Guided Lab: Setting Up and Managing a Database on an Amazon EC2 Instance

Description

Amazon EC2 (Elastic Compute Cloud) provides a versatile and scalable environment for deploying various applications and services, including relational databases. When setting up a database, you have the option to deploy it directly on an EC2 instance or use Amazon RDS (Relational Database Service), a managed database solution provided by AWS.

Deploying a database on an EC2 instance offers full control over the database environment. This means you handle the installation, configuration, and management of the database software yourself. This approach is ideal for custom setups, advanced configurations, and scenarios where you require complete control over the database.

On the other hand, Amazon RDS abstracts much of the database management complexity. With RDS, AWS manages the underlying database software, including installation, patching, backups, and scaling. This allows you to focus more on using the database rather than managing its infrastructure, making it a great choice for many standard use cases.

In this guide, we'll walk you through the process of setting up a relational database on an EC2 instance. You'll learn how to launch an EC2 instance, install and configure one of the popular database software MySQL. We'll also briefly touch on how this approach differs from using Amazon RDS, helping you choose the best option for your needs.

Prerequisites

This lab assumes you have basic knowledge of Amazon EC2 instance (Linux), MySQL Relational Database and SSH client (e.g. GitBash, PuTTy, etc.)

If you find any gaps in your knowledge, consider taking the following lab:

- **Creating an Amazon EC2 instance (Linux)**
- **Creating an Amazon RDS database**

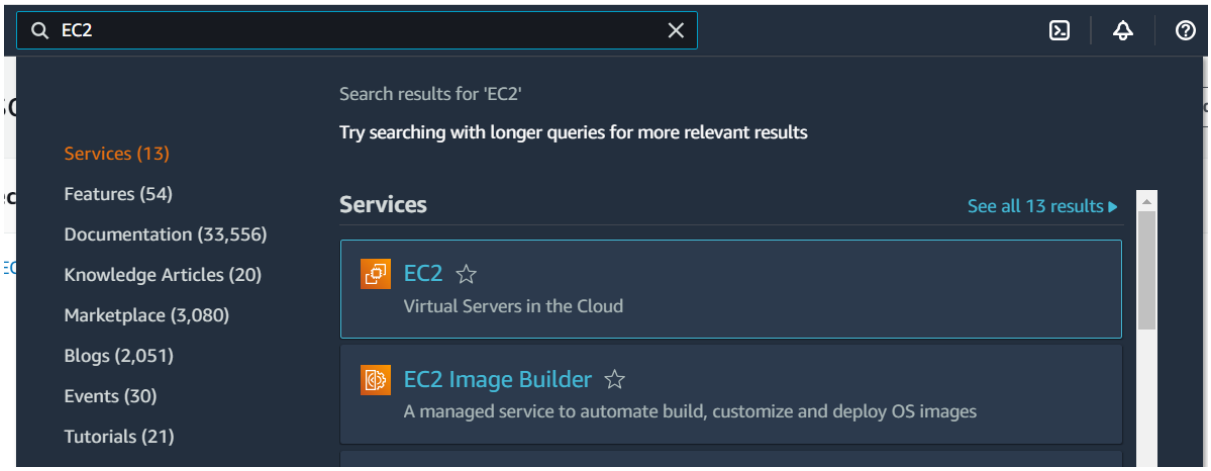
Objectives

In this lab, you will:

- **Set up a relational database software MySQL on an EC2 instance.**
- **Optimize it for performance and security.**
- **Differentiate Amazon RDS from Self-managed Relational Database Installed in EC2 Instance**

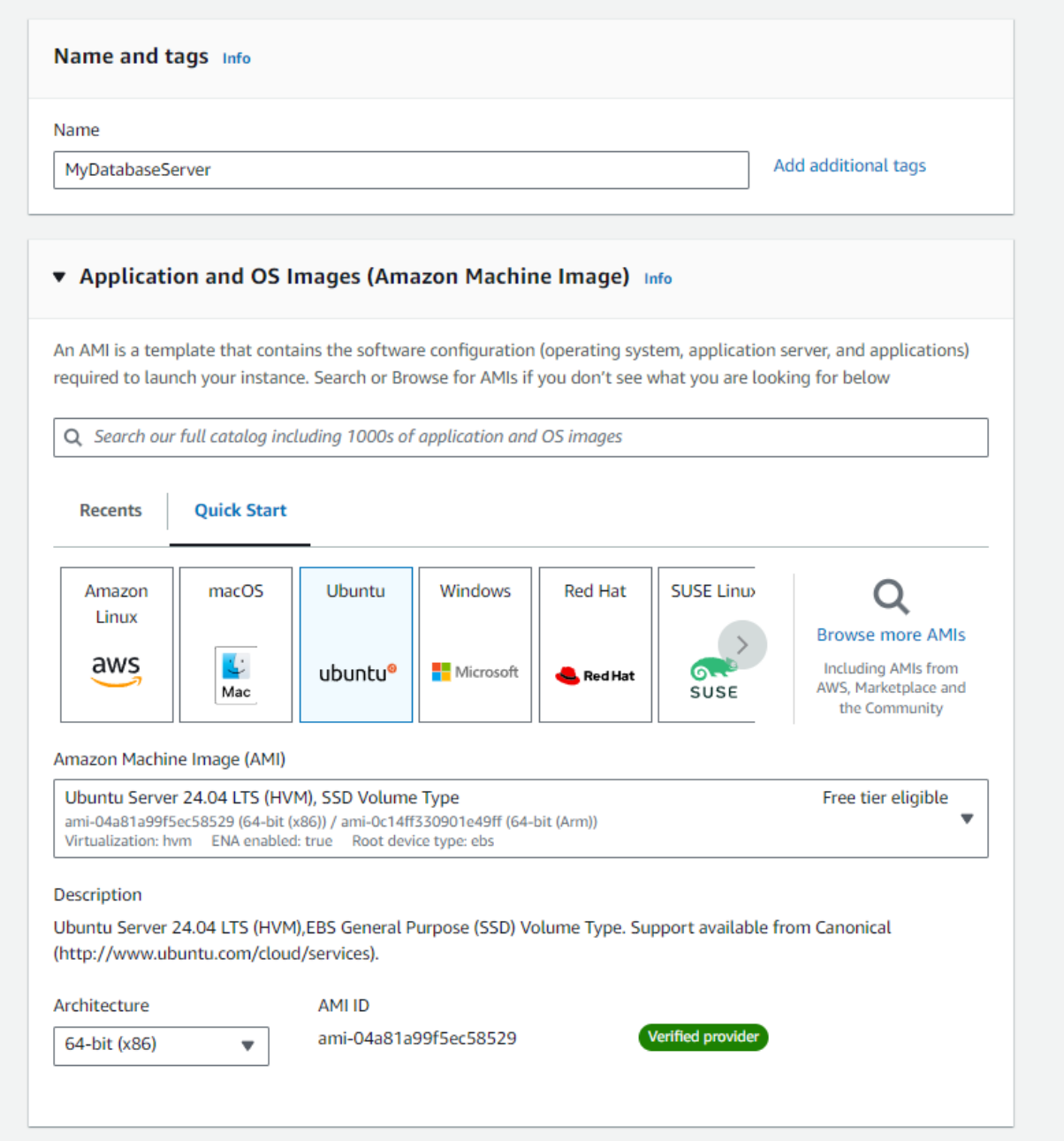
Launch your Instance

1. **Navigate to EC2 Dashboard**

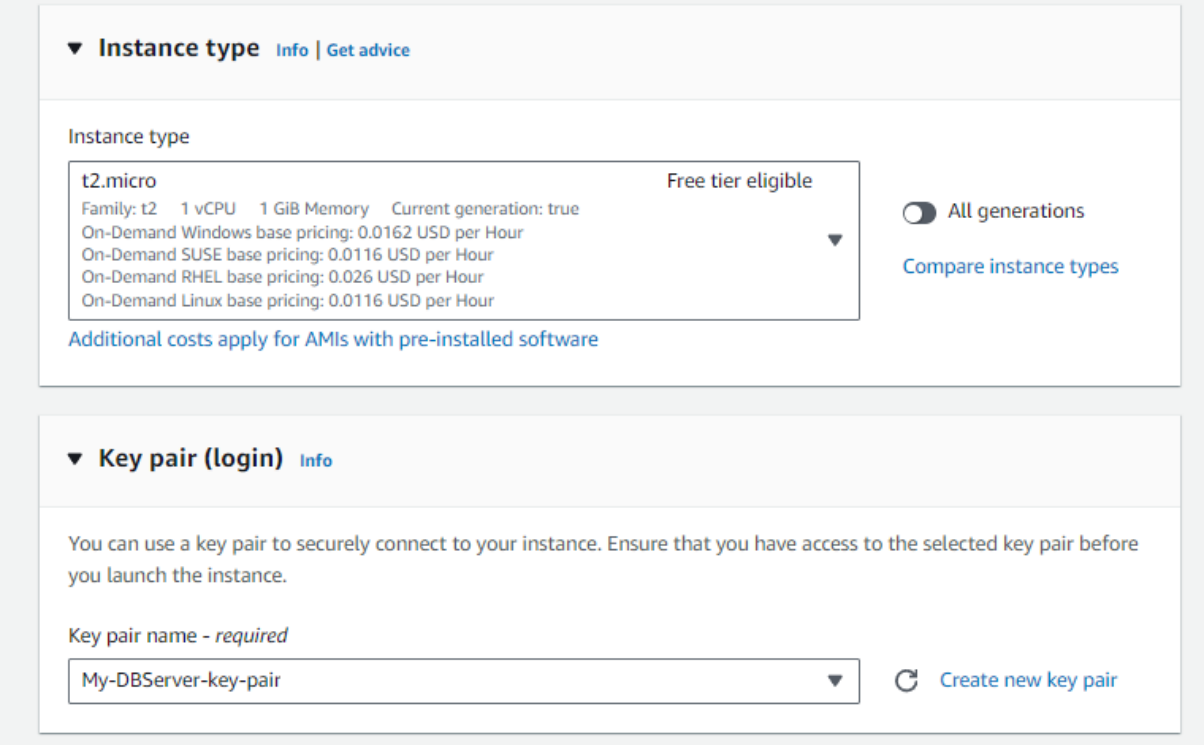


2.**Launch an EC2 Instance with the following configuration:**

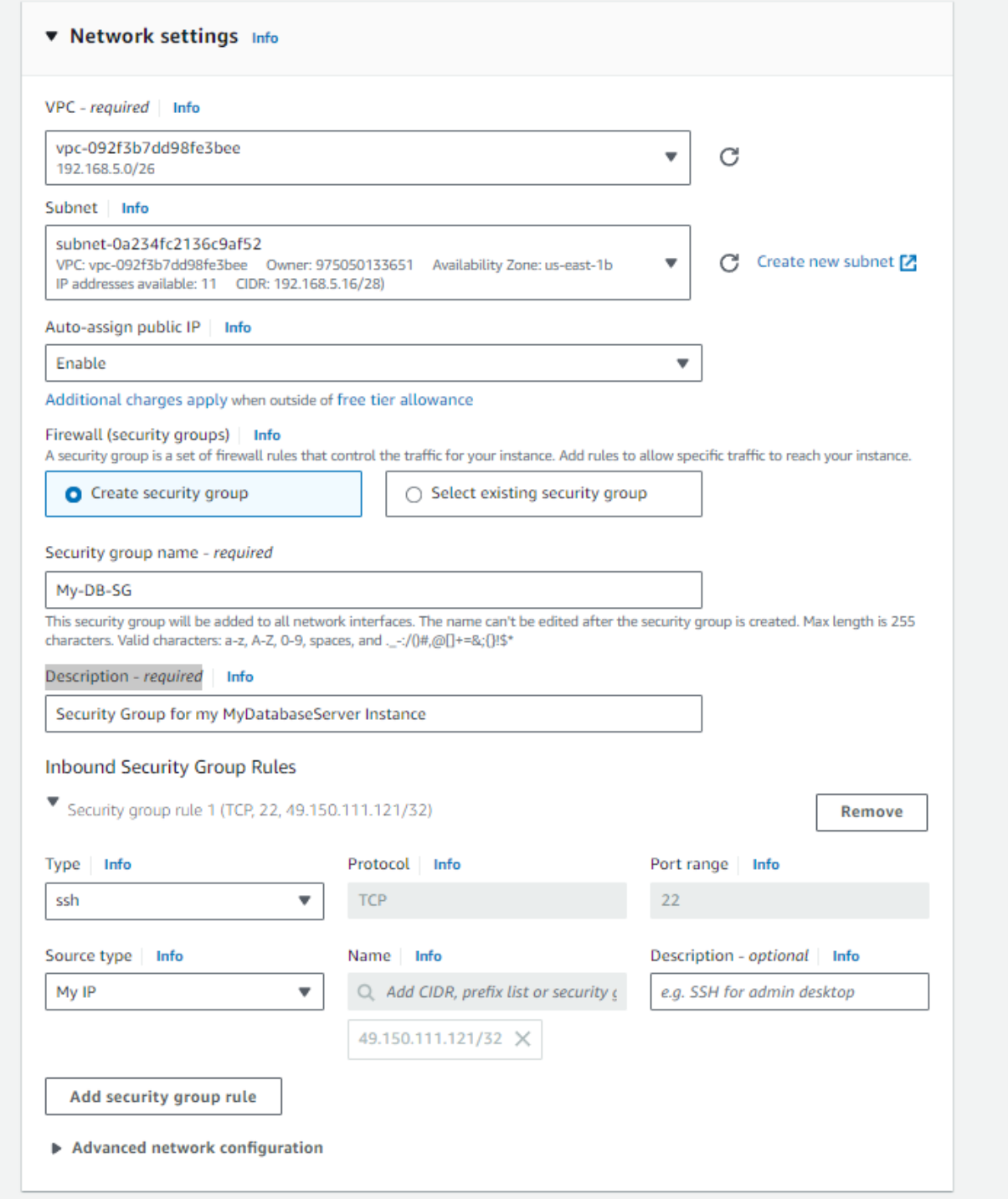
- **Name:** MyDatabaseServer
- **AMI:** Ubuntu Server 24.04 LTS



- **Instance type:** t2.micro
- **Key pair:** (Please **create a new one.**)
    - **Key pair name:** My-DBServer-key-pair
    - **Key pair type:** RSA
    - **Private key file format:** .pem



- **Network settings:** (Click on **Edit**)
    - Leave **VPC** and **Subnet** with the default
    - **Auto-assign public IP:** Select Enable
    - **Firewall (security groups):** (Select the radio button **Create security group**)
        - **Security group name – *required:*** My-DB-SG
        - **Description – *required***: Security Group for my MyDatabaseServer Instance
    - **Allow SSH traffic from :** My IP



- Click **Launch instance**

nstalling MySQL Database Software

1. After launching the instance, connect to it via SSH.

2. Use the following command to connect:

- Navigate where you downloaded your key-pair.pem

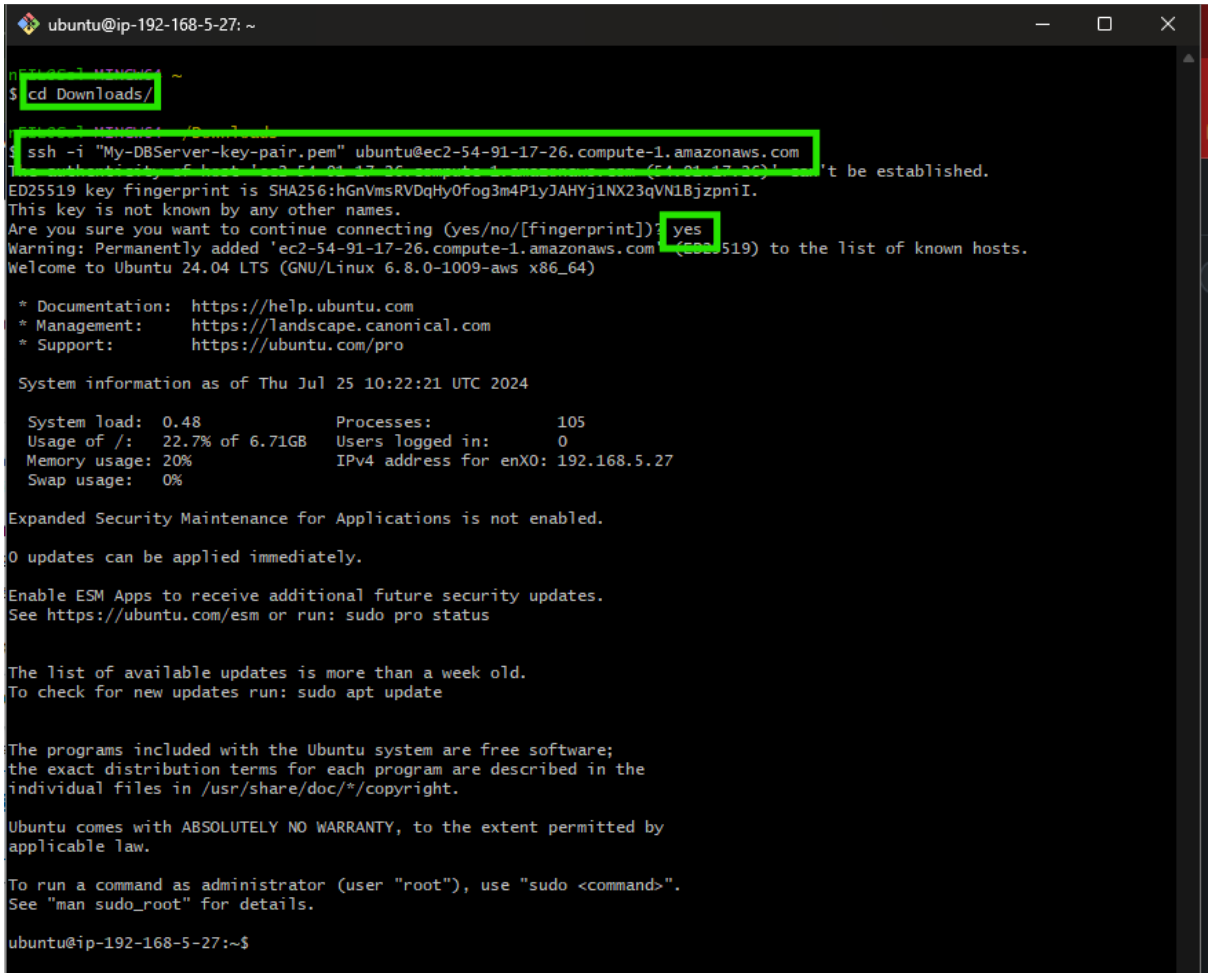
Do not forget to change the <placeholders>

cd <directory-of-the-key-pair>

- Paste the following command and resposnse yes when question prompted

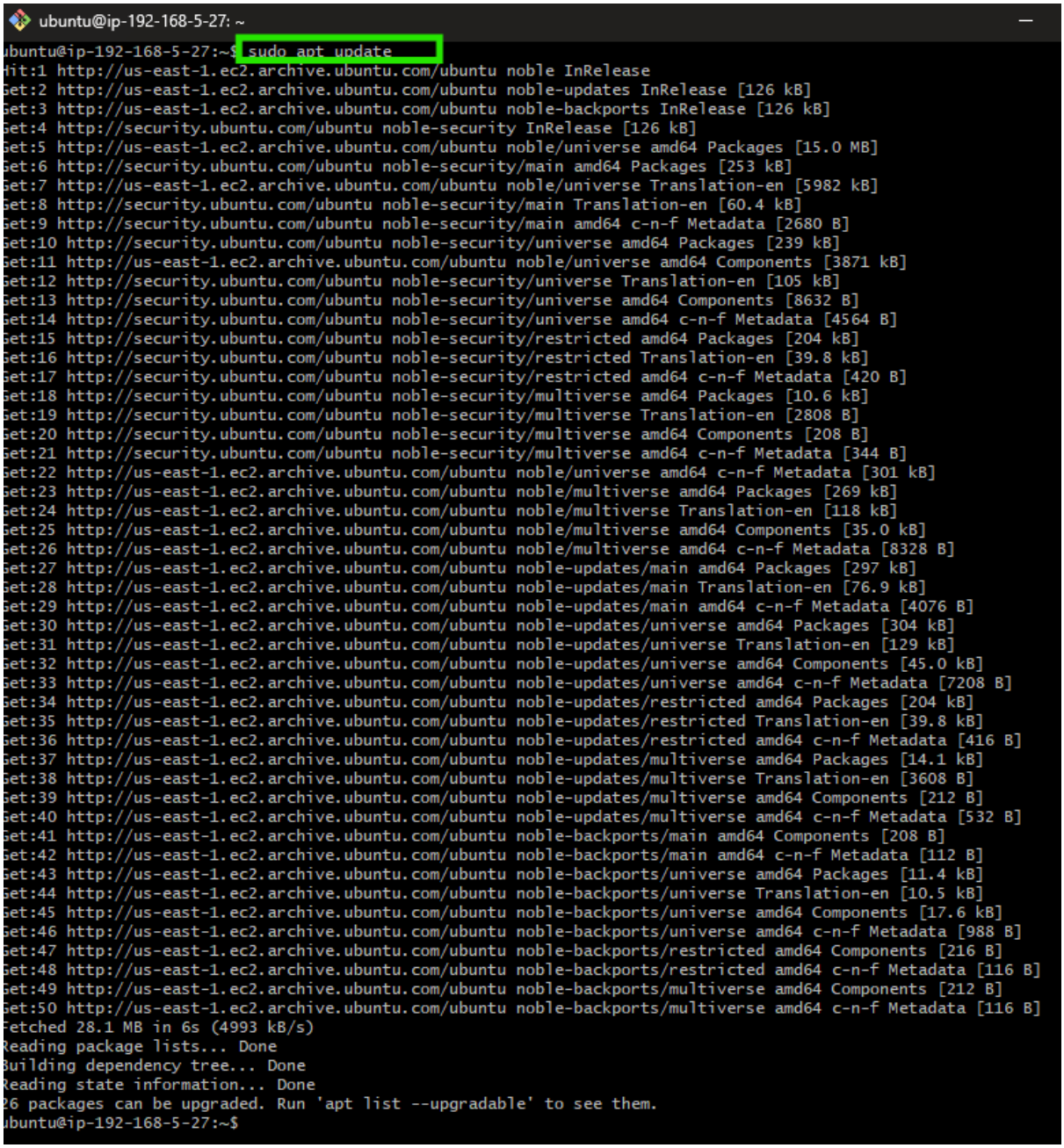
Do not forget to change the <placeholders>

ssh -i <your-key.pem> <ec2-user@your-ec2-public-dns>

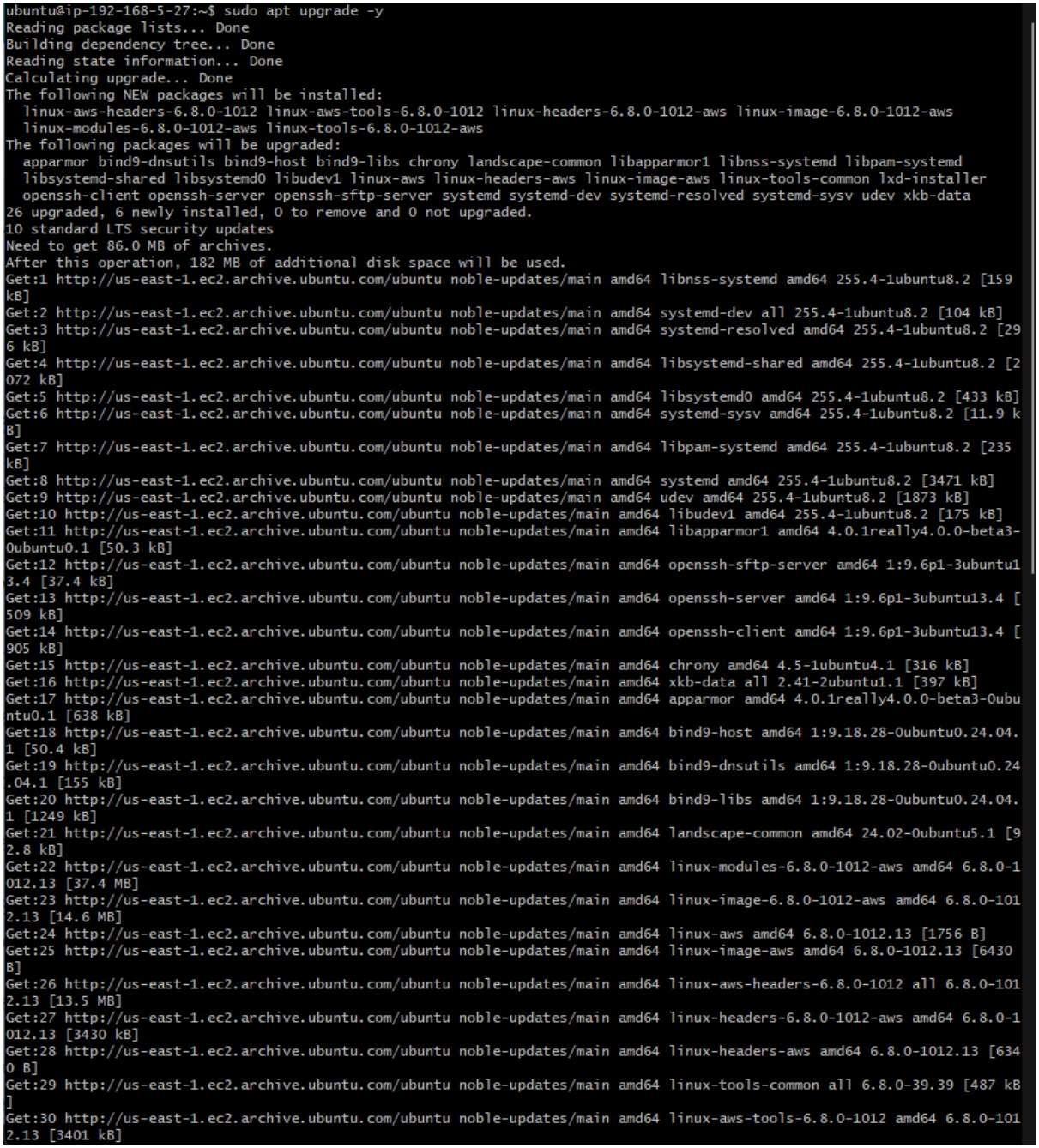


3. It's always a good practice to update your package lists and upgrade existing packages. Copy and Paste the following command:

sudo apt update



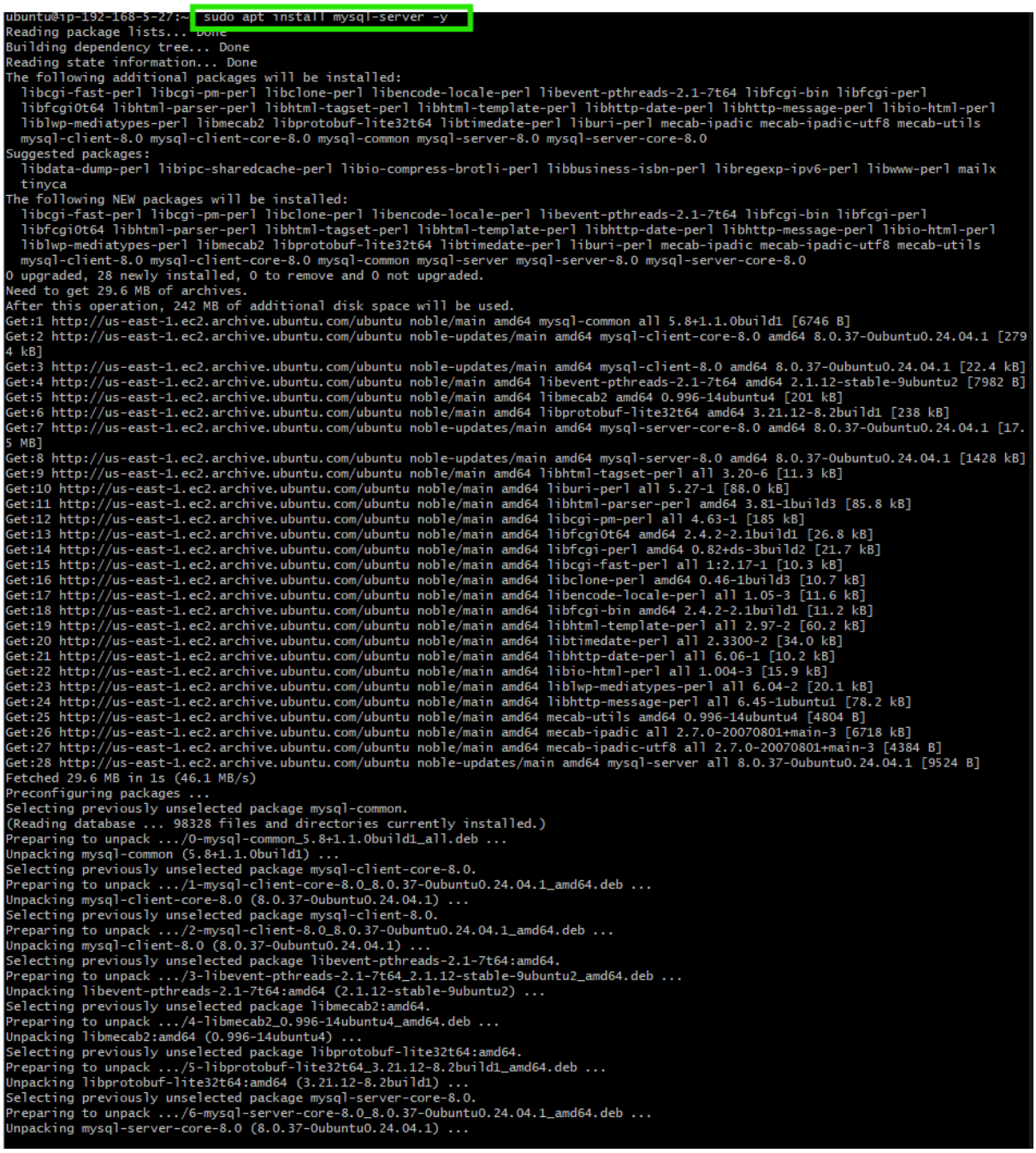
sudo apt upgrade -y



Wait for it to finished.

4. Install MySQL Server using the command below and wait for it to finished installing:

sudo apt install mysql-server -y



5. Secure MySQL Installation

While you can install MySQL without this step, it is best practice to add security measures when installing your MySQL server in the production environment.

a. Run the security script to improve the security of your MySQL installation:

sudo mysql_secure_installation

```
ubuntu@ip-192-168-5-27:~$ sudo mysql_secure_installation

Securing the MySQL server deployment.

Connecting to MySQL using a blank password.

VALIDATE PASSWORD COMPONENT can be used to test passwords
and improve security. It checks the strength of password
and allows the users to set only those passwords which are
secure enough. Would you like to setup VALIDATE PASSWORD component?

Press y|Y for Yes, any other key for No:
```

b. Review and follow the prompts to set the root password and secure the installation.

The following response was chosen to simplify the lab. You can choose any response as you desired.

```
ubuntu@ip-192-168-5-27: ~
ubuntu@ip-192-168-5-27:~$ sudo mysql_secure_installation

Securing the MySQL server deployment.

Connecting to MySQL using a blank password.

VALIDATE PASSWORD COMPONENT can be used to test passwords
and improve security. It checks the strength of password
and allows the users to set only those passwords which are
secure enough. Would you like to setup VALIDATE PASSWORD component?

Press y|Y for Yes, any other key for No: y

There are three levels of password validation policy:

LOW    Length >= 8
MEDIUM Length >= 8, numeric, mixed case, and special characters
STRONG Length >= 8, numeric, mixed case, special characters and dictionary                  file

Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG: 1

Skipping password set for root as authentication with auth_socket is used by default.
If you would like to use password authentication instead, this can be done with the "ALTER_USER" command.
See https://dev.mysql.com/doc/refman/8.0/en/alter-user.html#alter-user-password-management for more information.

By default, a MySQL installation has an anonymous user,
allowing anyone to log into MySQL without having to have
a user account created for them. This is intended only for
testing, and to make the installation go a bit smoother.
You should remove them before moving into a production
environment.

Remove anonymous users? (Press y|Y for Yes, any other key for No) : N

 ... skipping.


Normally, root should only be allowed to connect from
'localhost'. This ensures that someone cannot guess at
the root password from the network.

Disallow root login remotely? (Press y|Y for Yes, any other key for No) : y
Success.

By default, MySQL comes with a database named 'test' that
anyone can access. This is also intended only for testing,
and should be removed before moving into a production
environment.


Remove test database and access to it? (Press y|Y for Yes, any other key for No) : N

 ... skipping.
Reloading the privilege tables will ensure that all changes
made so far will take effect immediately.

Reload privilege tables now? (Press y|Y for Yes, any other key for No) : y
Success.

All done!
ubuntu@ip-192-168-5-27:~$
```

6. Access MySQL:

sudo mysql -u root -p

- -u root: The -u flag specifies the MySQL user you want to log in as. In this case, root is the user with administrative privileges in MySQL.
- -p: This flag tells the MySQL client to prompt you for a password.

If prompted for a password type a password accordingly for example H@ppyL34rn1ng

```
ubuntu@ip-192-168-5-27:~$ sudo mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.37-0ubuntu0.24.04.1 (Ubuntu)

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql>
```

7. Now, you can start creating your Database and User

Create a Database and User

1. **Create a New Database**

- Once you're logged in and at the MySQL prompt (mysql>), Replace mydatabase with your desired database name:

SHOW DATABASES;

```
MySQL [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.007 sec)

MySQL [(none)]>
```

- Create a new database using the CREATE DATABASE statement.

Do not forget to change the placeholder <database-name-desired> *for the name of database you want.*

Create DATABASE <database-name-desired>;

```
mysql> Create DATABASE EmployeesDB;
Query OK, 1 row affected (0.01 sec)

mysql>
```

- To check if this was created we can use the same command:

SHOW DATABASES;

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| EmployeesDB        |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.03 sec)

mysql>
```

2. **Create a New User**

- Lets check and display the list of default users, first, and review them:

SELECT User, Host FROM mysql.user;

```
mysql> SELECT User, Host FROM mysql.user;
+------------------+-----------+
| User             | Host      |
+------------------+-----------+
| debian-sys-maint | localhost |
| mysql.infoschema | localhost |
| mysql.session    | localhost |
| mysql.sys        | localhost |
| root             | localhost |
+------------------+-----------+
5 rows in set (0.00 sec)

mysql>
```

- Now, create a new user with the CREATE USER statement.

Do not forget to change the placeholder <user-name> and <password> *with your preferences.*
And do not forget to follow the password validation policy you set earlier (for example, if choose 1 = MEDIUM, it must have numeric, mixed case, and have special case)

CREATE USER '<user-name>'@'localhost' IDENTIFIED BY '<password>';

```
mysql> CREATE USER 'User-Jose'@'localhost' IDENTIFIED BY 'myPasswordis123!';
Query OK, 0 rows affected (0.02 sec)

mysql>
```

- Check the list of users again, using the command:

SELECT User, Host FROM mysql.user;

```
mysql> SELECT User, Host FROM mysql.user;
+------------------+-----------+
| User             | Host      |
+------------------+-----------+
| User-Jose        | localhost |
| mysql.infoschema | localhost |
| mysql.session    | localhost |
| mysql.sys        | localhost |
| root             | localhost |
+------------------+-----------+
6 rows in set (0.00 sec)

mysql>
```

3. Grant the necessary privileges to the new user for the database. This example grants all privileges on the EmployeesDB database to User-Jose:

Do not forget to change the placeholders

GRANT ALL PRIVILEGES ON <user-name>.* TO '<user-name>'@'localhost';

```
mysql> GRANT ALL PRIVILEGES ON EmployeesDB.* TO 'User-Jose'@'localhost';
Query OK, 0 rows affected (0.01 sec)

mysql>
```

4. To apply the changes, run the FLUSH PRIVILEGES command:

FLUSH PRIVILEGES;

```
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

5. To exit the MySQL prompt, use:

exit

```
mysql> exit
Bye
```

That's it! Congratulations! You have successfully set up a relational database on an EC2 instance by installing and configuring MySQL. This exercise provided a foundational understanding of deploying and managing a self-hosted database environment, including installation, security, and user management.

This lab highlighted the flexibility and control offered by a self-managed approach. However, it's important to note that this is just the beginning. There are many advanced configurations and optimizations available for self-managed databases, and comparing this with Amazon RDS can help you decide the best approach for your needs.