

# NullClass Cloud Technology (AWS)

**1.Create an RDS MySQL Database and Connect from EC2** Launch an RDS MySQL instance inside your VPC. Modify the security group to allow connections from your EC2 instance. Connect to the database from your EC2 instance using MySQL CLI.

This guide explains how to set up an **Amazon RDS MySQL database** and connect to it from an **EC2 instance**. We will launch an EC2 server, create an RDS MySQL database in the same VPC, configure security groups to allow communication, install the MySQL client on EC2, and then connect to the database. Finally, we will create a table, insert sample records, and verify the connection.

## 1. Launch an EC2 Instance

1. Log in to the **AWS Management Console**.
2. Navigate to **EC2 > Instances > Launch Instance**.
3. Choose an Amazon Machine Image (AMI):
  - o Select **Amazon Linux 2023** (or Amazon Linux 2).
4. Choose instance type:
  - o Example: **t2.micro** (Free Tier eligible).
5. **Key Pair**: Create or select an existing key pair (e.g., sonu-3.pem).
6. Network settings:
  - o Attach to your default VPC.
  - o Enable a public IP.
7. **Launch** the instance.

The screenshot shows the AWS EC2 Instances page. On the left, a sidebar navigation includes EC2, Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, and AMI Catalog. Below these are Elastic Block Store, Volumes, Snapshots, and Lifecycle Manager. At the bottom of the sidebar are CloudShell and Feedback links. The main content area displays a table of instances with one row for "mydb-2". The table columns include Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability Zone. The instance "mydb-2" is listed as "Running" with a t2.micro type, 2/2 checks passed, and located in us-east-1a. A "Select an instance" dropdown is open below the table. The top right of the page shows account information (Account ID: 8582-6528-6505) and a region selector for United States (N. Virginia). The bottom of the page has standard AWS footer links: CloudShell, Feedback, Privacy, Terms, and Cookie preferences.

The screenshot shows the Instance summary page for instance i-096805f91dca3a202. The left sidebar is identical to the previous screenshot. The main content area is titled "Instance summary for i-096805f91dca3a202 (mydb-2)". It contains detailed information about the instance, organized into three columns. The first column includes fields like Instance ID (i-096805f91dca3a202), IPv6 address (empty), Hostname type (IP name: ip-172-31-86-227.ec2.internal), Answer private resource DNS name (IPv4 (A)), Auto-assigned IP address (54.174.158.139 [Public IP]), IAM Role (empty), IMDSv2 (Required), and Operator (empty). The second column includes fields like Public IPv4 address (54.174.158.139), Instance state (Running), Private IP DNS name (IPv4 only) (ip-172-31-86-227.ec2.internal), Instance type (t2.micro), VPC ID (vpc-02e06b741fc8ee298), Subnet ID (subnet-0c3892124977de756), Instance ARN (arn:aws:ec2:us-east-1:858265286505:instance/i-096805f91dca3a202), and Managed (false). The third column includes fields like Private IPv4 addresses (172.31.86.227), Public DNS (ec2-54-174-158-139.compute-1.amazonaws.com), Elastic IP addresses (empty), AWS Compute Optimizer finding (Opt-in to AWS Compute Optimizer for recommendations), and Auto Scaling Group name (empty). The top right of the page shows account information (Account ID: 8582-6528-6505) and a region selector for United States (N. Virginia). The bottom of the page has standard AWS footer links: CloudShell, Feedback, Privacy, Terms, and Cookie preferences.

## 2. Launch an RDS MySQL Database

1. Go to **RDS > Databases > Create Database**.
2. Choose **Standard Create → MySQL**.
3. Select version: Example **MySQL 8.0**.

4. Choose **Free Tier** template.
5. DB instance identifier:
  - Example: mydatabase.
6. Set username & password:
  - Username: databasemysql.
  - Enter a secure password.
7. Connectivity:
  - VPC: Select the same VPC as your EC2 instance.
  - Subnet group: Default.
  - Public access: **No** (private DB).
  - Security group: Select or create one (we'll configure it next).
8. Create the database.

The screenshot shows the AWS Aurora and RDS console in a web browser. The left sidebar navigation includes links for Dashboard, Databases (which is selected), Query editor, Performance insights, Snapshots, Exports in Amazon S3, Automated backups, Reserved instances, Proxies, Subnet groups, Parameter groups, Option groups, Custom engine versions, Zero-ETL integrations, Events, and Event subscriptions. The main content area displays a table titled "Databases (1)". The table has columns for DB Identifier, Status, Role, Engine, Region ..., and Size. One row is shown, corresponding to the database "mydatabase". The status is "Available", the role is "Instance", the engine is "MySQL Co...", the region is "us-east-1b", and the size is "db.t3.micro". There are buttons for Group resources, Modify, Actions, and Create database. The top right corner shows the account ID "8582-6528-6505" and the region "United States (N. Virginia) shnthv\_45". The bottom of the screen shows the Mac OS X dock with various application icons.

Databases (1)						
	DB Identifier	Status	Role	Engine	Region ...	Size
<input type="radio"/>	mydatabase	Available	Instance	MySQL Co...	us-east-1b	db.t3.micro

### 3. Configure Security Groups

For EC2 to connect to RDS, security rules must allow traffic:

- **On RDS Security Group:**
  1. Go to **EC2 > Security Groups**.
  2. Find the RDS security group (e.g., rds-ec2-1).
  3. Edit inbound rules → Add rule:
    - **Type:** MySQL/Aurora
    - **Protocol:** TCP
    - **Port:** 3306
    - **Source:** Custom → Select your EC2's Security Group
- **On EC2 Security Group:**
  1. Add inbound rule:
    - **Type:** SSH
    - **Protocol:** TCP
    - **Port:** 22
    - **Source:** 0.0.0.0/0 (for testing; later restrict to your IP).

Now EC2 can reach RDS on port **3306**.

EC2 > Security Groups > sg-030970b34ea6c8907 - sg1

**Details**

Security group name sg1	Security group ID sg-030970b34ea6c8907	Description sg1	VPC ID vpc-02e06b741fc8ee298
Owner 858265286505	Inbound rules count 2 Permission entries	Outbound rules count 2 Permission entries	

**Inbound rules** | Outbound rules | Sharing - new | VPC associations - new | Tags

**Inbound rules (2)**

Name	Security group rule ID	IP version	Type	Protocol	Port range
-	sgr-09a62c715c9cde282	IPv4	SSH	TCP	22
-	sgr-0be046af58db09704	-	MySQL/Aurora	TCP	3306

## 4. Connect to Your EC2 Instance via SSH

1. Open a terminal (Linux/Mac) or PowerShell (Windows).
2. Locate your key file: sonu-3.pem.
3. Set permissions so it's not publicly visible:
4. chmod 400 sonu-3.pem
5. Connect using EC2's **Public DNS**:
6. ssh -i "sonu-3.pem" ec2-user@ec2-54-174-158-139.compute-1.amazonaws.com
7. Once connected, you'll be inside your EC2 shell.

## 5. Install MySQL Client on EC2

On Amazon Linux 2023, run:

```
sudo dnf update -y
```

```
sudo dnf install -y mariadb105
```

This installs the MySQL/MariaDB client, which lets you connect to RDS.

## 6. Connect from EC2 to RDS

1. Go to **RDS > Databases > your database (mydatabase)**.
2. Copy the **endpoint** (e.g., mydatabase.cuf2q40i6294.us-east-1.rds.amazonaws.com)
3. From your EC2 terminal, connect:
4. mysql -h mydatabase.cuf2q40i6294.us-east-1.rds.amazonaws.com -u databasemysql -p
5. Enter your DB password when prompted.
6. You are now connected to your MySQL RDS database

Aurora and RDS > Databases > mydatabase

<b>Aurora and RDS</b>	<b>Subnets</b> subnet-01af2c369e5f2261d subnet-05d33a19210e80703 subnet-0768c2aec47180ef4 subnet-0145efd42ec030e3d subnet-09d0306958fdab94a subnet-0c3892124977de756	<b>Certificate authority</b> <a href="#">Info</a> rds-ca-rsa2048-g1
	<b>Network type</b> IPv4	<b>Certificate authority date</b> May 26, 2061, 05:04 (UTC+05:30)
		<b>DB instance certificate expiration date</b> August 05, 2026, 18:38 (UTC+05:30)

**Connected compute resources (1) [Info](#)**

Connections to compute resources that were created automatically by RDS are shown here. Connections to compute resources that were created manually aren't shown.

Resource identifier	Resource type	Availability Zone	VPC security group	Compute resource security group	Connected
<a href="#">i-096805f91dca3a202</a>	EC2 Instance	<a href="#">us-east-1a</a>	<a href="#">rds-ec2-1</a>	<a href="#">ec2-rds-1</a>	-

**Proxies (0) [Create proxy](#)**

No proxies

You don't have any proxies.

Aurora and RDS > Databases > mydatabase

<b>Summary</b>	<b>Recommendations</b> <a href="#">2 Informational</a>									
<b>DB identifier</b> mydatabase  <b>CPU</b> 	<b>Status</b>  Available <b>Role</b> Instance <b>Engine</b> MySQL Community <b>Region &amp; AZ</b> us-east-1b									
<b>Connectivity &amp; security</b> <a href="#">Modify</a> <a href="#">Actions</a> <ul style="list-style-type: none"> <li><a href="#">Monitoring</a></li> <li><a href="#">Logs &amp; events</a></li> <li><a href="#">Configuration</a></li> <li><a href="#">Zero-ETL integrations</a></li> <li><a href="#">Maintenance &amp; backups</a></li> </ul>										
<h3>Connectivity &amp; security</h3> <table border="1"> <tr> <td><b>Endpoint &amp; port</b></td> <td><b>Networking</b></td> <td><b>Security</b></td> </tr> <tr> <td> <b>Endpoint</b>   <a href="#">mydatabase.cuf2q40l6294.us-east-1.rds.amazonaws.com</a> </td> <td> <b>Availability Zone</b> us-east-1b   <b>VPC</b> <a href="#">vpc-02e06b741fc8ee298</a> </td> <td> <b>VPC security groups</b>  <a href="#">rds-ec2-1 (sg-0227caec5e6dd5ca3)</a>   Active  <a href="#">default (sg-01be6af2bf2e366e3)</a>   Active         </td> </tr> <tr> <td> <b>Port</b> 3306         </td> <td> <b>Subnet group</b> <a href="#">default-vpc-02e06b741fc8ee298</a>   <b>Subnets</b>  <a href="#">subnet-01af2c369e5f2261d</a>  <a href="#">subnet-05d33a19210e80703</a> </td> <td> <b>Publicly accessible</b> No   <b>Certificate authority</b> <a href="#">Info</a> rds-ca-rsa2048-g1         </td> </tr> </table>		<b>Endpoint &amp; port</b>	<b>Networking</b>	<b>Security</b>	<b>Endpoint</b>  <a href="#">mydatabase.cuf2q40l6294.us-east-1.rds.amazonaws.com</a>	<b>Availability Zone</b> us-east-1b  <b>VPC</b> <a href="#">vpc-02e06b741fc8ee298</a>	<b>VPC security groups</b> <a href="#">rds-ec2-1 (sg-0227caec5e6dd5ca3)</a>  Active <a href="#">default (sg-01be6af2bf2e366e3)</a>  Active	<b>Port</b> 3306	<b>Subnet group</b> <a href="#">default-vpc-02e06b741fc8ee298</a>  <b>Subnets</b> <a href="#">subnet-01af2c369e5f2261d</a> <a href="#">subnet-05d33a19210e80703</a>	<b>Publicly accessible</b> No  <b>Certificate authority</b> <a href="#">Info</a> rds-ca-rsa2048-g1
<b>Endpoint &amp; port</b>	<b>Networking</b>	<b>Security</b>								
<b>Endpoint</b>  <a href="#">mydatabase.cuf2q40l6294.us-east-1.rds.amazonaws.com</a>	<b>Availability Zone</b> us-east-1b  <b>VPC</b> <a href="#">vpc-02e06b741fc8ee298</a>	<b>VPC security groups</b> <a href="#">rds-ec2-1 (sg-0227caec5e6dd5ca3)</a>  Active <a href="#">default (sg-01be6af2bf2e366e3)</a>  Active								
<b>Port</b> 3306	<b>Subnet group</b> <a href="#">default-vpc-02e06b741fc8ee298</a>  <b>Subnets</b> <a href="#">subnet-01af2c369e5f2261d</a> <a href="#">subnet-05d33a19210e80703</a>	<b>Publicly accessible</b> No  <b>Certificate authority</b> <a href="#">Info</a> rds-ca-rsa2048-g1								

## 7. Create a Database and Table

Inside the MySQL shell, run:

CREATE DATABASE company;

USE company;

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(50),

email VARCHAR(100)

);

## 8. Insert and Verify Data

Insert some sample records:

```
INSERT INTO users (name, email) VALUES ('Alice', 'alice@example.com');
```

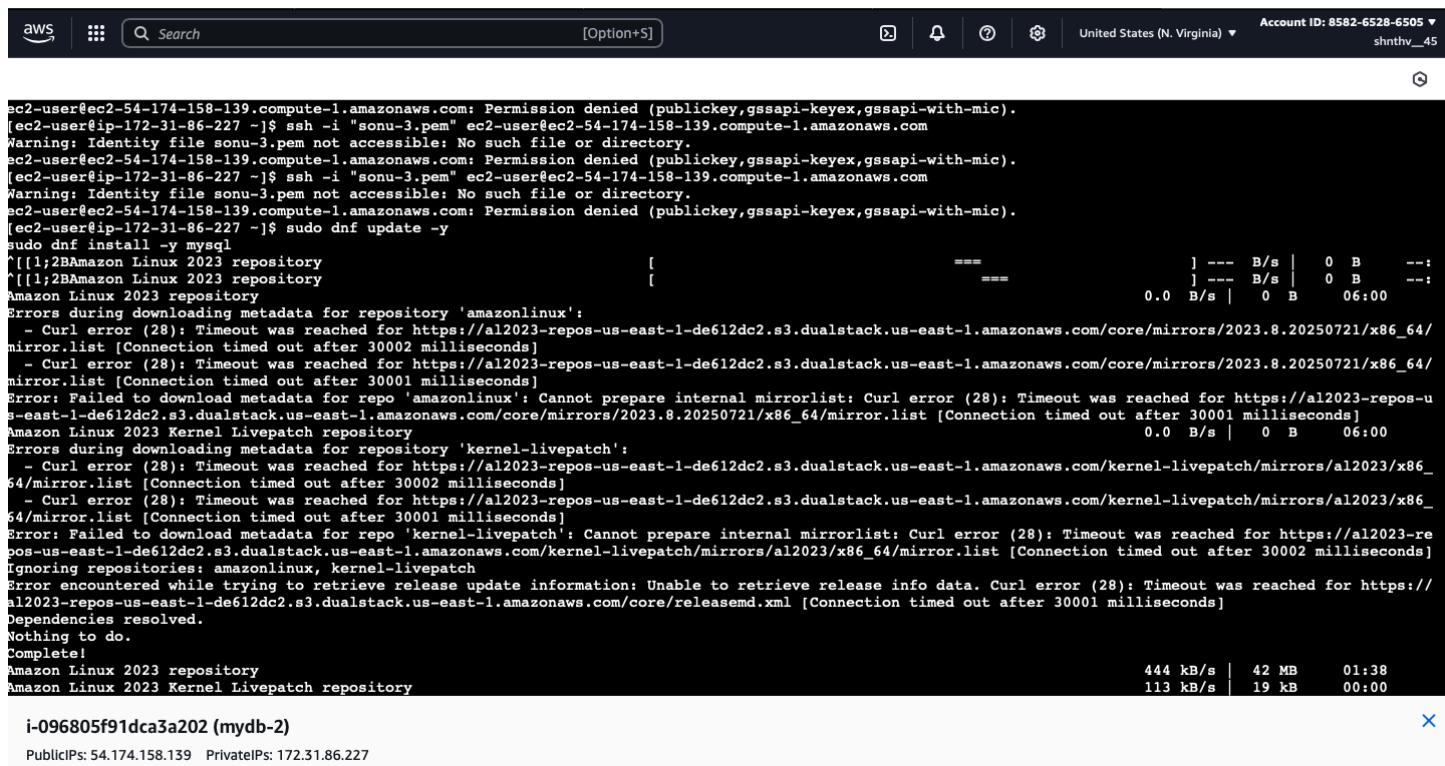
```
INSERT INTO users (name, email) VALUES ('Bob', 'bob@example.com');
```

```
INSERT INTO users (name, email) VALUES ('Charlie', 'charlie@example.com');
```

```
INSERT INTO users (name, email) VALUES ('Diana', 'diana@example.com');
```

Check the data:

```
SELECT * FROM users;
```



The screenshot shows a terminal window titled "CloudShell" with the AWS logo. The URL is "https://cloudshell.us-east-1.amazonaws.com". The session ID is "shnthy\_45". The account ID is "8582-6528-6505". The terminal shows the following MySQL command being run:

```
ec2-user@ec2-54-174-158-139.compute-1.amazonaws.com: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[ec2-user@ip-172-31-86-227 ~]$ ssh -i "sonu-3.pem" ec2-user@ec2-54-174-158-139.compute-1.amazonaws.com
Warning: Identity file sonu-3.pem not accessible: No such file or directory.
ec2-user@ec2-54-174-158-139.compute-1.amazonaws.com: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[ec2-user@ip-172-31-86-227 ~]$ ssh -i "sonu-3.pem" ec2-user@ec2-54-174-158-139.compute-1.amazonaws.com
Warning: Identity file sonu-3.pem not accessible: No such file or directory.
ec2-user@ec2-54-174-158-139.compute-1.amazonaws.com: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[ec2-user@ip-172-31-86-227 ~]$ sudo dnf update -y
sudo dnf install -y mysql
[[1;2BAmazon Linux 2023 repository
[[1;2BAmazon Linux 2023 repository
Amazon Linux 2023 repository
Errors during downloading metadata for repository 'amazonlinux':
 - Curl error (28): Timeout was reached for https://al2023-repos-us-east-1-de612dc2.s3.dualstack.us-east-1.amazonaws.com/core/mirrors/2023.8.20250721/x86_64/mirror.list [Connection timed out after 30002 milliseconds]
 - Curl error (28): Timeout was reached for https://al2023-repos-us-east-1-de612dc2.s3.dualstack.us-east-1.amazonaws.com/core/mirrors/2023.8.20250721/x86_64/mirror.list [Connection timed out after 30001 milliseconds]
Error: Failed to download metadata for repo 'amazonlinux': Cannot prepare internal mirrorlist: Curl error (28): Timeout was reached for https://al2023-repos-us-east-1-de612dc2.s3.dualstack.us-east-1.amazonaws.com/core/mirrors/2023.8.20250721/x86_64/mirror.list [Connection timed out after 30001 milliseconds]
Amazon Linux 2023 Kernel Livepatch repository
0.0 B/s | 0 B 06:00
Errors during downloading metadata for repository 'kernel-livepatch':
 - Curl error (28): Timeout was reached for https://al2023-repos-us-east-1-de612dc2.s3.dualstack.us-east-1.amazonaws.com/kernel-livepatch/mirrors/al2023/x86_64/mirror.list [Connection timed out after 30002 milliseconds]
 - Curl error (28): Timeout was reached for https://al2023-repos-us-east-1-de612dc2.s3.dualstack.us-east-1.amazonaws.com/kernel-livepatch/mirrors/al2023/x86_64/mirror.list [Connection timed out after 30001 milliseconds]
Error: Failed to download metadata for repo 'kernel-livepatch': Cannot prepare internal mirrorlist: Curl error (28): Timeout was reached for https://al2023-repos-us-east-1-de612dc2.s3.dualstack.us-east-1.amazonaws.com/kernel-livepatch/mirrors/al2023/x86_64/mirror.list [Connection timed out after 30002 milliseconds]
Ignoring repositories: amazonlinux, kernel-livepatch
Error encountered while trying to retrieve release update information: Unable to retrieve release info data. Curl error (28): Timeout was reached for https://al2023-repos-us-east-1-de612dc2.s3.dualstack.us-east-1.amazonaws.com/core/releasemd.xml [Connection timed out after 30001 milliseconds]
Dependencies resolved.
Nothing to do.
Complete!
Amazon Linux 2023 repository
Amazon Linux 2023 Kernel Livepatch repository
444 kB/s | 42 MB 01:38
113 kB/s | 19 kB 00:00
-i-096805f91dca3a202 (mydb-2)
PublicIPs: 54.174.158.139 PrivateIPs: 172.31.86.227
```

aws CloudShell Search [Option+S] United States (N. Virginia) ▾ Account ID: 8582-6528-6505 ▾ shnthy\_45

```
Complete!
[ec2-user@ip-172-31-86-227 ~]$ sudo yum install -y mysql
Last metadata expiration check: 0:00:56 ago on Mon Aug 18 20:49:20 2025.
No match for argument: mysql
Error: Unable to find a match: mysql
[ec2-user@ip-172-31-86-227 ~]$ sudo dnf install -y mariadb105
Last metadata expiration check: 0:01:34 ago on Mon Aug 18 20:49:20 2025.
Dependencies resolved.

=====
  Package           Architecture   Version      Repository    Size
=====
Installing:
  mariadb105      x86_64        3:10.5.29-1.amzn2023.0.1      amazonlinux  1.5 M
Installing dependencies:
  mariadb-connector-c      x86_64        3.3.10-1.amzn2023.0.1      amazonlinux  211 k
  mariadb-connector-c-config      noarch      3.3.10-1.amzn2023.0.1      amazonlinux  9.9 k
  mariadb105-common      x86_64        3:10.5.29-1.amzn2023.0.1      amazonlinux  28 k
  perl-Sys-Hostname      x86_64        1.23-477.amzn2023.0.7      amazonlinux  16 k

Transaction Summary
=====
Install 5 Packages

Total download size: 1.8 M
Installed size: 19 M
Downloading Packages:
(1/5): mariadb-connector-c-config-3.3.10-1.amzn2023.0.1.noarch.rpm          204 kB/s | 9.9 kB   00:00
(2/5): mariadb-connector-c-3.3.10-1.amzn2023.0.1.x86_64.rpm                  3.5 MB/s | 211 kB   00:00
(3/5): mariadb105-10.5.29-1.amzn2023.0.1.x86_64.rpm                      18 MB/s | 1.5 MB   00:00
(4/5): mariadb105-common-10.5.29-1.amzn2023.0.1.x86_64.rpm                 724 kB/s | 28 kB   00:00
(5/5): perl-Sys-Hostname-1.23-477.amzn2023.0.7.x86_64.rpm                  347 kB/s | 16 kB   00:00

Total                                         13 MB/s | 1.8 MB   00:00

Running transaction check
Transaction check succeeded.
```

i-096805f91dca3a202 (mydb-2)

Public IPs: 54.174.158.139 Private IPs: 172.31.86.227

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

aws CloudShell Search [Option+S] United States (N. Virginia) ▾ Account ID: 8582-6528-6505 ▾ shnthy\_45

```
Verifying : mariadb105-common-3:10.5.29-1.amzn2023.0.1.x86_64          4/5
Verifying : perl-Sys-Hostname-1.23-477.amzn2023.0.7.x86_64              5/5

=====
WARNING:
 A newer release of "Amazon Linux" is available.

Available Versions:

Version 2023.8.20250808:
 Run the following command to upgrade to 2023.8.20250808:
 
  dnf upgrade --releasever=2023.8.20250808

Release notes:
 https://docs.aws.amazon.com/linux/al2023/release-notes/relnotes-2023.8.20250808.html

Version 2023.8.20250818:
 Run the following command to upgrade to 2023.8.20250818:
 
  dnf upgrade --releasever=2023.8.20250818

Release notes:
 https://docs.aws.amazon.com/linux/al2023/release-notes/relnotes-2023.8.20250818.html

=====

Installed:
 mariadb-connector-c-3.3.10-1.amzn2023.0.1.x86_64      mariadb-connector-c-config-3.3.10-1.amzn2023.0.1.noarch      mariadb105-3:10.5.29-1.amzn2023.0.1.x86_64
 mariadb105-common-3:10.5.29-1.amzn2023.0.1.x86_64      perl-Sys-Hostname-1.23-477.amzn2023.0.7.x86_64

Complete!
[ec2-user@ip-172-31-86-227 ~]$ mysql --version
mysql Ver 15.1 Distrib 10.5.29-MariaDB, for Linux (x86_64) using EditLine wrapper
[ec2-user@ip-172-31-86-227 ~]$ mysql -h mydatabase.cuf2q40i6294.us-east-1.rds.amazonaws.com -u mydatabase -p
Enter password: [REDACTED]
```

i-096805f91dca3a202 (mydb-2)

Public IPs: 54.174.158.139 Private IPs: 172.31.86.227

aws Search [Option+S] United States (N. Virginia) ▾ Account ID: 8582-6528-6505 ▾ shnthy\_45

```
Version 2023.8.20250818:  
Run the following command to upgrade to 2023.8.20250818:  
  
dnf upgrade --releasever=2023.8.20250818  
  
Release notes:  
https://docs.aws.amazon.com/linux/al2023/release-notes/relnotes-2023.8.20250818.html  
=====  
Installed:  
mariadb-connector-c-3.3.10-1.amzn2023.0.1.x86_64 mariadb-connector-c-config-3.3.10-1.amzn2023.0.1.noarch mariadb105-3:10.5.29-1.amzn2023.0.1.x86_64  
mariadb105-common-3:10.5.29-1.amzn2023.0.1.x86_64 perl-Sys-Hostname-1.23-477.amzn2023.0.7.x86_64  
  
Complete!  
[ec2-user@ip-172-31-86-227 ~]$ mysql --version  
mysql Ver 15.1 Distrib 10.5.29-MariaDB, for Linux (x86_64) using EditLine wrapper  
[ec2-user@ip-172-31-86-227 ~]$ mysql -h mydatabase.cuf2q40i6294.us-east-1.rds.amazonaws.com -u mydatabase -p  
Enter password:  
ERROR 1045 (28000): Access denied for user 'mydatabase'@'172.31.86.227' (using password: YES)  
[ec2-user@ip-172-31-86-227 ~]$ mysql -h mydatabase.cuf2q40i6294.us-east-1.rds.amazonaws.com -u mydatabase -p  
Enter password:  
ERROR 1045 (28000): Access denied for user 'mydatabase'@'172.31.86.227' (using password: YES)  
[ec2-user@ip-172-31-86-227 ~]$ mysql -h mydatabase.cuf2q40i6294.us-east-1.rds.amazonaws.com -u mydatabase -p  
Enter password:  
Welcome to the MariaDB monitor. Commands end with ; or \g.  
Your MySQL connection id is 1320  
Server version: 8.0.41 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)]>  
```

i-096805f91dca3a202 (mydb-2)

Public IPs: 54.174.158.139 Private IPs: 172.31.86.227

[CloudShell](#) [Feedback](#)

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

aws Search [Option+S] United States (N. Virginia) ▾ Account ID: 8582-6528-6505 ▾ shnthy\_45

```
MySQL [(none)]>  
MySQL [(none)]> CREATE DATABASE testdb;  
Query OK, 1 row affected (0.014 sec)  
  
MySQL [(none)]>  
MySQL [(none)]> USE testdb;  
Database changed  
MySQL [testdb]>  
MySQL [testdb]> CREATE TABLE users (  
    ->     id INT AUTO_INCREMENT PRIMARY KEY,  
    ->     name VARCHAR(50) NOT NULL,  
    ->     email VARCHAR(100) UNIQUE NOT NULL,  
    ->     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
    -> );  
Query OK, 0 rows affected (0.063 sec)  
  
MySQL [testdb]>  
MySQL [testdb]> INSERT INTO users (name, email) VALUES ('Alice', 'alice@example.com');  
Query OK, 1 row affected (0.009 sec)  
  
MySQL [testdb]> INSERT INTO users (name, email) VALUES ('Bob', 'bob@example.com');  
Query OK, 1 row affected (0.007 sec)  
  
MySQL [testdb]>  
MySQL [testdb]> SELECT * FROM users;  
+----+----+-----+  
| id | name | email           | created_at      |  
+----+----+-----+  
| 1  | Alice | alice@example.com | 2025-08-18 21:02:26 |  
| 2  | Bob   | bob@example.com  | 2025-08-18 21:02:26 |  
+----+----+-----+  
2 rows in set (0.001 sec)  
  
MySQL [testdb]>  
```

i-096805f91dca3a202 (mydb-2)

Public IPs: 54.174.158.139 Private IPs: 172.31.86.227

[CloudShell](#) [Feedback](#)

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

```
MySQL [testdb]> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
| testdb |
+-----+
5 rows in set (0.001 sec)

MySQL [testdb]> CREATE DATABASE projectdb;
Query OK, 1 row affected (0.006 sec)

MySQL [testdb]> USE projectdb;
Database changed
MySQL [projectdb]> CREATE TABLE users (
    -->   id INT AUTO_INCREMENT PRIMARY KEY,
    -->   name VARCHAR(50) NOT NULL,
    -->   email VARCHAR(100) UNIQUE NOT NULL,
    -->   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    --> );
Query OK, 0 rows affected (0.038 sec)

MySQL [projectdb]> INSERT INTO users (name, email) VALUES ('Alice', 'alice@example.com');
Query OK, 1 row affected (0.005 sec)

MySQL [projectdb]> INSERT INTO users (name, email) VALUES ('Bob', 'bob@example.com');
Query OK, 1 row affected (0.004 sec)

MySQL [projectdb]> INSERT INTO users (name, email) VALUES ('Charlie', 'charlie@example.com');
Query OK, 1 row affected (0.005 sec)

MySQL [projectdb]> INSERT INTO users (name, email) VALUES ('Diana', 'diana@example.com');
```

i-096805f91dca3a202 (mydb-2)

Public IPs: 54.174.158.139 Private IPs: 172.31.86.227

## **2. Configure Auto Scaling and Load Balancing** Create a launch template for EC2 instances. Set up an auto-scaling group with minimum and maximum instance limits. Attach an application load balancer to distribute traffic.

Auto Scaling Group + Load Balancer on AWS

This guide explains **from scratch** how to set up an Auto Scaling Group (ASG) with a Load Balancer (ALB) to run a simple website on EC2 instances.

We'll do this in 5 main parts:

1. **Launch Template** → Blueprint for EC2.
2. **Security Group** → Firewall for EC2.
3. **Auto Scaling Group (ASG)** → Manages EC2 automatically.
4. **Application Load Balancer (ALB)** → Distributes traffic.
5. **Install Apache Web Server** → Host a webpage.

### Step 1: Create a Security Group

A **Security Group** acts like a firewall. It controls what traffic can enter and leave your EC2 instance.

1. Go to **EC2** → **Security Groups** → **Create Security Group**.
2. Fill details:
  - **Name:** asg-alb-sg
  - **Description:** Security Group for ASG + ALB
  - **VPC:** Default VPC
3. Under **Inbound Rules**, add:
  - **SSH (22)** → Source: My IP (to connect to instance)
  - **HTTP (80)** → Source: Anywhere (0.0.0.0/0)
4. Leave **Outbound Rules** as default (all traffic allowed).
5. Click **Create Security Group**

This allows us to connect via SSH and serve a website.

Screenshot of the AWS EC2 Security Groups page showing the details of a security group named "sg-018a21940624ad48f - asg-alb-sg".

**Details:**

- Security group name: asg-alb-sg
- Security group ID: sg-018a21940624ad48f
- Description: Security group for Auto Scaling + A LB project
- VPC ID: vpc-02e06b741fc8ee298
- Owner: 858265286505
- Inbound rules count: 3 Permission entries
- Outbound rules count: 1 Permission entry

**Inbound rules (3):**

Name	Security group rule ID	IP version	Type	Protocol	Port range
-	sgr-0021c2a067334209a	IPv4	SSH	TCP	22
-	sgr-05ef5469b94a9c697	IPv4	HTTPS	TCP	443
-	sgr-025f870caac2641b4	IPv4	HTTP	TCP	80

Screenshot of the AWS EC2 Security Groups page showing the details of the same security group "sg-018a21940624ad48f - asg-alb-sg".

**Details:**

- Security group name: asg-alb-sg
- Security group ID: sg-018a21940624ad48f
- Description: Security group for Auto Scaling + A LB project
- VPC ID: vpc-02e06b741fc8ee298
- Owner: 858265286505
- Inbound rules count: 3 Permission entries
- Outbound rules count: 1 Permission entry

**Outbound rules (1):**

Name	Security group rule ID	IP version	Type	Protocol	Port range
-	sgr-02a2dc2a563a4c63d	IPv4	All traffic	All	All

## Step 2: Create a Launch Template

A **Launch Template** is a blueprint. It tells AWS how new EC2 instances should be created (OS, size, key, firewall).

## 1. Go to EC2 → Launch Templates → Create Launch Template.

### 2. Fill details:

- **Name:** my-launch-template
- **AMI (OS):** Amazon Linux 2023 (free tier eligible)
- **Instance type:** t2.micro
- **Key pair:** Choose your .pem key (example: sonu-3.pem)
- **Security Group:** Select asg-alb-sg (created above)
- **Storage:** 8GB (default EBS volume)

### 3. Click Create Launch Template

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with navigation links like Dashboard, EC2 Global View, Events, Instances, Images, Elastic Block Store, Network & Security, and CloudShell. The main area displays a table of instances. The table has columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability Zone. Three instances are listed:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
mydb-2	i-096805f91dca3a202	Running	t2.micro	2/2 checks passed	<a href="#">View alarms</a>	us-east-1a
my-launch-te...	i-0d17f7545a71ce02b	Running	t2.micro	2/2 checks passed	<a href="#">View alarms</a>	us-east-1a
ASG-Web-Server	i-00d8912acd90c79a3	Running	t2.micro	2/2 checks passed	<a href="#">View alarms</a>	us-east-1b

Below the table, a specific instance is selected: i-0d17f7545a71ce02b (my-launch-template). The instance summary section shows details like Public IPv4 address (52.90.251.152), Private IP address (172.31.92.2), and Public DNS name (ec2-52-90-251-152.compute-1.amazonaws.com).

Now AWS knows how to launch new EC2 instances automatically.

## Step 3: Create an Auto Scaling Group (ASG)

An **Auto Scaling Group** ensures the right number of EC2 instances are always running. If one crashes, it creates a new one. If load increases, it adds more.

### 1. Go to EC2 → Auto Scaling Groups → Create Auto Scaling Group.

### 2. Select:

- **Launch Template:** my-launch-template
- **Name:** my-asg

### 3. Select VPC: Default VPC

- Choose **at least 2 subnets** in different Availability Zones (e.g. us-east-1a and us-east-1b) → This ensures high availability.

### 4. Under Load Balancing, select:

- **Attach to an existing load balancer** → (We'll create ALB next)

- Or choose **Create new ALB** directly here.

## 5. Configure **Group Size**:

- **Desired capacity:** 1 (start with one instance)
- **Min:** 1
- **Max:** 3 (allows scaling up to 3 instances)

## 6. Add **Scaling Policy**:

- **Target tracking policy** → Keep average CPU utilization at **50%**

## 7. Notifications (Optional):

- Choose or create **SNS topic** → Get emails when instances launch/terminate.

## 8. Add **Tags**:

- Example → Key: Name | Value: ASG-Web-Server

The screenshot shows the AWS EC2 Instances details page for an instance named 'ASG-Web-Server'. The instance ID is i-00d8912acd90c79a3. The instance is currently running. Key details include its Public IPv4 address (98.84.170.86), Private IP DNS name (ip-172-31-28-73.ec2.internal), VPC ID (vpc-02e06b741fc8ee298), and Auto Scaling Group name (my-asg). The instance is managed by the ASG. The left sidebar shows other EC2-related options like Global View, Events, and Launch Templates.

Click **Create Auto Scaling Group**

Now AWS will launch your first EC2 instance automatically.

## Step 4: Create an Application Load Balancer (ALB)

A **Load Balancer** makes sure user traffic is distributed across healthy instances. If one fails, traffic goes to another.

1. Go to **EC2 → Load Balancers → Create Load Balancer → Application Load Balancer**.

2. Fill details:

- **Name:** my-asg-alb
- **Scheme:** Internet-facing
- **IP type:** IPv4

### 3. Network:

- Choose **VPC**: Default
  - Select **2 subnets** in different Availability Zones (same as ASG).

#### 4. Security Group:

- Select **asg-alb-sg** (so it allows HTTP).

## 5. Listener:

- Add **HTTP** on port 80.

## 6. Create Target Group:

- **Name:** my-asg-target-group

- Target type: Instances

- **Protocol:** HTTP:80

- **Health Check Path:** / (root page)

7. Register instances → Select EC2 instances from your ASG.

## 8. Click Create Load Balancer

The screenshot shows the AWS EC2 Load Balancers console. The left sidebar includes sections for Instances, Images, and Elastic Block Store. The main content area displays the details for a load balancer named "my-asg-1". Key information shown includes:

- Load balancer type:** Application
- Status:** Active
- VPC:** vpc-02e06b741fc8ee298
- Load balancer IP address type:** IPv4
- Scheme:** Internet-facing
- Hosted zone:** Z35SXDOTRQ7X7K
- Availability Zones:** subnet-0c3892124977de756 (us-east-1a), subnet-09d0306958fdb94a (us-east-1b)
- Date created:** August 24, 2025, 10:18 (UTC+05:30)
- Load balancer ARN:** arn:aws:elasticloadbalancing:us-east-1:858265286505:loadbalancer/app/my-asg-1/6601d3f6fe1dbdde
- DNS name:** my-asg-1-731819568.us-east-1.elb.amazonaws.com (A Record)

Below the details, tabs for "Listeners and rules", "Network mapping", "Resource map", "Security", "Monitoring", "Integrations", "Attributes", and "Capacity" are visible. A banner at the top right informs users about Application Load Balancers supporting public IPv4 IP Address Management (IPAM).

Screenshot of the AWS CloudWatch Metrics console showing the CloudWatch Metrics Metrics Insights page. The search bar at the top contains the query "CloudWatch Metrics Metrics Insights". The results table lists various metrics, such as "CloudWatch Metrics Metrics Insights" with a value of 1.000000 and a timestamp of 2025-08-24T12:00:00Z.

Screenshot of the AWS CloudWatch Metrics Metrics Insights page. The search bar at the top contains the query "CloudWatch Metrics Metrics Insights". The results table lists various metrics, such as "CloudWatch Metrics Metrics Insights" with a value of 1.000000 and a timestamp of 2025-08-24T12:00:00Z.

Now ALB sits in front of your ASG and distributes traffic.

## Step 5: Install Apache on EC2 Instance

Now we'll install a **web server** (Apache) to serve a webpage.

1. Go to **EC2** → **Instances** → **Select instance** → **Connect**  
(or SSH from terminal: ssh -i sonu-3.pem ec2-user@<Public-IP>).
2. Run these commands one by one:

```
# Update all packages
```

```
sudo yum update -y
```

```
# Install Apache (httpd)
```

```
sudo yum install -y httpd
```

```
# Start Apache
```

```
sudo systemctl start httpd
```

```
sudo systemctl enable httpd
```

```
# Create a custom HTML page
```

```
echo '<h1 style="font-size:50px; color:green; text-align:center;">Hello from Auto Scaling Group Instance</h1>
```

```
<p style="text-align:center; font-size:20px;">This page is served through Apache on an EC2 instance managed by an Auto Scaling Group and Load Balancer.</p>' | sudo tee /var/www/html/index.html
```

Complete!

```
[ec2-user@ip-172-31-28-73 ~]$ sudo yum update -y
sudo yum install -y httpd
sudo systemctl start httpd
sudo systemctl enable httpd
echo '<h1 style="font-size:50px; color:green; text-align:center;">Hello from Auto Scaling Group Instance</h1><p style="text-align:center; font-size:20px;">This page is served through Apache on an EC2 instance managed by an Auto Scaling Group and Load Balancer.</p>' | sudo tee /var/www/html/index.html
Last metadata expiration check: 0:45:29 ago on Sun Aug 24 05:23:27 2025.
```

=====

**WARNING:**  
A newer release of "Amazon Linux" is available.

**Available Versions:**

**Version 2023.8.20250808:**  
Run the following command to upgrade to 2023.8.20250808:  
  
`dnf upgrade --releasever=2023.8.20250808`

**Release notes:**  
<https://docs.aws.amazon.com/linux/al2023/release-notes/relnotes-2023.8.20250808.html>

**Version 2023.8.20250818:**  
Run the following command to upgrade to 2023.8.20250818:  
  
`dnf upgrade --releasever=2023.8.20250818`

**Release notes:**  
<https://docs.aws.amazon.com/linux/al2023/release-notes/relnotes-2023.8.20250818.html>

**i-00d8912acd90c79a3 (ASG-Web-Server)**

PublicIPs: 98.84.170.86 PrivateIPs: 172.31.28.73

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Safari File Edit View History Bookmarks Window Help us-east-1.console.aws.amazon.com Sun 24 Aug 11:39 AM

Instance details | EC2 | us-east-1 Load balancer details | EC2 | us-east-1 Target group details | EC2 | us-east-1 EC2 Instance Connect | us-east-1 my-asg-1-731819568.us-east-1.elb.am... Account ID: 8582-6528-6505 shnthy\_45

AWS Search [Option+S]

Complete!

```
[ec2-user@ip-172-31-28-73 ~]$ sudo yum update -y
sudo yum install -y httpd
sudo systemctl start httpd
sudo systemctl enable httpd
echo '<h1 style="font-size:50px; color:green; text-align:center;">Hello from Auto Scaling Group Instance</h1><p style="text-align:center; font-size:20px;">This page is served through Apache on an EC2 instance managed by an Auto Scaling Group and Load Balancer.</p>' | sudo tee /var/www/html/index.html
Last metadata expiration check: 0:45:30 ago on Sun Aug 24 05:23:27 2025.
```

=====

**Version 2023.8.20250808:**  
Run the following command to upgrade to 2023.8.20250808:  
  
`dnf upgrade --releasever=2023.8.20250808`

**Release notes:**  
<https://docs.aws.amazon.com/linux/al2023/release-notes/relnotes-2023.8.20250808.html>

**Version 2023.8.20250818:**  
Run the following command to upgrade to 2023.8.20250818:  
  
`dnf upgrade --releasever=2023.8.20250818`

**Release notes:**  
<https://docs.aws.amazon.com/linux/al2023/release-notes/relnotes-2023.8.20250818.html>

Dependencies resolved.  
Nothing to do.  
Complete!  
Last metadata expiration check: 0:45:30 ago on Sun Aug 24 05:23:27 2025.  
Package httpd-2.4.62-1.amzn2023.x86\_64 is already installed.  
Dependencies resolved.  
Nothing to do.  
Complete!  
<h1 style="font-size:50px; color:green; text-align:center;">Hello from Auto Scaling Group Instance</h1><p style="text-align:center; font-size:20px;">This page is served through Apache on an EC2 instance managed by an Auto Scaling Group and Load Balancer.</p>

[ec2-user@ip-172-31-28-73 ~]\$

**i-00d8912acd90c79a3 (ASG-Web-Server)**

PublicIPs: 98.84.170.86 PrivateIPs: 172.31.28.73

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms

Now your EC2 is serving a webpage.

## Step 6: Test Your Setup

1. Copy the **DNS of your Load Balancer** (looks like my-asg-1-731819568.us-east-1.elb.amazonaws.com).
2. Open it in your browser.
3. You should see:

**"Hello from Auto Scaling Group Instance"** in **big green letters**.

If you refresh multiple times, ALB may send requests to different EC2 instances.

If one instance fails, ASG will replace it automatically.

