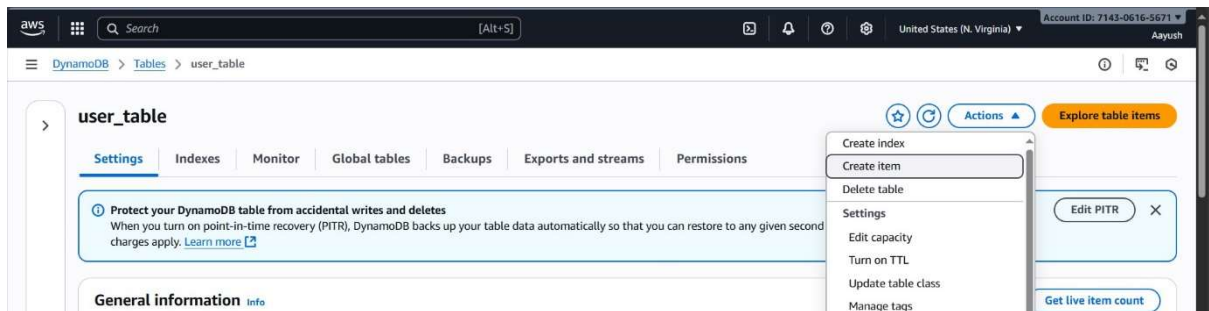
JSON: { "Statement": [{ "Effect": "Allow", "Principal": "*", "Action": "dynamodb:*", "Resource": "*" }] }

[Security](#) [Errors](#) [Warnings](#) [Suggestions](#)

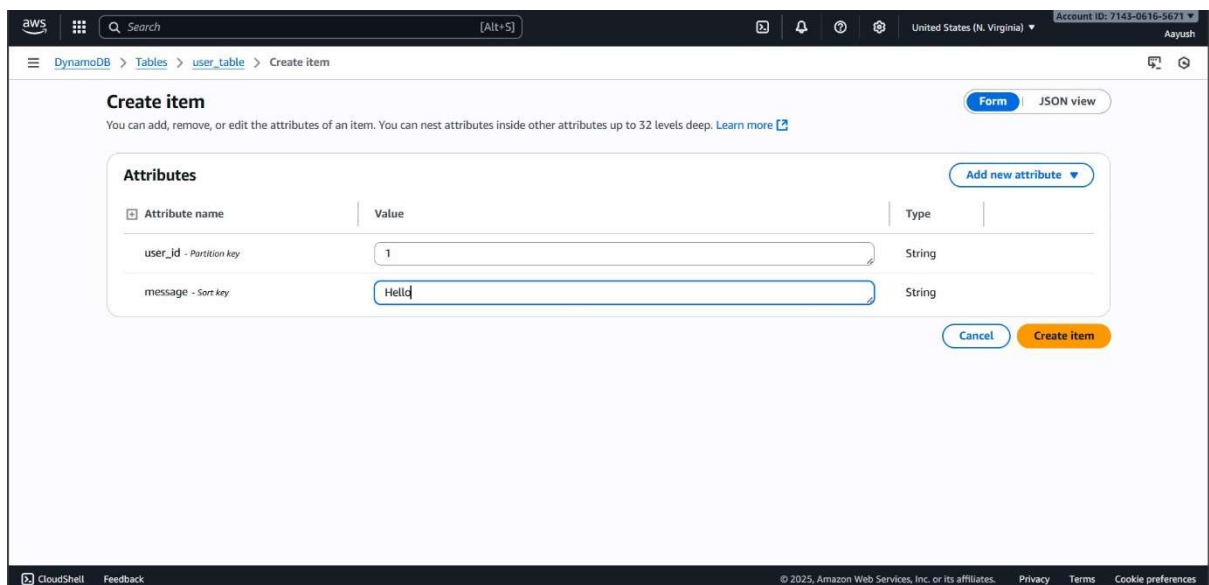
[Preview external access](#)

[Cancel](#) [Create table](#)

Step 2: - Now goto DynamoDB dashboard and on table name and then click on Action button then click on Create item.



Step 3: - Now enter value and you can add a new attribute then click on Create item.



Step 4: - Adding multiple custom item in table.

Create item

You can add, remove, or edit the attributes of an item. You can nest attributes inside other attributes up to 32 levels deep. [Learn more](#)

Attributes

Attribute name	Value	Type
user_id - Partition key	2	String
message - Sort key	How are you	String
message_id	1	String

[Add new attribute](#) [Remove](#) [Cancel](#) [Create item](#)

Step 5: - Now you can add parameters and click on scan and then click on query to run.

user_table

[Autopreview](#) [View table details](#)

Scan or query items

☐ Scan ☒ Query

Select a table or index: **Table - user_table**

Select attribute projection: **All attributes**

Partition key: user_id
1

Sort key: message
Equal to ☐ Sort descending

Filters - optional

[Run](#) [Reset](#)

Completed - Items returned: 2 - Items scanned: 2 - Efficiency: 100% - RCUs consumed: 0.5

Table: user_table - Items returned (2)

Query started on September 16, 2025, 02:16:50

	user_id (String)	message (String)	message_id
<input type="checkbox"/>	1	Bye	2
<input type="checkbox"/>	1	Hello	1