

# AWS RDS (Relational Database Service)



## 💡 What is AWS RDS?

Amazon RDS (Relational Database Service) is a **fully managed** service by AWS that allows you to set up, operate, and scale relational databases in the cloud with just a few clicks. It automates time-consuming tasks like provisioning, patching, backup, and recovery, so you can focus on your application.

## 📁 Supported Database Engines

RDS supports several popular relational database engines:

- **MySQL**
- **PostgreSQL**
- **MariaDB**
- **Oracle**
- **Microsoft SQL Server**
- **Amazon Aurora** (MySQL & PostgreSQL compatible)

## 🚀 Why is RDS Used?

- Easy to set up and manage
- Built-in security and backups 🔒
- Automated software patching ↻
- Monitoring and performance tuning with Amazon CloudWatch 📊
- Supports multiple DB engines

---

## 😬 Why use RDS instead of a DB on EC2?

- **RDS is Managed:** No need to manage hardware, OS, or database software manually
- **Automatic Backups & Snapshots** 📁
- **Easier to Scale** 📏

- **Optimized for Cost and Performance** 💰
- **Built-in High Availability (Multi-AZ)** 🔄
- **IAM Integration & Encryption** 🔒

## 📦 RDS Storage Scaling

- You can **scale up storage** without downtime in many cases
- Storage types: General Purpose (SSD), Provisioned IOPS (SSD), Magnetic
- Supports **auto-scaling** storage for certain DB engines 📦

## 📖 What is RDS Read Replica?

A **Read Replica** in Amazon RDS is a **read-only copy** of your primary database that is created using **asynchronous replication**. This means data from the primary DB is copied to the replica with a slight delay. Read Replicas are primarily used to handle **read-heavy workloads** and improve database performance without affecting the primary database operations.

## 🔧 Use Case for Read Replica

- Offload read-heavy traffic (like reporting dashboards)
- Test query performance on real data
- Geo-distributed read access 🌐

**Important:** Read Replicas are only for **SELECT** operations (i.e., read-only queries)

---

## 🌐 Read Replica - Network Cost

- **Within Same AZ or Region:** Minimal or no data transfer cost
- **Cross-Region:** Data transfer charges apply, but useful for global applications 🌐

---

## 🛡️ RDS Multi-AZ - Disaster Recovery







- Two copies of DB in different Availability Zones
- Automatic failover to standby in case of failure ⚠️
- Sync replication (not async like Read Replica)
- Minimal downtime and data loss

**Best for mission-critical applications needing high availability and durability.**


## AWS RDS - Hands-On Guide

This hands-on walkthrough helps you set up a simple RDS instance using the AWS Management Console.

### Step-by-Step Guide

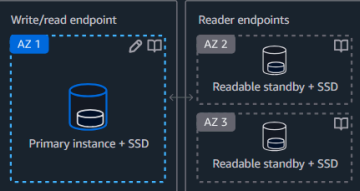
1. Go to RDS Dashboard
  - Sign in to the [AWS Management Console](#) 
  - In the search bar, type RDS and click on RDS
2. Click on "Create database"
  - Hit the Create database button 
3. Choose a creation method
  - Select Standard Create 
4. Engine Options
  - Choose your preferred database engine (e.g., MySQL, PostgreSQL) 
5. Templates
  - Under Templates, select Free tier 
6. Availability and Durability
  - Select Single-AZ DB instance deployment (1 instance) 
  - This is suitable for dev/test environments

**Availability and durability**

**Deployment options** [Info](#)  
Choose the deployment option that provides the availability and durability needed for your use case. AWS is committed to a certain level of uptime depending on the deployment option you choose. Learn more in the [Amazon RDS service level agreement \(SLA\)](#) .

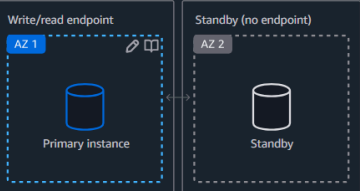
**Multi-AZ DB cluster deployment (3 instances)**  
Creates a primary DB instance with two readable standbys in separate Availability Zones. This setup provides:

- 99.95% uptime
- Redundancy across Availability Zones
- Increased read capacity
- Reduced write latency



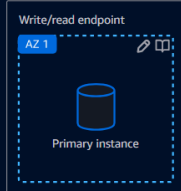
**Multi-AZ DB instance deployment (2 instances)**  
Creates a primary DB instance with a non-readable standby instance in a separate Availability Zone. This setup provides:






- 99.95% uptime
- Redundancy across Availability Zones






**Single-AZ DB instance deployment (1 instance)**  
Creates a single DB instance without standby instances. This setup provides:


- 99.5% uptime
- No data redundancy





7.  For Availability and durability, choose Single-AZ DB instance deployment
8.  Set DB instance identifier to **MyDBInstance**
9.  In Credentials settings, set your Master username and password
10.  Under Instance configuration, select Burstable classes (e.g., db.t3.micro)
11.  Choose your preferred Storage type (e.g., General Purpose SSD)

12.  In Connectivity section:

-  Set Compute resource to Don't connect to an EC2 compute resource
-  Network: Use IPv4
-  Select a VPC

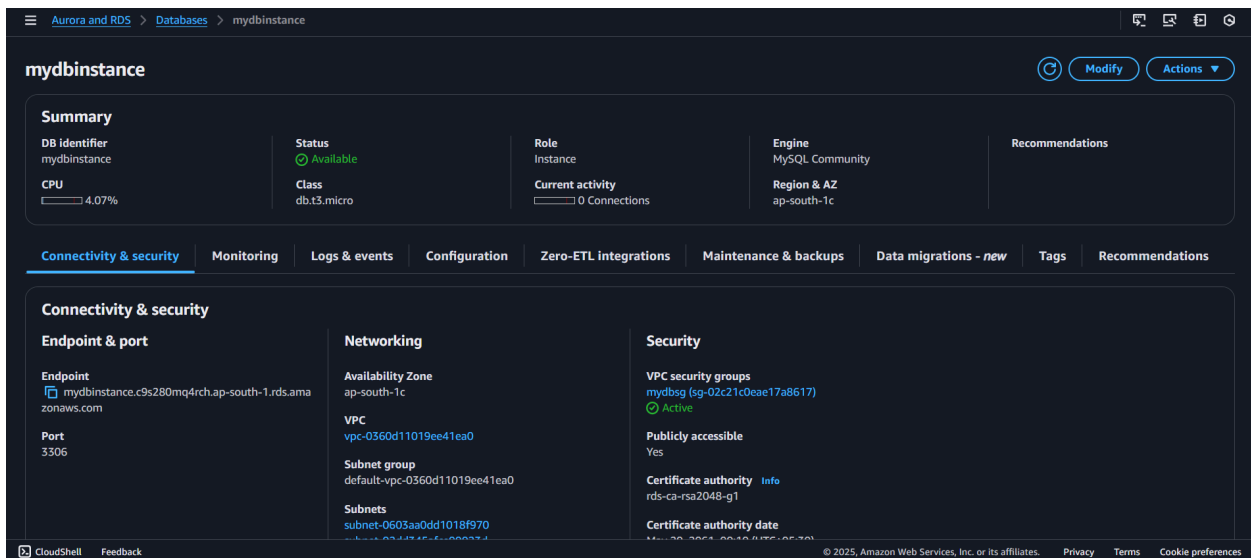
13.  Set Public access to Yes or No (based on use case)

14.  Configure a Security Group to allow access (e.g., port 3306)

15.  Set Database port to **3306**

16.  Choose Password authentication under Database authentication

17.  Click Create database










The screenshot shows the AWS RDS console for a database instance named 'mydbinstance'. The 'Connectivity & security' tab is selected, displaying the following details:




- Endpoint & port:** Endpoint is 'mydbinstance.c9s280mq4rch.ap-south-1.rds.amazonaws.com', and Port is '3306'.
- Networking:** Availability Zone is 'ap-south-1c', VPC is 'vpc-0360d11019ee41ea0', Subnet group is 'default-vpc-0360d11019ee41ea0', and Subnets include 'subnet-0603aa0dd1018f970'.
- Security:** VPC security groups include 'mydbsg (sg-02c21c0cae17a8617)' which is 'Active'. 'Publicly accessible' is set to 'Yes'. 'Certificate authority' is 'rds-ca-rsa2048-g1'.

## Launch EC2 Instance to Connect with RDS

Follow these steps to launch an EC2 instance and connect it with your RDS instance:

### Steps to Launch EC2 and Connect to RDS:


1.  Go to EC2 in AWS Console
2.  Click Launch Instance
3.  Name your instance (e.g., **MyEC2Instance**)
4.  Choose an Amazon Machine Image (AMI) — e.g., Amazon Linux 2023
5.  Select Instance type — e.g., **t2.micro** (Free Tier eligible)
6.  Create or choose an existing Key pair for SSH access
7.  In Network Settings:

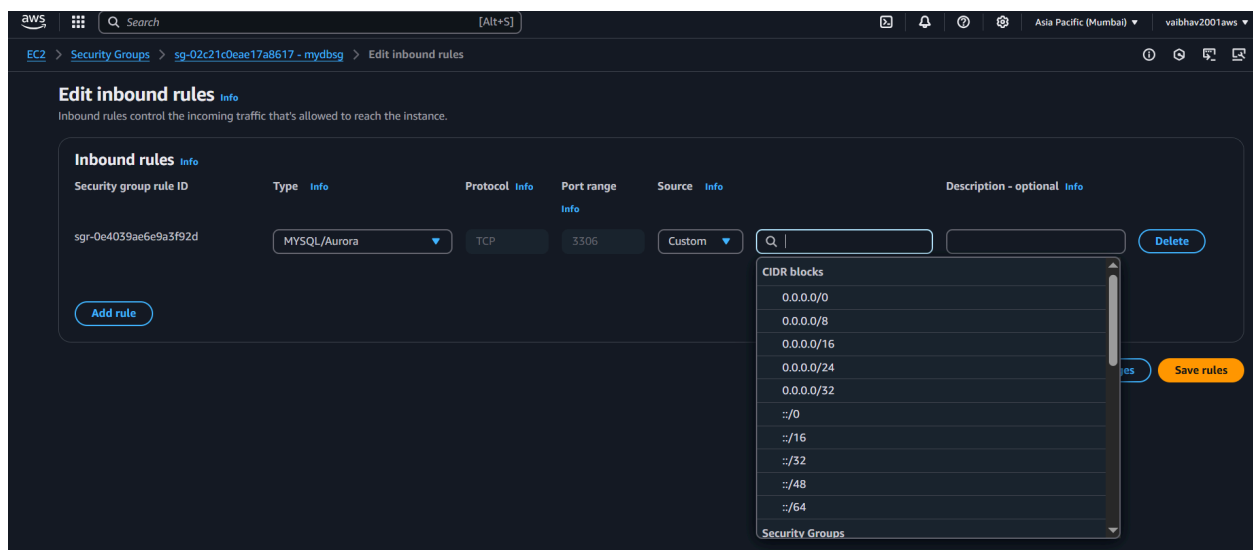
- Select the same VPC as your RDS instance
  - Choose a public subnet
  - Enable Auto-assign Public IP 
  - Select or create a Security Group
    - Allow SSH (port 22) from your IP
    - Allow MySQL/Aurora (port 3306) to connect to RDS
8.  Leave storage as default or increase as needed
9.  Click Launch Instance

 "Now you have to connect EC2 instance to RDS so the SG of RDS should allow traffic coming from SG of EC2" means:

In AWS, Security Groups (SGs) act like virtual firewalls that control inbound and outbound traffic to your resources (like EC2 and RDS).

When your EC2 instance tries to connect to the RDS database, the RDS Security Group needs to allow incoming traffic from the EC2's Security Group on the database port (usually **3306** for MySQL).

 Instead of allowing traffic from any IP (which is insecure), you allow it only from the EC2 instance's SG, making it safer and more controlled.








**Important :-** Choose SG of you ec2 from the list.

## Connect EC2 Instance to RDS (MySQL)

Follow these steps to connect your EC2 instance to your RDS database using EC2 Instance Connect and the MySQL client.

### Steps:

1.  **Connect to EC2 Instance**
  - Use **EC2 Instance Connect** from the AWS Console
2.  **Install MySQL Client**
  - Run the following command:  
`sudo yum install mysql -y`
3.  **Connect to RDS Database**
  - Run this command:  
`mysql -h <RDS-endpoint> -u <your-username> -p`
  - Replace **<RDS-endpoint>** with your RDS DB endpoint
  - Replace **<your-username>** with the DB username you set up
4.  **Enter Password**
  - Type in the password you configured for the database user
5.  **Done!**
  - You are now connected to your RDS database from your EC2 instance!
  - You can start using SQL commands to interact with your database

```
[ec2-user@ip-172-31-41-173 ~]$ sudo mysql -h mydbinstance.c9s280mq4rch.ap-south-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 33
Server version: 8.0.40 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)]>
```

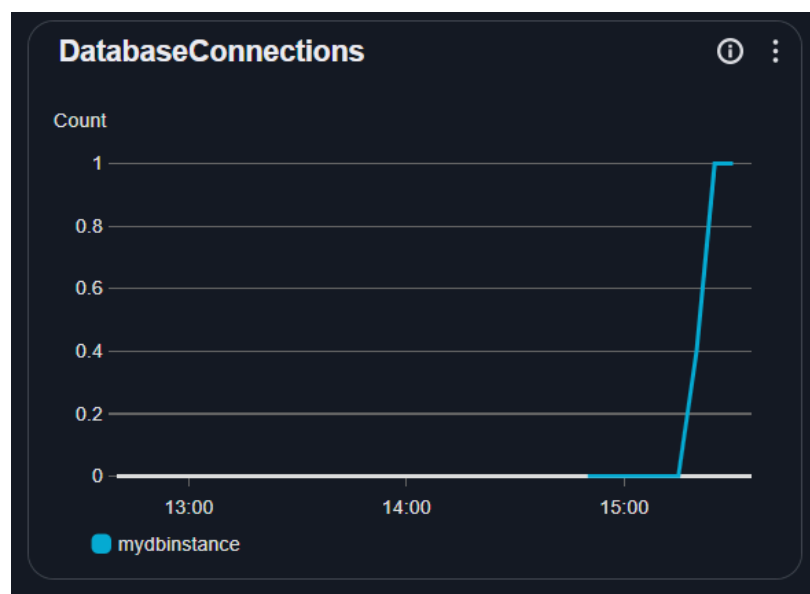
```
MySQL [(none)]> create database employee;
Query OK, 1 row affected (0.00 sec)

MySQL [(none)]> show databases;
+-----+
| Database |
+-----+
| employee |
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.01 sec)

MySQL [(none)]> 
```

<b>Status</b> ✔ Available	<b>Role</b> Instance
<b>Class</b> db.t3.micro	<b>Current activity</b> 1 Connections

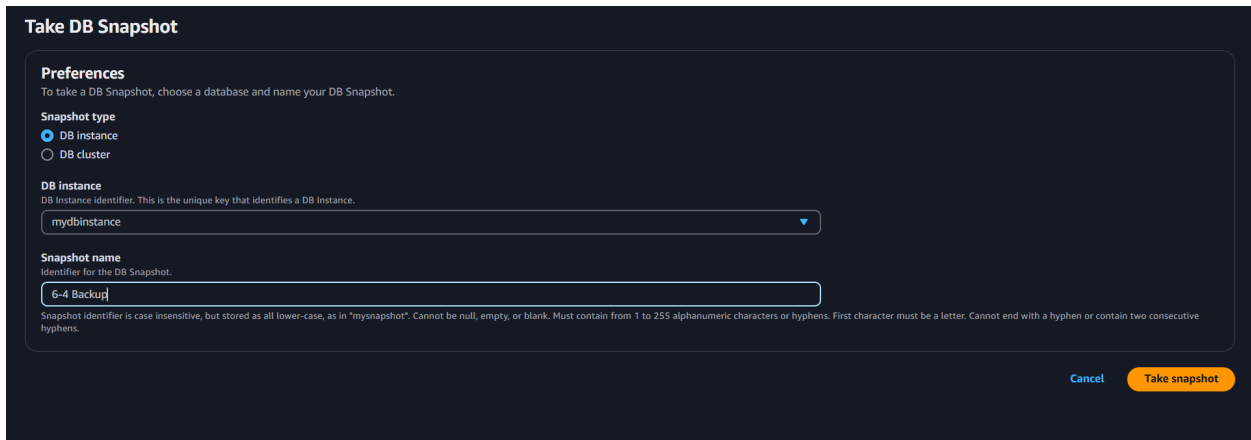
*You can the connection is increased to 1 from 0.*



## ✓ How to Take a Snapshot

1. 📁 Go to **RDS** in the AWS Console
2. 📋 Select your **Database instance** from the list
3. ⚙️ Click on **Actions**
4. 📷 Choose **Take snapshot**
5. 🏷️ Give your snapshot a **name**
6. 🚀 Click **Take Snapshot**

🕒 It may take a few minutes depending on your database size.



The screenshot shows the 'Take DB Snapshot' dialog box in the AWS Management Console. It has a dark theme. The title is 'Take DB Snapshot'. Below the title is a 'Preferences' section with the instruction: 'To take a DB Snapshot, choose a database and name your DB Snapshot.' There are three main sections: 'Snapshot type' with radio buttons for 'DB instance' (selected) and 'DB cluster'; 'DB instance' with a dropdown menu showing 'mydbinstance'; and 'Snapshot name' with a text input field containing '6-4 Backup'. Below the input field is a small note: 'Snapshot identifier is case insensitive, but stored as all lower-case, as in "mysnapshot". Cannot be null, empty, or blank. Must contain from 1 to 255 alphanumeric characters or hyphens. First character must be a letter. Cannot end with a hyphen or contain two consecutive hyphens.' At the bottom right are 'Cancel' and 'Take snapshot' buttons.

## 💡 Uses of Snapshots:

- 🛠️ **Backup:** Create a backup before making major changes
- 🔧 **Testing:** Restore to a new instance for testing purposes
- 🚨 **Disaster Recovery:** Recover data if the original DB is lost or corrupted
- 📦 **Migration:** Move data to another region or account easily



## What You Achieved

### 1. Provisioned an RDS Database

You created a fully managed MySQL database instance using the AWS RDS console, with configuration for storage, networking, credentials, and high availability options.

### 2. Launched and Configured an EC2 Instance

You spun up an Amazon Linux EC2 instance within the same VPC and subnet, ensuring it could securely connect to the RDS instance.

### 3. Implemented Secure Access

Instead of allowing open access, you configured **Security Groups** to allow MySQL traffic **only from the EC2 instance**, following the principle of least privilege 🗝️.

### 4. Connected to the Database

You installed the MySQL client on EC2 and successfully connected to the RDS instance using the RDS endpoint, demonstrating how cloud-based apps and services connect to databases.

### 5. Took Manual Snapshots for Backup

You created a snapshot of your RDS instance, showing how to perform point-in-time backups and prepare for disaster recovery, testing, or migration scenarios.